ARISTOTLE UNIVERSITY OF THESSALONIKI

Machine Learning Approaches for Time Series Problems

by

Christoforos Nalmpantis

A thesis submitted in partial fulfilment for the
degree of Doctoral of Philosophy

in the
Faculty of Sciences
School of Informatics

July 6, 2022

ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF SCIENCES
SCHOOL OF INFORMATICS

Doctor of Philosophy

by Christoforos Nalmpantis

Dedicated to my family, Apostolina and Stefania.

ABSTRACT

This thesis presents original research in the intersection of machine learning and time series. Time series data is everywhere in nature or our daily life. Every sequence of data where the order in terms of time matters consists of a time series. This type of data has unique properties. Time series data emerge from the dynamics of complex systems and are high dimensional. These aspects bring out many new challenges when trying to extract useful information from time series data using machine learning methods.

We focus on both practical and theoretical aspects of machine learning methods when dealing with time series data. The three main pillars of our contributions include novel machine learning architectures, time series embedding representations and designing deep learning models based on information theoretic principles. Our research goals are driven by three common time series problems such as blind source separation, classification and regression. The domains of the data that are used in this work span energy, speech and health. Regarding the first domain we selected the problem of non-intrusive load monitoring (NILM). From the domain of speech the problems that are tackled are speech commands classification and speaker recognition. Finally, in the domain of health we conducted experiments related to mining electrocardiograms (ECGs) and bone age assessment. One of our first contributions is a comprehensive investigation of the aforementioned real-world problems, where we identified research opportunities.

The first pillar of our scientific contributions comprises novel machine learning solutions for the problem of NILM. We propose a new stacking machine learning system, a transfer learning mechanism that utilises imaging techniques for time series and three novel neural architectures. Our methods are compared against existing ones and demonstrate state-of-the-art results. Furthermore, we address several evaluation issues by introducing new evaluation measures/approaches and developing new benchmark frameworks.

The second major contribution of this research regards time series representations. Time series representation has been an active research direction the last few decades. With the rise of the internet of things (IoT) and the explosion of time series data from various sensors, it is apparent that efficient representations without losing useful information are imperative. We propose a novel time series embedding representation, named Signal2Vec. Signal2Vec is an unsupervised approach to transform a given time series into vectors. The proposed algorithm is compared against existing time series representation methods from the domain of signal processing and chaos theory. The evaluation tasks are a multiclass classification of energy data and the problem of NILM seen as a multilabel classification approach. Signal2Vec outperforms the other methods and demonstrates state-of-the-art results on the above-mentioned tasks.

Our third scientific addition touches two research disciplines, deep learning and information theory. Information theory has been a powerful tool for machine learning since the first steps of the field. Nowadays, information theoretic principles are utilised in order to bridge the gap between theory and practice in deep learning. Despite the success of deep learning in many domains, designing such models relies on the process of trial and error. A rigorous mathematical theory of overparameterized models is still missing. Our efforts focus on a popular component of modern neural architectures, called pooling and includes both theoretical and empirical outcomes. We study existing pooling operations via the lens of information theory and try to answer the question of which function should be chosen when designing a new neural network given a task and some data. Then, we develop two novel pooling operations based on the principle of maximum entropy and the principle of information bottleneck. Our theoretical outcomes are verified via experimentation. The proposed methods are compared against others and show superior performance in classification and regression tasks, when solving problems in the areas of speech and health.

This scientific work is a stepping stone towards more reliable and efficient machine learning methods for time series problems. We hope that our findings will motivate future researchers and will serve as tools for engineers in high impact industrial applications.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Prof. Dimitris Vrakas for all the support and guidance he has offered me throughout the thesis. His assistance has been invaluable all these years. He offered me a great academic environment, surrounded by bright colleagues and always tried to support me financially. I would also like to thank him for his trust and the academic freedom regarding my first steps in research. I wish to further collaborate with him in the near future.

In addition, I would like to thank my other two advisors, Prof. Ioannis Vlahavas and Prof. Grigorios Tsoumakas for their scientific knowledge, their honest feedback and career advice that they provided. I would also like to express my gratitude to my co-authors, Odysseas Krystalakos, Nasos Lentzas, Lamprini Kyrkou, Nikolaos Symeonidis, Lazaros Vrysis and Nikolaos Virtsionis Gkalinikis, for the excellent collaboration and for their contributions. Being part of the Intelligent Systems group in the School of Informatics offered me not only knowledge, but also the opportunity to meet great friends and colleagues, who I wish to express my appreciation for their humor and their smile all these years.

Doing research was one of my ambitions which this thesis comes to fulfil, but I would never be able to achieve without the support and love of my parents Stefanos and Isaia, my sister Maria and my brother Aggelos. Finally, I wish to thank my wife Apostolina for her patience, the continuous support and encouraging me to reach my goals. I dedicate this thesis to my family: Apostolina and Stefania.

This thesis was partially co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH–CREATE–INNOVATE (project code: T1EDK-00343(95699)-Energy Controlling Voice Enabled Intelligent Smart Home Ecosystem).

CONTENTS

List of Figures

List of Tables

1 Introduction

"You can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future."
— Steve Jobs

When the human brain is learning activities like language, vision, and motion, time is a natural element that is always present. Most real-world data have a temporal component, whether they are measurements of natural processes (such as weather or sound waves) or measures of man-made activities (stock market, robotics). Analysis of time-series data has been the subject of active research for decades and is regarded as one of the top challenges in AI due to its unique characteristics.

This thesis addresses issues of applying modern AI techniques when dealing with dynamic complex systems. The research goal is to develop novel machine learning methodologies in order to overcome common obstacles of time series problems. The area of research lies in the intersection of machine learning, time series representations and time series analysis. This section briefly introduces these research areas and presents the main contributions of the thesis.

1.1 Research Areas

The contributions of this work span three fundamental research areas: machine learning, learning representations and time series analysis. This section introduces the subject of each of these domains and presents current research directions.

1.1.1 Machine Learning

The era of big data is upon us and is going to expand even more with the rise of Internet of Things (IoT). There are more than 1 trillion web pages; one hour of video is uploaded to YouTube every second, equivalent to ten years of content per day; the genomes of thousands of people have been sequenced by various laboratories; Walmart processes over 1 million transactions per hour and maintains databases containing more than 2.5 petabytes of data (Cukier, 2010); and so on.

Machine learning is the tool that offers automatic data analysis in order to cope with this explosion of data. More specifically, machine learning is defined as a collection of methods and tools for automatic pattern detection in data and then using these observations to forecast future behaviour or make other types of uncertain decisions (Murphy, 2012).

The two main pillars of machine learning are supervised and unsupervised learning. The former one is a predictive approach aiming to learn a mapping from some data x to some targets y. In this case the training set $\mathcal{D}$ consists of N pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$. Each training input x_i can be as simple as a D-dimensional vector, where the dimensions are also called features. In real world the data inputs can be more complex structures such as images, text, time series, chemical compounds, a graph structure and others. In the same way, the targets y_i can have any form of data. In the simplest cases the targets are categorical data from a finite set or scalars from a continuous space. Depending on the type of the target, a machine learning task can be classified as classification when data are categorical or regression when data belong to a continuous space. There are also other variants such as ordinal regression when there is a natural order of the values. As far as unsupervised learning is concerned it is also known as descriptive learning and the model is given only the input data $\mathcal{D} = \{x_i\}_{i=1}^{N}$ aiming to automatically find patterns that characterise the data as a population. This approach is also called knowledge discovery, because the goal of the algorithm is to automatically discover the knowledge that underlies some observations. Unsupervised learning is not a well defined problem and we don't know a priory what kind of knowledge will be revealed (Murphy, 2012). Apart from the two main types of machine learning, there is a third one which lies in the intersection of machine learning and decision theory, named reinforcement learning. Reinforcement learning is a method useful for tasks where the goal is to learn how to act or take a decision within an interactive environment. The learning agent gets feedback from the environment in the form of a reward or a punishment depending on the target task.

Supervised learning is one of the most popular types of machine learning because of its large impact in the industry. Starting with classification, the goal is to learn a mapping from inputs x to outputs y, with $y \in \{1, ..., C\}$ and C the number of classes. It is very common to have only two classes and then we have the so called binary classification problem. For more than two classes the problem is called multiclass classification. If the class labels are not mutually exclusive, we refer to the problem as multi-label classification, but it is more accurately described as predicting multiple related binary class labels or multiple output model. In the literature the term classification refers to multiclass classification with a single output. The problem can be formalized as a function approximation, by assuming that $y = f(x)$ for an unknown function f. Then, the objective of learning is to estimate the function f given a labeled training set. After finding a function f, it can be used to make predictions $\widehat{y} = \widehat{f}(x)$, where the hat symbol indicates an estimate. The primary goal is to make predictions on unseen inputs, that is, ones that we have not been encountered previously. This is known in machine learning as generalization.

In order for a machine learning model to generalize well on unseen data, making hard predictions is not ideal because there should be a way to measure the uncertainty of the model. the best way to solve such problems is to use the tools of probability theory. Given an input vector x and training set $\mathcal{D}$, the probability distribution over the space of targets y is $p(y|x, \mathcal{D})$. When there are many models and we want to include them in the probability distribution we denote $p(y|x, \mathcal{D}, M)$, with M representing a model. In this context the estimation of the model is the maximum a posteriori (MAP), which corresponds to the class label with the maximum probability and is denoted as follows:

$$\widehat{y} = \widehat{f}(x) = \arg\max_{c=1}^{C} p(y = c|x, \mathcal{D}) \tag{1.1}$$

Some popular applications of supervised learning span document classification, email spam filtering, image classification, keyword spotting in speech, handwriting recognition, face detection and many others.

When the target output of a machine learning model is in continuous space, we have a regression problem. The goal is to map a real values $x_i \in \mathcal{R}$ to another real value $y_i \in \mathcal{R}$. The simplest case of regression is fitting a straight line or a quadratic function. Numerous expansions to this fundamental problem are possible, including high-dimensional inputs, outliers, and non-smooth responses. Some examples of regression include the following:

- Forecasting the temperature for a specific area, given historical data of the previous days.

- Forecasting the price of a stock for the next day, given historical data and the price of other stocks in the market.

- Predict the time that a user might spend on a website, given the user's age, sex, education and types of websites that she visited in the past.

- Predict the blood pressure of a human given a number of clinical measurements.

In unsupervised learning a model is provided with pure data observations and there is no specific target. The objective is to uncover hidden structures within the data a task that is occasionally referred to as knowledge discovery. In contrast to supervised learning, we are not informed of the desired output for each input. The task is formally described as estimating a density $p(x_i|\theta)$. In the context of probability theory, supervised learning corresponds to conditional density estimation, whereas unsupervised learning corresponds to unconditional density estimation. Unsupervised learning is, in many ways, more representative of human and animal cognition. Since there is no need for human expertise to manually label the data, the applicability of unsupervised learning covers a much wider spectrum of problems. Labeled data is not only difficult to obtain, but it also provides a limited amount of information, which is insufficient to properly estimate the parameters of complex models. On the other hand, unsupervised learning is considered much harder than supervised one.

A typical problem of unsupervised learning is clustering data into groups. Usually, there is no evidence on how many groups exist. Let the number of groups or clusters be denoted as K. The initial goal is to estimate the probability distribution $p(K|\mathcal{K})$, which is often approximated by its mode $K^* = \arg\max_K p(K|\mathcal{D})$. In contrast to the supervised case where the number of categories is fixed and known a priory, in the unsupervised case the number of clusters can be an assumption. Next, each data point should be grouped into the most appropriate cluster denoted as $z_i \in \{1, ..., K\}$. The variable z is an example of hidden or latent variable that is not directly observed. The cluster of each data point is estimated by the formula $z^* = \arg\max_k p(z_i = k|x_i, \mathcal{D})$. Some applications of clustering are listed below:

- In an application such as Netflix, the users can be grouped into categories depending on the type of movies they watch.

- Given some clinical measurements a group of people can be clustered into different categories representing the risk of their health when they are infected by a specific virus e.g. Covid19.

- In biology, cells can be grouped into subgroups based on flow-cytometry.

Another important example of unsupervised learning is the problem of dimensionality reduction. When dealing with large amounts of high-dimensional data, it is frequently beneficial to reduce the dimensionality of the data by projecting it to a lower-dimensional subspace that captures the key characteristics of the data. The rationale for using this technique is that, despite the fact that the data appears to be high dimensional, there may only be a small number of degrees of variability, which correspond to latent components, in the data. If you are modeling the appearance of a facial image, for example, it is possible that there are only a few underlying latent elements that account for the majority of the variability. These factors include lighting, position, identity, and so on.

As input to other statistical models, low-dimensional representations of objects frequently produce superior predicted accuracy than high-dimensional representations because they concentrate on the key characteristics of the object while filtering away inessential aspects. Another advantage of using low-dimensional representations is that they allow for faster nearest neighbor searches, whereas two-dimensional projections are quite beneficial for viewing high-dimensional data. Principal component analysis, known as PCA, is the most widely used method of dimensionality reduction in engineering. This can be regarded as an unsupervised variant of multi-output linear regression, in which we observe the high-dimensional response y, but not the low-dimensional "cause" z, as opposed to the supervised version. Dimensionality reduction and more specifically PCA, has been employed in several real-world applications including:

- It is possible to utilize a technique known as spike-triggered covariance analysis, which is a version of Principal Components Analysis in Neuroscience, to determine the specific characteristics of an external stimulus that increase the likelihood of a neuron generating an action potential.
- Latent semantic analysis is a variant of PCA. It is defined as the process of examining relationships between a set of documents and the terms they include by constructing a set of ideas that are related to the documents and the terms. It is also known as distributional semantics in natural language processing.
- In signal processing independent component analysis (ICA) is used for the problem of blind source separation.
- Projecting motion capture data to a low-dimensional environment and then using that data to create animations is widespread practice in computer graphics.

Sometimes we assess a group of associated variables and want to know which ones are more correlated with which others. This is known as correlation analysis. Given a graph G, nodes represent variables and edges indicate direct dependence between variables. This problem is formally described by $\widehat{G} = \arg\max p(G|\mathcal{D})$. Just as there are two main applications for unsupervised learning in general, there are two main applications for learning sparse graphs: discovering new information and improving joint probability density estimators. Some examples of applications are listed below.

- From time-series EEG data, it is possible to reconstruct the neuronal "wiring diagram" of a specific species of bird. The reconstructed structure was remarkably similar to the previously documented functional connections of this region of the bird's brain. (Smith et al., 2006)
- Another example is the ability to foresee traffic congestion on the freeway. A deployed system named JamBayes is described for predicting traffic flow in the Seattle area. Predictions are made using a graphical model whose structure was learned from data, as described in the paper (Horvitz et al., 2005).

We occasionally encounter missing data, which is comprised of variables whose values are uncertain. It is possible that we ran a poll in which a number of participants did not respond to some questions or we may have a variety of sensors, some of which fail. In this case, the associated design matrix will have empty cells. These missing entries are frequently represented by the symbol NaN, which stands for "not a number." The goal is to fill these gaps with reasonable values, a task which is referred to as matrix completion. Some applications of an imputation-like task include image inpainting for reconstructing missing regions in an image, collaborative filtering for recommendations, market basket analysis to uncover associations between items etc.

So far we have seen the two main categories of machine learning tasks, as well as subcategories and examples of applications. Table 1.1.1 summarizes the aforementioned types of machine learning. Next, some fundamental machine learning concepts are presented.

Finding a model and its associated parameters with the purpose of having the resulting predictor perform well on previously unseen data is the ultimate goal of learning. When considering machine learning algorithms, there are three basic algorithmic phases that can be distinguished conceptually:

- Prediction or inference

Space	Supervised Learning	Unsupervised Learning
Discrete	Classification or categorization	Clustering
Continuous	Regression	Dimension reduction

Table 1.1.1: Summary of machine learning tasks.

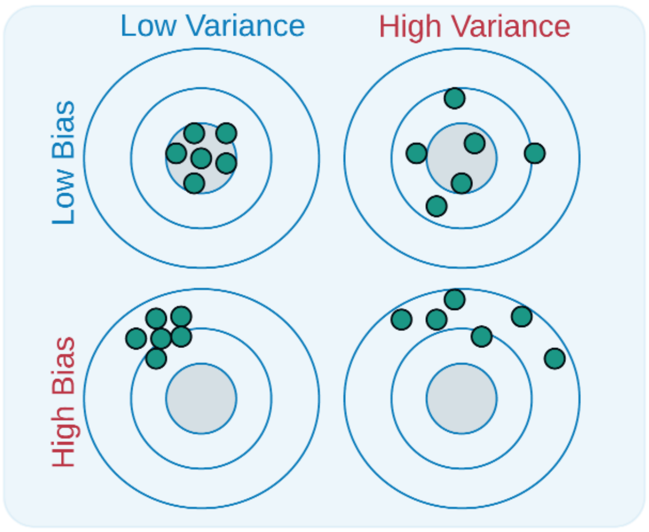


Figure 1.1.1: Visualizing bias and variance trade off using a bulls-eye diagram (Gudivada et al., 2017).

- Training or parameter estimation
- Hyperparameter tuning or model selection

During the prediction step, we apply a trained predictor to previously unseen test data. In other words, the predictor is applied to new vectors representing new input data points, while the parameters and model selection are already fixed. The prediction phase is also referred to as inference. We build our predictive model depending on training data during the training or parameter estimation phase. Given training data, we want to discover good predictors, and there are two primary approaches for doing so: finding the best predictor based on some measure of quality or utilizing Bayesian inference. Finding a point estimate is applicable to both types of predictors, probabilistic and non-probabilistic, but Bayesian inference necessitates the use of probabilistic models. We apply the notion of empirical risk minimization to the non-probabilistic model. Empirical risk minimization presents a direct optimization problem for determining good parameters. The maximum likelihood concept is used with statistical models to discover a good set of parameters. A probabilistic model can also be used to model parameter uncertainty. To discover good parameters that "fit" the data, we employ numerical methods. Most training methods may be viewed as hill-climbing strategies to finding the maximum of an objective, such as the maximum of a likelihood. We use gradients and numerical optimization approaches to apply hill-climbing approaches.

In order to learn a model based on data and make sure that it performs well on unseen data as well, fitting a training set might not be adequate. The method of cross-validation is utilized in order to validate the behaviour of the model on unseen data. Another approach of avoiding overfitting on the training data, is adding priors to our model or using regularization techniques. The dilemma that is often encountered when developing a machine learning system is the bias-variance trade off. Bias error refers to the difference between the prediction and ground truth. Variance error regards the variability of prediction when data changes. Figure 1.1.1 illustrates the bias-variance trade off using a bulls-eye diagram. Given a supervised learning model $y = f(x) + \sigma$ and considering a regression task with mean squared error (MSE), the bias-variance trade off is formally defined below:

$$
\begin{aligned}
\mathrm{Err(x)} &= \mathrm{E}\left[(Y - \hat{f}(x))^2\right] \\
&= \mathrm{E}\left[(Y - E(\hat{f}(x)) + E(\hat{f}(x)) - \hat{f}(x))^2\right] \\
&= \mathrm{E}\left[(Y - E(\hat{f}(x)))^2 + 2(Y - E(\hat{f}(x)))(E(\hat{f}(x)) - \hat{f}(x)) + (E(\hat{f}(x)) - \hat{f}(x))^2\right] \\
&= \mathrm{E}\left[(Y - E(\hat{f}(x)))^2\right] + E[2(Y - E(\hat{f}(x)))(E(\hat{f}(x)) - \hat{f}(x))] + E\left[(E(\hat{f}(x)) - \hat{f}(x))^2\right] \\
& Y - E(\hat{f}(x)) \text{ is a constant} \\
&= \mathrm{E}\left[(Y - E(\hat{f}(x)))^2\right] + 2(Y - E(\hat{f}(x)))(E(\hat{f}(x)) - E(\hat{f}(x))) + E\left[(E(\hat{f}(x)) - \hat{f}(x))^2\right] \\
&= \mathrm{E}\left[(Y - E(\hat{f}(x)))^2\right] + E\left[(E(\hat{f}(x)) - \hat{f}(x))^2\right]
\end{aligned}
\tag{1.2}
$$

The last line of equation 1.2 has two terms. The first term represents the variance and the second one the square of bias. In practice this trade off is interpreted as follows. A simple model with very few parameters might have high bias and low variance, a phenomenon known as underfitting. In contrast, a complex model with large number of parameters might have high variance and low bias, known as overfitting. A good model is the one that balances bias and variance and minimizes the total loss. According to Occam's razor, the general rule in machine learning is to favor simple models with fewer coefficients over complicated models such as ensembles.

We frequently need to make high-level modeling decisions concerning the predictor's structure, such as the number of components to utilize or the class of probability distributions to take into account. The number of parameters is an example of a hyperparameter, and it can have a considerable impact on the model's performance. Model selection refers to the problem of selecting a model from among several options. Model selection for non-probabilistic models is frequently carried out through nested cross-validation techniques.

1.1.2 Time Series Representations

With the rise of the internet of things (IoT), time series data are generated at an exponential rate. Time series consist an integral part of any modern application domain such as astronomy, medicicine, bioengineering, finance, manufacturing and many others. As a result, there has been a huge increase in interest in querying and mining such data, which requires tremendous amounts of computational power. That lead researchers in developing new approaches for indexing, classification, clustering, and time series approximation.

Representation methods and similarity measurements are two critical components for obtaining effectiveness and efficiency while dealing with time series data. Time series are essentially high-dimensional data (Han et al., 2011), and dealing with such data in its raw state is prohibitively expensive in terms of processing and storage costs. Hence, it is highly desired to create representation strategies that can minimize the dimensionality of time series while keeping the core characteristics of a certain data collection. Furthermore, unlike canonical data types, such as nominal or ordinal variables, where the distance definition is simple, the distance between time series must be properly defined in order to reflect the underlying similarity of such data. This is especially useful for similarity-based retrieval, classification, clustering, and other time series mining operations (Ding et al., 2008).

Several high-level time series representations have been proposed for dimensionality reduction, data mining and similarity measurements. Many symbolic representations of time series are also inspired by the

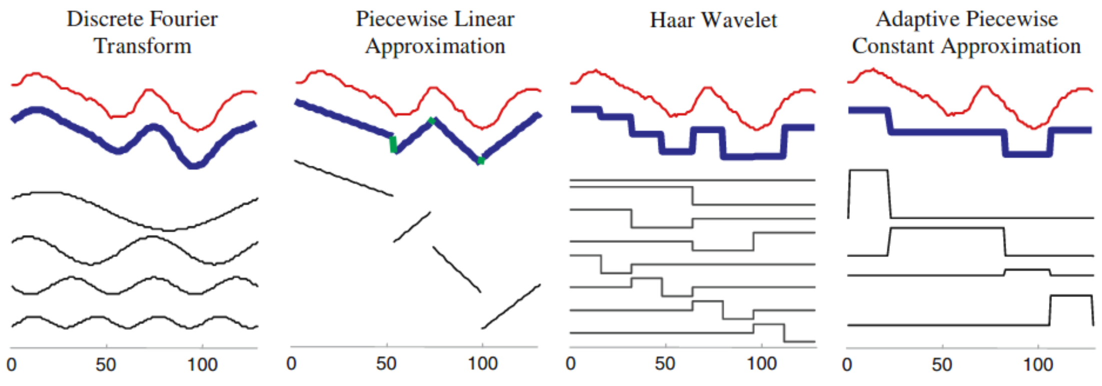


Figure 1.1.2: Visualization of some popular time series representations where a signal is approximated with a linear combination of basis functions (Lin et al., 2007).

domains of natural language processing and bioinformatics, with the hope to take advantage of the variety of data structures and algorithms in these areas. Some popular time series representations include: Discrete Fourier Transformation (DFT) (Faloutsos et al., 1994), Single Value Decomposition (SVD) (Faloutsos et al., 1994), Discrete Cosine Transformation (DCT) (Korn et al., 1997), Discrete Wavelet Transformation (DWT) (Chan and Fu, 1999), Piecewise Aggregate Approximation(PAA) (Keogh et al., 2001a), Adaptive Piecewise Constant Approximation (APCA) (Keogh et al., 2001b), Chebyshev polynomials (CHEB) (Cai and Ng, 2004), Symbolic Aggregate approXimation (SAX) (Lin et al., 2007), Indexable Piecewise Linear Approximation (IPLA) (Chen et al., 2007a) etc. Figure 1.1.2 illustrates for reference the way that four popular time series representations approximate a signal using a linear combination of bases functions (Lin et al., 2007). Furthermore, in the literature there are many distance measures for time series data similarity including Euclidean distance (ED) (Faloutsos et al., 1994), DynamicTime Warping(DTW) (Berndt and Clifford, 1994; Keogh and Ratanamahatana, 2005), distance based on Longest Common Subsequence (LCSS) (Vlachos et al., 2002), Edit Distance with Real Penalty (ERP) (Chen and Ng, 2004), Edit Distance on Real sequence (EDR) (Chen et al., 2005), DISSIM (Frentzos et al., 2007), Sequence Weighted Alignment model (Swale) (Morse and Patel, 2007), Spatial Assembling Distance (SpADe) (Chen et al., 2007b) and similarity search based on Threshold Queries (TQuEST) (Aßfalg et al., 2006).

The majority of the previous work on time series is based on the assumption that time is discrete. A formal description of time series data is given below:

$$T = [(p_1, t_1), (p_2, t_2), ..., (p_i, t_i), ..., (p_n, t_n)] \tag{1.3}$$

with $(t_1 < t_2 < ... < t_i < ... < t_n)$, where each p_i is a d-dimensional data point and each t_i is the respective time. The raw representation of a time series is given by the sequence of the data points without the timestamps. However, the representation of two time series might not be equivalent if the sampling rates are different. Even worse, real data are often noisy and some values are missing adding extra complexity in time series processing. The length of a time series is the number of each elements, and a part of it between two elements is called a segment.

A taxonomy of representation methods for time series is depicted in figure 1.1.3. There are two main categories, data adaptive and non-data adaptive. The former one regards a single representation for all time series in a database with the goal to minimize the global reconstruction error. The latter one, concerns methods that take into account local features and generate an approximate representation. One very important property of time series representations is the ability to estimate a lower bound. This means that we can define a distance measurement on representations with reduced dimensions, that is guaranteed to

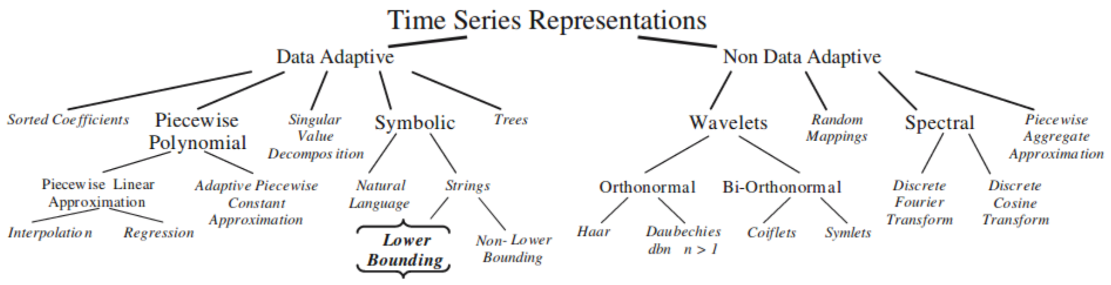


Figure 1.1.3: A classification of all time series representations found in the literature. The leaf nodes refer to the actual representation, whereas the inside nodes refer to the approach classification (Lin et al., 2007).

be less than or equal to the true distance measured on raw data. This lower bounding characteristic enables the use of representations to index data while ensuring that no false negatives occur. Such representations include Interpolation, Adaptive Piecewise Constant Approximation, Singular Value Decomposition, Clipped Data, SAX, Wavelets, DFT, DCT, Chebyshev Polynomials and Piecewise Aggregate Approximation.

A taxonomy of distance measures for time series is listed below.

- Lock-step Measure
  - L_p-norms
    - L_1-norm (Manhattan Distance)
    - L_2-norm (Euclidean Distance)
    - L_{int}-norm
  - DISSIM
- Elastic Measure
  - Dynamic Time Warping (DTW)
  - Edit distance based measure
    - Longest Common SubSequence (LCSS)
    - Edit Sequence on Real Sequence (EDR)
    - Swale
    - Edit Distance with Real Penalty (ERP)
  - DISSIM
- Threshold-based Measure
  - Threshold query based similarity search (TQuEST)
- Pattern-based Measure
  - Spatial Assembling Distance (SpADe)

A similarity function Dist, given two time series T1 and T2, determines the distance between the two time series, represented by Dist (T1, T2). Distance measures that compare the i^{th} point of one time series to the i^{th} point of another are known as lock-step measures e.g. Euclidean distance and the other Lp norms. Distance measures that allow comparison of one-to-many points (e.g., DTW) and one-to-many/one-to-none points e.g. LCSS are referred as elastic measures. Two other categories are threshold-based and pattern-based measures (Ding et al., 2008).

1.1.3 Data Mining for Time Series

With respect to time series data, the most common processing tasks or operations are query by content, clustering, classification, prediction, anomaly detection, motif recognition, and rule discovery, among others (Esling and Agon, 2012). The existence of these issues has been extensively documented in the pattern recognition and data mining fields for many years. For a variety of reasons, however, the well-established pattern recognition and data mining approaches are ineffective for processing time series data. First and foremost, the dimensionality of time series is extremely high, perhaps reaching tens of thousands of dimensions. Second, the corresponding elements of two time series may not be aligned due to differences in length, scale, translation, shift, or non-uniform spacing between adjacent elements between the two time series' elements. Finally, the notion of similarity in the context of time series differs significantly from the notion of similarity employed in pattern recognition. In contrast to pattern recognition, where all elements of pattern vectors are utilized to determine similarity between two patterns, just subsets of elements of two time series may be used to establish their similarity when comparing two time series. Time series can be compared if they contain identical subsequences of appropriate length or numerous identical patterns in the same time order. In this case, the time series are deemed similar (Bettaiah and Ranganath, 2014).

Due to the aforementioned issues, an ideal time series representation that achieves dimensionality reduction should also enable users to conduct any of the actions listed above directly from their representations. Next, the most common time series tasks are discussed, providing representative examples of applications.

The goal of query by content is to retrieve time series from the database that are comparable in information content to the specified query time series Q, in order to complete the task. Among the different types of content-based searches (Esling and Agon, 2012), the range query and the nearest neighbor query stand out. All time series in the database that are within a given distance of the query time series are retrieved using the range query method. In a nearest neighbor query, the query time series's k nearest neighbors (based on a distance metric) are fetched and shown. There are numerous representations that may be used to construct an index structure that can be used to find sequences in the database that are similar to Q or sequences that contain subsequences that are similar to Q. However, there is no time series representation available to aid in the identification of sequences that share identical subsequences of considerable length with Q, which would make this task easier (Bettaiah and Ranganath, 2014). A very popular time series task is clustering. Let N time series with no labels denoted as $T_1, T_2, ..., T_N$ (Bettaiah and Ranganath, 2014). When a set is partitioned into K groups using a meaningful similarity measure, the process is known as clustering. The members of a group are similar to one another, and the members of other groups are considerably different from one another. Most techniques for clustering time series data are versions of algorithms that are used to cluster low-dimension vector data in the first place. A small number of features are computed to represent each time series in the feature-based approaches, and then clustering algorithms, such as k-means algorithms, are used to cluster the feature vectors. The model-based methods extract a set of parameters for each time series and then find clusters based on the parameters used in the clustering process. It is rare to see raw data-based approaches employed in time series analysis because of the large dimensionality of the data.

Classification is the process of allocating an input time series to one of the various recognized classes or categories based on its characteristics (Esling and Agon, 2012). All classification algorithms learn classification rules from training samples whose membership is known in advance of the training. The use of Bayes decision theory for time series classification is not practical because the computation of probability density functions for time series with a high degree of dimensionality is impractical. The number of samples necessary for linear classifiers such as the perceptron, least mean square approaches, and support vector machines is at least twice the dimensionality of the time series. The training of classifiers with patterns of such high dimensionality is computationally demanding, even when the needed training data are available. It becomes too expensive to use non-linear classifiers such as artificial neural networks because of the enormous network size and the large number of training samples that must be used. As a result, classification based on characteristics appears to be the only feasible method (Bettaiah and Ranganath, 2014).

Assume a time series with N values denoted as $T = x_1, x_2, ..., x_N$, then prediction is the task of estimating future values of T for $i > n$. The future values are anticipated based on the present evolution trend observed in the time series, or mathematical models such as the Hidden Markov Model constructed from historical time series data that is similar to the current time series are used to predict future values. Typically, the model is built around notable characteristics of time series (Esling and Agon, 2012).

Motif is a significant pattern in a time series or a group of time series if there is an approximately recurring subsequence that represents the pattern. As an example, motifs are employed in the analysis of stock market data to detect prevalent trading patterns. Motifs have been widely employed in the development of rules, grouping and categorization of time series data, among other applications. While it is difficult to distinguish between motifs of varying time-frames even when they share the same or very similar overall qualities, it is possible (Esling and Agon, 2012).

Rule discovery is the process of discovering temporal rules that are hidden or not visible in time series data. One technique is to convert a time series into a sequence of symbols and then use association rule mining algorithms to uncover the rules that govern the series. For example, well-established charting heuristics such as the cup-and-handle and bull-flag formations may be used to uncover trading principles. Another strategy is to convert a time series into a sequence of events and then use classification trees to learn the rules that govern the sequence of events (Esling and Agon, 2012).

1.2 Thesis Contributions and Structure

This section summarizes the thesis's pivotal research directions, briefly outlining the key ideas of our contributions. Each subsection that follows corresponds to one of these directions, describing the challenges and/or incentives that led to this direction, along with a concise summary of our methodology. Each of these research directions is connected to the respective chapter and the corresponding research article of our study.

1.2.1 AI Tasks Based on Time Series Problems

This research is driven by real-world applications that deal with fundamental time series problems. Chapter 3 introduces concrete examples of three categories of time series problems: non-intrusive load monitoring (NILM), two speech classification tasks and AI tasks for electronic health records. Both theoretical and technical aspects of these problems are discussed. A literature review of past and current methodologies is summarized, with emphasis on the role of machine learning as a tool.

NILM is an industrial application of the well known blind source separation problem with significant impact on a broader threat to humanity, climate change. The key contributions of our research is a systematic survey of the evolution of existing NILM methodologies, addressing practical problems when deploying NILM systems in the real world and some original metrics that will aid overcoming these problems (Nalmpantis and Vrakas, 2019a). This survey has been insightful, because it demonstrates how to compare very complex systems not only qualitatively but also in a quantitative way and has been inspired future NILM studies.

The Internet of Audio Things (IoAuT) is an emerging sub-field of the Internet of Things (IoT) and has attracted many researchers from different disciplines. It lies in the intersection of Internet of Things, sound recognition, machine learning and human computer interaction. IoAuT will be generating a huge amount of audio data which will be hard to clean and annotate. Learning from noisy data is a common problem for many machine learning tasks. Even when a model is trained with noisy data, there is no guarantee it will be robust when deployed in the real-world. Two popular tasks with significant impact on the domain of IoAuT are: speakder recognition and speech commands identification. These two AI tasks and the problems of IoAuT applications motivated our research to develop robust machine learning models that can meet the requirements of a system in production.

A third domain where sequential data are abundant and AI is going to play a significant role in the near future is healthcare and medicine. Indeed, computing has been an integral part of medical applications for several years, assisting physicians with hardware and software applications. Our research is driven by two basic types of health records: electrocardiographs (ECGs) and bone age assessment. The former one is a classic type of signal in medicine and is used for the classification of heard disorders or the prediction of critical heath events. The latter one is mainly an image based task, but it can be reformulated as a time series problem by extracting data where the sequence matters e.g. the outlines of bones.

1.2.2 Machine Learning Methods for NILM

Chapter 4 presents our main contributions towards solving the problem of NILM. Firstly a novel benchmark framework of NILM systems is presented by analyzing the importance of a standardized evaluation process. The framework was firstly introduced in our study (Symeonidis et al., 2019) and is used for evaluating next generation NILM systems. At the same work stacking machine learning methods are examined for solving the problem of power disaggregation. Then there is a series of original neural network architectures that tackle this problem as a regression task and predicting the energy consumption of individual appliances. The first neural architecture, which has been one of the state-of-the-art models, is called online-GRU (Krystalakos et al., 2018). This study introduced the rolling window approach in contrast to previous methods that used sequence to sequence or sequence to point approaches. Furthermore, it performs on par or better when compared to other state-of-the-art models and it has less than half the parameters of a previous recurrent neural network. Then our research was focused on the knowledge transfer from other domains such as image recognition and it was the first time that imaging time series techniques were introduced in NILM research (Kyrkou et al., 2019). Our next steps stem from one basic practical problem when deploying real-time NILM systems on embedding devices. Indeed, such devices have limited computational and storage resources and modern deep learning models tend to be massive in size and require large computational power. Self-attentive energy disaggregator (SAED) (Gkalinikis et al., 2020; Virtsionis-Gkalinikis et al., 2021) is a novel neural network that has very small size and is very fast in inference. Our latest contribution, under this section, is a state-of-the-art neural network, named NFED, which performs on par with other state-of-the-art energy disaggregation models but has much smaller size and demonstrates very high inference speed (Nalmpantis et al., 2022a).

1.2.3 A Novel Time Series Representation

Representation learning has been an active subject of research and has played a central role in the success of deep learning. Time series representations is a sub-domain which lies in the intersection of time series and knowledge representation. Chapter 5 goes over classic time series representations analysing their benefits and drawbacks. It presents our research directions and motivations which are driven from the problems that are analyzed in previous sections.

The contribution of our research regarding time series representations is twofold. Firstly, a novel time series embedding representation is developed, known as Signal2Vec. Signal2Vec was firstly presented by Nalmpantis and Vrakas (2019b). Another similar algorithm but specialized only on energy data, was introduced previously in (Nalmpantis et al., 2018). In a nutshell, Signal2Vec converts any signal into vectors by using an embedding space that is learned in an unsupervised approach. The idea, as many other time series representations, is stimulated by the domain of natural language processing and more specifically from the model Word2Vec. Secondly, Signal2Vec is applied on concrete time series problems in the domain of energy. The first task regards time series classification problem, where segments of energy signals are classified according to the device that produces them. The second task is the problem of energy disaggregation seen as a multi-label classification problem. The latter task involves an original framework called multi-NILM which is introduced by Nalmpantis and Vrakas (2020). Multi-NILM which has the benefit to utilize only one multilabel classifier to identify multiple devices, given only the main power consumption of a house. The input to the classifier is a vector representation of the raw energy data, which is achieved by using the Signal2Vec algorithm. The experimental results verify the success of Signal2Vec and a comparative analysis against other representations shows that our proposed method achieves state-of-the-art results in the task of multi-label NILM.

1.2.4 Neural Architectures Based on Information Theory

The third major contribution of this thesis concerns both theoretical and practical aspects of deep learning when applied on time series problems. The motivation of this direction stems from the application of information theory in machine learning. Information theoretic learning is a very promising domain that could reveal the underlying mechanics of learning and shed light on unexplained phenomena of overparameterized learning models.

Chapter 6 focuses on the role of pooling operation in deep neural networks. As it is known modern deep learning models are discovered with the method of trial and error without any solid mathematical theory. Selecting the right pooling operation for a deep neural network is not an exemption. Despite the plethora of choices when it comes to a pooling function, e.g average, max etc, there is no rigorous way to find out which is one works best depending on the given data and model architecture. Our first research contribution is a theoretical analysis of feature pooling in deep learning via the lens of information theory (Nalmpantis et al., 2019). Indeed, pooling can be seen as the problem of maximum entropy sampling, which has been well studied by mathematicians and computer scientists in the past. In addition to the theoretical analysis, our work (Nalmpantis et al., 2019) also presents a novel pooling operation named entropy pooling. Entropy pooling, instead of the classic approaches, it doesn't depend on the data and tries to maximize the entropy of its output, by propagating a diversified group of features. Since the model doesn't depend on the data distributions and is robust to nuances, it has been incorporated in lightweight neural architectures for the problems of speech commands classification (Nalmpantis et al., 2021) and speaker recognition (Nalmpantis et al., 2022b). Extensive experiments show the noise invariance of entropy pooling when background sounds such as claps, traffic, white noise etc are injected into the original data.

Continuing our research efforts to bridge the gap between theory and practice of deep neural networks, we developed a variational approach of feature pooling based on the principle of Information Bottleneck. The novel pooling operation is called VIB-Pooling and is introduced by Nalmpantis and Vrakas (2022). Information bottleneck has been introduced in deep learning by Tishby and Zaslavsky (2015) in order to explain the learning dynamics of deep neural networks. Since then, there is a rich literature work utilizing this theory and developing variations of the original concept. Our work is inspired by Deep Variational Information Bottleneck (Alemi et al., 2016). VIB-Pooling has the properties of producing disentangled feature representations which are robust to noise. It is integrated on popular neural networks and the conducted experiments demonstrate that VIB-Pooling boosts the performance of various models when it replaces classic pooling layers. The tasks that VIB-Pooling is evaluated include medical time series as well as popular image classification tasks.

2 Background Knowledge

"If I have seen further it is by standing on the shoulders of Giants."
— Isaac Newton

This chapter is an overview of the prerequisite knowledge that most of this work is based upon. It starts with classic approaches in time series representations. Most of these approaches stem from signal processing or natural language processing and played a significant role in solving time series problems such as pattern matching, indexing, classification etc. Next, machine learning based approaches of representations are introduced, with a focus on learning distributed representations and how this lead to the quest of optimal representations. One of the desired properties of optimal representations is disentanglement of information and the latest advancements in this domain are summarized. Representation learning has been revolutionized by deep learning. A short description of the most important deep learning architectures relevant to this thesis is presented, including convolutional and recurrent neural nets. Finally, this chapter closes with an introduction of information theoretic principles in deep learning methods. Information theory has been an integral part of machine learning since its birth and is getting more and more attention nowadays. Current learning theory has been tested by deep learning and proved to be inadequate to explain various phenomena that are present in the regime of overparameterized models. Information theory is a promising direction that could possibly shed light on these obscured phenomena and bridge the gap between theory and practice. It also offers a mathematical framework that is indispensable towards the goal of optimal representations.

2.1 Classic Time Series Representations

Seven time series representations, that have been used in this research, are described in this section. A brief description of these algorithms is presented below, outlining their advantages and disadvantages.

2.1.1 Piecewise Aggregate Approximation (PAA)

One of the most popular time series approximations is Piecewise Aggregate Approximation (PAA) (Keogh et al., 2001a). The implementation is very straightforward. Firstly the series is divided into M time-frames of equal size. Then, the mean value of each time-frame is calculated, forming a representation of M dimensions. The algorithm of PAA is expressed as follows. Denote a time series $X = x_1, ..., x_n$, with length n. PAA is an approximation that represents time series X in M space by a vector $\bar{X} = (\bar{x}_1, ..., \bar{x}_M)$, where $M \leq n$. The ith element of $\bar{X}$ is calculated by the following equation:

$$\bar{x}_i = \frac{M}{n} \sum_{j=n/M(i-1)+1}^{(n/M)i} x_j \tag{2.1}$$

It is proved that the complexity of this transformation can be reduced from O(nM) to O(Mm) where m is the number of frames. Figure 2.1.1 shows a diagram of a random time series including its PAA representation.

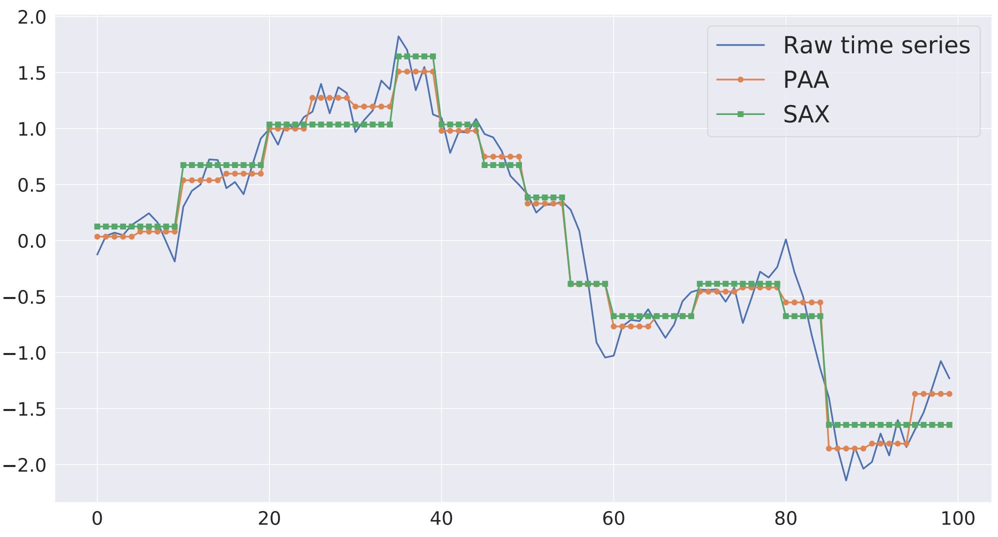


Figure 2.1.1: Illustration of a random time series and its PAA and SAX representations.

From this diagram, it can be concluded that PAA transforms a given time series to a sequence of steps. As a result this technique is proved to be quite robust when dealing with noisy data.

2.1.2 Symbolic Aggregate Approximation (SAX)

SAX is an extension of PAA, inheriting its simplicity and the low computational complexity (Lin et al., 2007). The main advantage of this algorithm is that it transforms a time series into a string representation. SAX has been very successful in many tasks related to time series such as classification, clustering, summarization, anomaly detection and indexing. The steps of the algorithms are: a) Apply PAA algorithm to segment the time series. b) Discretize the average values by mapping them to letters of an alphabet. The only requirement of the latter step is that the new symbols have to be produced with equiprobability. This is achieved with normalized time series because they have a Gaussian distribution. The difference between SAX and PAA is shown in Figure 2.1.1. More formally this is expressed by defining breakpoints as a list of ordered numbers $B = \beta_1, \beta_2, ..., \beta_{a-1}$ such that the area under the Gaussian curve between two adjacent breakpoints is constant. Consequently, a time series can be converted into words by assigning a symbol $alpha_j$ for each interval $[\beta_{j-1}, \beta_j)$. The mapping of PAA approximation $\bar{C}$ into a string $\hat{C}$ is given by the formula below:

$$\hat{c} * i = alpha * j,\ iif\ \bar{c} * i \in [\beta_{j-1}, \beta_j) \tag{2.2}$$

2.1.3 1d-SAX

An alternative version of SAX is 1d-SAX (Malinowski et al., 2013). The benefit of this version is that it takes into account the trend of each subsequence when mapping the values of the time series into symbols. Thus after applying PAA, instead of computing the average of a segment, we compute the linear regression. The two coefficients of linear regression for each segment are mapped into symbols independently and then combined together into one symbol. The statistical properties of the average values and the slope values form a Gaussian distribution and consequently the quantization step is achievable in the same way as in the original version of SAX.

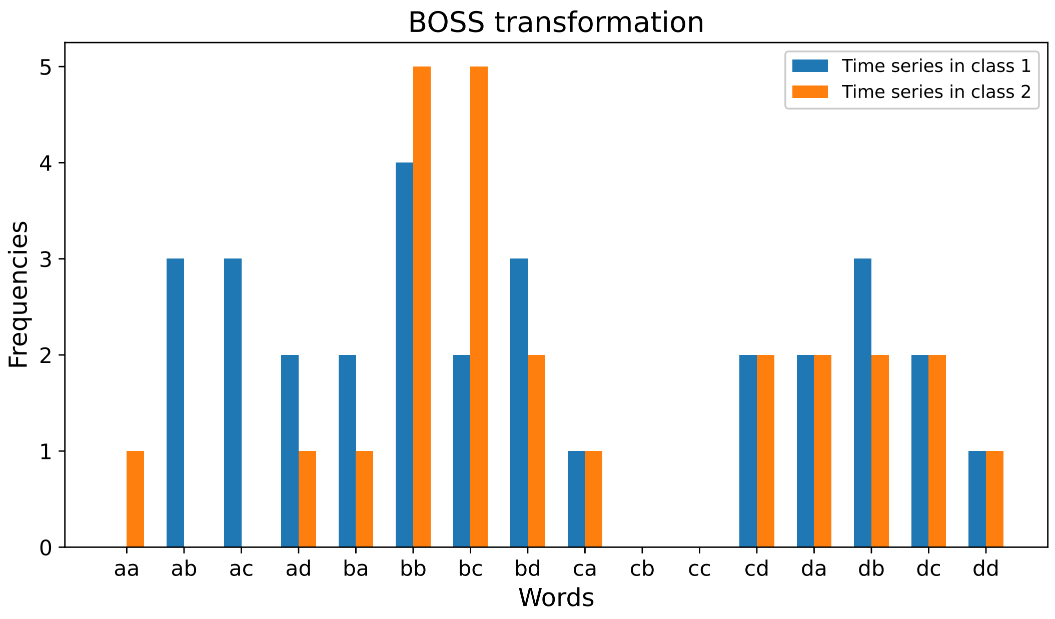


Figure 2.1.2: Illustration of the words and their frequencies that have been learned by BOSS

2.1.4 Discrete Fourier Transform (DFT)

One of the most common time series approximation algorithms is the Discrete Fourier Transform (DFT), which was firstly used for the problem of time series similarity search by Agrawal et al. (1993). The most important properties of DFT are that its coefficients consist an orthogonal basis, the first few coefficients contain most of the information, it is fast to compute due to Fast Fourier Transform (FFT) algorithm and it preserves the Euclidean distance due to Parseval's theorem.

2.1.5 Symbolic Fourier Approximation (SFA)

Schäfer and Högqvist (2012), inspired by DFT, suggested a novel time series representation named Symbolic Fourier Approximation. SFA consists of two steps, preprocessing and transformation. Preprocessing includes the DFT approximation and a quantization technique called multiple coefficient binning (MCB). MCB minimizes information loss during quantization by grouping the DFT coefficients of all subsequences, creating a histogram for each group and applying binning to each group. The transformation step maps the MCB discretization into symbols of a finite alphabet.

2.1.6 Bag-of-SFA-Symbols (BOSS)

An extension and improvement of SFA is a time series representation named Bag-of-SFA-Symbols (BOSS), which is very robust in noise (Schäfer, 2015). This feature is important in time series tasks, as real world data tend to be very noisy or erroneous. In order to obtain BOSS representation of a given time series T, sliding windows $S_{i;w}$ of size w are extracted. Next each window is converted to unordered SFA words. The transformations $SFA(S_{i;w}) \in \Sigma^l$, for $i = 1, 2, ..., (n - w + 1)$ are used to build a histogram. The BOSS histogram is a function, $B : \Sigma^l \to N$, which maps the SFA word space into the space of natural numbers. BOSS has the property of phase shift invariance because it doesn't take into account the order of SFA words and uses numerosity reduction (Lin et al., 2012, 2007) to avoid outweighing segments with constant values. An example of BOSS transformation for a time series with two classes is shown in Figure 2.1.2. It illustrates the words that are extracted from the given time series and the features representing frequencies of each word.

2.1.7 Word ExtrAction for time SEries cLassification (WEASEL)

WEASEL is a novel time series representation which has demonstrated very promising results on the task of time series classification (Schäfer and Leser, 2017). The efficiency of this method is attributed to two new methods named discriminative approximation and discriminative quantization. The former one uses the one-way ANOVA F-test during the approximation step, to select the Fourier coefficients that are characteristic to class labels. The latter method maximizes the information gain, resulting in low entropy feature set of class labels. Given a time series, WEASEL firstly extracts normalized windows of various sizes. Then Fourier coefficients are calculated and filtered, keeping the ones that are characteristic to this particular time series. The filtering process is achieved by using the ANOVA F-test. Next, the Fourier coefficients are quantized. The quantization process is completed using information gain binning to best separate the various time series classes. The unigrams and bigrams are then used to construct a bag-of-patterns, filtering out irrelevant words using the Chi-Squared test. Bag-of-patterns approaches have linear computational complexity, therefore WEASEL achieves the best trade off between accuracy and speed.

2.2 From Distributed to Disentangled Representations

The performance of machine learning algorithms is often dependent on data representation, which is also known as feature representation. It is assumed that alternative representations might entangle and obscure many explanatory aspects of variation behind the data. Although specialized domain knowledge can be utilized to aid in the creation of representations, learning using generic priors can also be used. The study of AI is encouraging the development of increasingly powerful representation-learning algorithms that use such priors. This section summarizes recent improvements in unsupervised feature learning and deep learning, including advances in probabilistic models and deep networks. Long-term unresolved research questions concern objectives for learning the right representations, computing representations (i.e., inference), and the geometrical links between representation learning, density estimation, and manifold learning (Bengio et al., 2013).

2.2.1 Distributed Representations

One of the most essential tools for representation learning is distributed representations of concepts, which are representations made of multiple pieces that can be set apart from one another. Distributed representations are powerful because they can describe k^n different concepts using n features with k values, with $n, k \in Z$. They are encapsulated by the learnt representations of both neural networks with multiple hidden units and probabilistic models with multiple latent variables. Modern deep learning models are driven by the idea that the hidden units can learn to represent the underlying causal elements that explain the data. Distributed representations are natural for this strategy because each direction in representation space might correspond to the value of a separate underlying configuration variable. In a symbolic representation, the input is connected with a single symbol or category. If the dictionary contains n symbols, one might assume n feature detectors, each detecting the related category. In that instance, only n different representation space arrangements are feasible, slicing n different regions in input space.

Some examples algorithms that are based on non-distributed representations include clustering algorithms, k-nearest neighbors, decision trees, Gaussian mixtures, kernel machines, models based on n-grams and others. For clustering algorithms, such as the k-means, each input point is assigned to a specific cluster. In k-nearest neighbors algorithms a given input is associated with one or more instances. When k > 1, each input is represented by many values. However these values are not controlled independently, hence this is not a truly distributed representation. Regarding decision trees, when an input is given to the model only one leaf and the nodes on the path from root to leaf are active. Gaussian mixtures and kernel machines have the same issue with k-nearest neighbors that each input is represented by multiple values, but they cannot be changed independently. As far as n-gram models are concerned such as language or translation models, the sequences are partitioned according to a tree structure. Therefore, only a small percentage of parameter sharing is possible as each leaf corresponds to separate parameters.

An essential related idea that distinguishes a distributed representation from a symbolic representation is that generalization occurs as a result of common properties between different concepts. Let the symbols "cat" and "dog" then the distance between each other is as far as any random pair of symbols. However, if they are associated with a meaningful distributed representation, there are many common properties between cats and dogs. For example, our distributed representation may include items like "has fur" or "number of legs" that have the same value for the embedding of both "cat" and "dog". Neural language models that act on distributed representations of words generalize substantially better than models that work directly on one-hot representations of words. Distributed representations generate a rich similarity space in which semantically near ideas (or inputs) are close in distance.

When an ostensibly complex structure can be compressed using a limited number of parameters, distributed representations can have a statistical advantage. Some conventional non-distributed learning methods generalize because of the smoothness assumption, which asserts that if $u \approx v$, then the target function f to be learned has the property that $f(u) \approx f(v)$ in general. There are numerous ways to formalize such an assumption, but the final result is that if we have an example (x, y) for which we know that $f(x) \approx y$, then we select an estimator f that approximates these restrictions while changing as little as possible when we shift to a neighboring input $x + \epsilon$. This assumption is obviously highly beneficial, but it suffers from the curse of dimensionality. In order to learn a goal function that increases and drops many times in many different regions, we may require at least as many samples as the number of identifiable regions. Consider each of these regions to be a category or symbol, then by giving each symbol (or region) its own degree of freedom, we may learn an arbitrary decoder mapping from symbol to value. However, we are unable to generalize to new symbols for new regions as a result of this. If we're lucky, the desired function will have some regularity in addition to being smooth. A convolutional network with max pooling, for example, can recognize an object irrespectively of its location in the image, even if the object's spatial translation does not correspond to smooth changes in the input space.

Assume a subset of a distributed representation learning method that extracts binary features by setting a threshold for linear functions of the input. Then, each binary feature divides R^d into two half-spaces. The number of areas that this distributed representation learner can differentiate is determined by the exponentially huge number of intersections of n of the respective half-spaces. Using a general result about the intersection of hyperplanes (Zaslavsky, 1975), it is possible to demonstrate (Pascanu et al., 2014) that the number of regions that this binary feature representation can identify is:

$$\sum_{j=0}^{d} \binom{n}{j} = \mathcal{O}(n^d) \tag{2.3}$$

As a result, we see an exponential increase in input size and a polynomial increase in the number of hidden units.

This gives a geometric explanation for distributed representation's generalization power. Given $\mathcal{O}(nd)$ parameters for n linear threshold features in R_d, we can distinctly represent $\mathcal{O}(nd)$ regions in input space. If we instead made no assumptions about the data and utilized a representation with one unique symbol for each region and independent parameters for each symbol to identify its equivalent section of R_d, defining $\mathcal{O}(nd)$ regions would necessitate $\mathcal{O}(nd)$ examples. The distributed representation argument could also be extended to the case where, instead of utilizing linear threshold units, we utilize nonlinear, possibly continuous, feature extractors for each of the attributes in the distributed representation. In this case, the argument is that if a parametric transformation with k parameters can learn about r regions in input space, with $k \ll r$, and if obtaining such a representation was useful to the task of interest, then we could possibly generalize much better in this way than in a non-distributed setting, where we would need $\mathcal{O}(r)$ examples to get the same features and related partitioning of the input space into r regions. When we use fewer parameters to represent the model, we have fewer parameters to fit and consequently considerably fewer training instances to generalize successfully.

Another reason why models based on distributed representations generalize effectively is that their capacity remains restricted despite their ability to encode so many different regions. The VC dimension of a neural network of linear threshold units, for example, is just $\mathcal{O}(w \log w)$, where w is the number of weights (Sontag et al., 1998). This limitation arises because, although we may assign a large number of unique codes to representation space, we cannot use the entire code space, nor can we use a linear classifier to learn arbitrary functions mapping from the representation space h to the output y. The combination of a distributed representation and a linear classifier consist a prior belief that the classes to be recognized are linearly separable as a function of the underlying causal factors recorded by h.

Although the concepts mentioned thus far have been abstract, they could also be experimentally verified. Hidden units of a deep convolutional network trained on the ImageNet and Places datasets learn features that are frequently interpretable, matching to a label that people would naturally assign, according to Zhou et al. (2015). Radford et al. (2015) demonstrated that a generative model can learn a representation of photos of faces, with different representation space orientations capturing different underlying factors of variation. For example one direction in the representation space can correspond to the gender of a person and another one to other characteristics such as wearing glasses. These characteristics were discovered automatically rather than being predetermined. There is no need to label the hidden unit classifiers because gradient descent on an objective function of interest automatically learns semantically interesting features as long as the task requires them. We can learn about the distinction between male and female, or about the presence or absence of glasses, without having to characterize all of the configurations of the rest $n - 1$ features with examples that cover all of these combinations. This type of statistical separability is what enables one to generalize to novel configurations of a person's features that were not seen during training (Goodfellow et al., 2016).

2.2.2 Disentangled Representations

Two important properties of a good representation are abstraction and invariance. Deep learning models have been verified that their first layers learn abstract representations and the following ones represent less abstract concepts. This is a natural order as more abstract ones can be used to construct more specific ones. One such mechanism that leads to abstraction of the information is the pooling mechanism. In general more abstract concepts are invariant to local perturbations and noise. As a result, the representations that capture these notions are typically extremely non-linear functions of the raw input. This is obviously true with category concepts, where more abstract representations detect categories that encompass a wider range of phenomena, potentially giving them higher prediction power. Abstraction can also exist in high-level continuous-valued features that are sensitive to certain types of input changes. Learning such invariant features has been an open research question for several years.

Apart from the two aforementioned properties of features, distributed and invariant, another very important one is disentanglement. The goal of disentanglement is to clearly separate factors of variations (Bengio et al., 2013). Various causal elements of the data tend to change independently of each other in the input distribution. It is assumed that only a few at a time change given a sequence of real-world inputs. Complex data emerges from the rich interaction of multiple factors. These components interact in a complicated system, which can challenge AI tasks like image classification. For instance, an image is made up of the interaction of one or more light sources, object geometries, and the physical properties of the surfaces in the image. Shadows from objects in the scene might collide in intricate patterns, generating the illusion of edges and corners where none exist and radically altering the observed object shape. The solution to these issues should rely on the data itself, using massive amounts of unlabeled samples to develop representations that distinguish the many factors of variation. This should result in a representation that is substantially more resistant to the complex and deeply structured fluctuations found in natural data sources for AI challenges.

It is critical to differentiate the concepts of invariant and disentangled features. The former one refers to lower sensitivity to irrelevant information, whereas the latter one to the separation of causal factors relevant to a task. Unfortunately, it is hard to predict which set of features and variants will eventually be important given a specific task. Furthermore, the feature set being trained may be intended to be employed in numerous tasks with diverse subsets of relevant features. Considerations like these lead to the conclusion that the most effective strategy to feature learning is to disentangle as many factors as possible while rejecting as little information as possible.

In representation learning finding the right objective to train a model is not as clear as in task specific models such as classification ones. To set an example, in classification the goal is to reduce the number of wrong classifications given a training dataset. In the case of representation learning, the goal is not necessarily linked to the tasks that the learnt features will be used on. This challenge is analogous to the reward function issue faced in reinforcement learning. It is agreed that a good representation is one that separates the underlying sources of variation but it is not obvious at all how to incorporate this goal into formal training criteria. This is an open research question that will occupy researchers for several years. Next, the most recent advancements in learning disentangled representations are summarized.

As it has been mentioned, the most broadly accepted definition of disentanglement is to separated independent factors of variation, which usually align with the generative factors of the data. The idea dates back to the work of (Schmidhuber, 1992), whereas the term was firstly introduced by (Bengio, 2009). Recently Higgins et al. (2018) formulated a group theoretic definition arguing that the structure of the world that disentangled representations should capture are the symmetry transformations of the world state.

Disentangled representations are considered meaningful in the sense that they are robust, transferable, interpretable, explainable and fair. In a more formal definition disentanglement leads to optimal representations. According to Achille and Soatto (2018a), given the Markov chain $x \to z \to y$ an optimal representation z should be sufficient, minimal and invariant. Sufficient for the task y, for example $I(y;z) = I(y;x)$, means that information about y is not lost. Minimal means that $I(z;x)$ is minimized so that it retains as little about x as possible. Invariant to the effect of nuisances $I(z;n) = 0$ is required to avoid learning spurious correlations between nuisances n and labels y. An additionally property of optimal representations is to be maximally disentangled. If a representations can be maximally disentangled then it is not a unique solution. This property is required in order to avoid higher-order correlations between the components of z. Then the total correlation $TC(z)$ would be equal to zero.

$$TC(z) = KL(p(z)||\prod_i p(z_i)) \tag{2.4}$$

Disentanglement methods assume that there is an underlying set of independent ground truth variables that govern the generative process of observable data (Träuble et al., 2021). This process is formally described as follows:

$$\mathbf{x} \sim \int_c p^*(\mathbf{x}|\mathbf{c})p^*(\mathbf{c})dc \tag{2.5}$$

where the prior over ground truth factors **c**:

$$p^*(c_1, c_2, ..., c_n) = \prod_{i=1}^{n} p^*(c_i) \tag{2.6}$$

The goal of disentanglement is to learn independent factors of variations that are equivalent with the ground truth variables:

$$p(\mathbf{z}) = \prod_{i=1}^{n} p(z_i) \tag{2.7}$$

In the context of deep learning the natural choice of model to study optimal representations is the architecture of variational autoencoders (VAE). The basic model of variational autoencoders is described as follows. Firstly, we assume a prior $P(\mathbf{z})$ on a latent feature space. Then, we parameterize the conditional probability distribution $P(\mathbf{z}|\mathbf{x})$ with a deep neural network. Next, $P(\mathbf{z}|\mathbf{x})$ is approximated using a variational distribution $Q(\mathbf{z}|\mathbf{x})$, parameterized with a deep neural network. Finally the evidence lower bound (ELBO) is utilized as the objective function which is described by the following formula.

$$\max_{\phi,\theta} \mathbb{E}_{p(x)}[\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))] \tag{2.8}$$

Figure 2.2.1 illustrates the details of the general framework of VAEs. The main benefit of VAEs is that they offer a way to discover the latent factors of variations in an unsupervised way. In the literature there are many variations of the VAE framework. Three such versions of VAEs are described by the following equations.

- Higgins et al. (2017) proposed $\beta-$VAE:

$$\max_{\phi,\theta} \mathbb{E}_{p(x)}[\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta D_{KL}(q_\phi(z|x)||p(z))] \tag{2.9}$$

- Kim and Mnih (2018) proposed $Factor$VAE and Chen et al. (2018) proposed β-TCVAE which have the general equation:

$$\max_{\phi,\theta} \mathbb{E}_{p(x)}[\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))] - \gamma D_{KL}(q(z)||\prod_{j=1}^{d} q(z_j)) \tag{2.10}$$

- Kumar et al. (2018) suggested DIP-VAE:

$$\begin{aligned}\max_{\phi,\theta} &\mathbb{E}_{p(x)}[\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))] \\ &- \lambda_{od}\sum_{i\neq j}[Cov_{q_\phi}[z]]_{ij}^2 - \lambda_d \sum_{i\neq j}([Cov_{q_\phi}[z]]_{ii} - 1)^2\end{aligned} \tag{2.11}$$

Measuring disentanglement is not a trivial task. There are many metrics and researchers have not agreed which ones are the right ones. Quite often these metrics are proposed along with a model and it is not always clear if they are biased by the actual method. Eastwood and Williams (2018) proposed a disentanglement

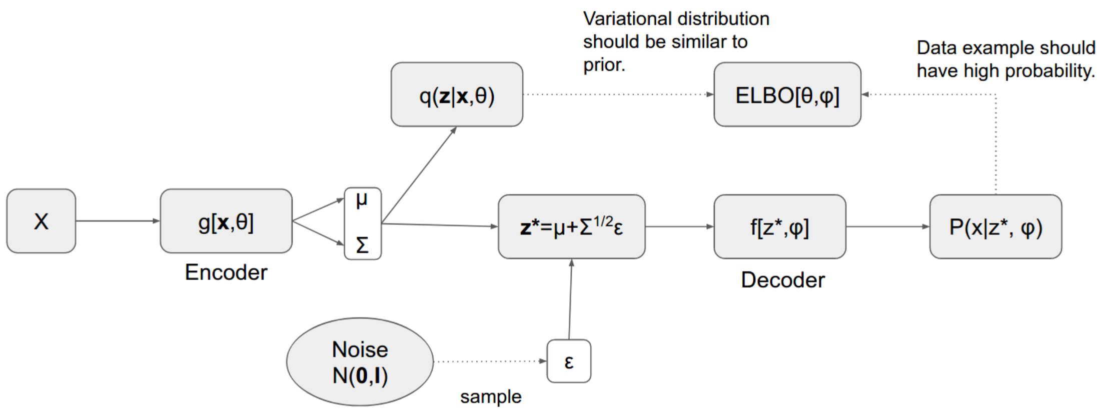

Variational distribution should be similar to prior.
Data example should have high probability.
q(z|x,θ)
ELBO[θ,φ]
X
g[x,θ]
μ
Σ
z*=μ+Σ^{1/2}ε
f[z*,φ]
P(x|z*, φ)
Encoder
Decoder
Noise N(0,I)
ε
sample

Figure 2.2.1: Illustration of a variational autoencoder.

metric called DCI disentanglement. It computes the entropy of the distribution of the weighted learned representations. The importance of each latent variable in predicting factors of variation is computed with Lasso, Random Forest or gradient boosted trees and then it is normalized. Total correlation is a simple and robust metric. High values mean a representation is not factorizing. Total correlation is given by the following equation:

$$TC(z) = D_{KL}(p(z)|| \prod_i p(z_i)) \tag{2.12}$$

Other metrics include BetaVAE metric (Higgins et al., 2017), FactorVAE metric (Kim and Mnih, 2018), Mutual Information Gap (MIG) (Chen et al., 2018), Modularity (Ridgeway and Mozer, 2018), SAP score (Kumar et al., 2018) and others. As it is observed most of the aforementioned metrics are named after the methodology that is proposed to build a disentangled latent feature space.

Learning disentangled representations is an active research area in machine learning and it is getting more and more attention. Yet, there is not method that clearly separates factors of variations especially when the underlying mechanisms are mixed in a nonlinear way. Some important steps towards true disentangled representations are described next.

It is often desirable to construct disentangled representations using unsupervised methods. However it is proved that unsupervised learning of disentangled representations is not possible without inductive biases according to Theorem 1 by Locatello et al. (2019). Following, Locatello et al. (2020) showed that weak supervision facilitates disentanglement for uncorrelated factors of variations. However, real world data are correlated and the previous work applies on simple toy examples. Proposition 1 by Träuble et al. (2021) states that factorization-based inductive biases are not sufficient to learn disentangled representations. The authors address this issue and suggest that post-hoc solutions with very few labels and weakly-supervised solutions are helpful even when the true factors of variations are correlated.

Träuble et al. (2021) propose a post-hoc solution named fast adaptation and the steps are as follows. Firstly, we try to identify a pair of entangled dimensions z_i, z_j. Next, we need a small training set with input latent codes and target the true factors of variations (FoV) $c_j, 1 < j < M$. Then, a gradient boosting tree (GBT) is trained using the labels to get an importance weight for each pair FoV and latent codes. The entangled latent codes are identified by looking at the maximum feature importance. A new model is trained to infer the ground truth FoV from the entangled variables z_i, z_j e.g. $f_\theta(z_i, z_j) = (c_1, c_2)$. The predictions of FoV are used to replace the latent variables.

Locatello et al. (2020) propose a weakly-supervised disentanglement methodology based on the assumption that pairs of observations (x_1, x_2) share a random subset S of latent factors. The methodology does no assumption on which or how many factors are shared. By resampling one or more factors of the latent space z we obtain $\widehat{z}$. Next, pairs of real data with only a few factors different are used for training. Finally, a variant of β-VAE is optimized by enforcing $p(z_i|x_1) \neq p(z_i, x_2)$. The non-shared dimensions need to be efficiently used to encode the non-shared factors of x_1 and x_2. The equation of the objective function of the variant β-VAE is given by the formula below:

$$\begin{aligned} \max_{\phi,\theta} \mathbb{E}_{x_1,x_2} \mathbb{E}_{\widehat{q}_\phi(\widehat{z}|x_1)}[\log p_\theta(x_1|\widehat{z})] + \mathbb{E}_{\widehat{q}_\phi(\widehat{z}|x_2)}[\log p_\theta(x_2|\widehat{z})] \\ - \beta D_{KL}(\widehat{q}_\phi(\widehat{z}|x_1)||p(\widehat{z})) - \beta D_{KL}(\widehat{q}_\phi(\widehat{z}|x_2)||p(\widehat{z})) \end{aligned} \tag{2.13}$$

As it has been discussed above disentanglement is a desired property for good representations. This is a hard task and unsupervised methods are proved that are not suitable, especially for highly correlated data. Current experiments using relatively simple datasets, showcase that disentangled representations are indeed useful, but there is plenty room for improvement. Future research directions should explore the efficiency of post-hoc solutions or weakly-supervised training methods in learning disentangled representations of complex and high-dimensional data. Another research directions is to try to enhance post-hoc or weakly-supervised methods with text annotations and taking advantage of the compositionality of language models. Finally, more experiments should be conducted to investigate the usefulness of disentangled representations in real-world downstream tasks in terms of the sample complexity of learning.

2.3 Deep Learning Architectures

Hinton et al. (2006) introduced a groundbreaking work in feature learning and deep learning, which was quickly adopted by other researchers (Bengio et al., 2007; Ranzato et al., 2007; Lee et al., 2007). Bengio (2009) review and discuss the breakthroughs of deep learning extensively. The core idea is known as *greedy layerwise unsupervised pre-training*. It is a hierarchical way to learn features by layer using unsupervised feature learning. The learned features of each level are used to learn the new features of the next one. By stacking these layers, they can be used to train a supervised model, such as a neural network classifier, or a deep generative model, such as a Deep Boltzmann Machine (Salakhutdinov and Hinton, 2009). This methodology of stacking layers that work as feature extractors has been demonstrated very successful in terms of classification error (Larochelle et al., 2009; Erhan et al., 2010), quality of the generated samples in the case of probabilistic models (Salakhutdinov and Hinton, 2009) or in terms of invariance of the learned features (Goodfellow et al., 2009).

A deep network can represent functions of increasing complexity by adding more layers and more units per layer. This section describes the core concepts and the basic building blocks of modern neural networks.

2.3.1 Feedforward Neural Networks

Feedforward neural networks, also known as multilayer perceptrons (MLPs), are essential part of modern deep learning models. Feedforward neural networks can approximate any function f_*. For example given a classifier $y = f_*(x)$, a feedforward network defines a mapping $y = f(x; \theta)$ and approximates the original function f_* by learning the right parameters θ. The name feedforward derives from the fact that the information flows from the input x to the output via the computations defined by f. If extra feedback connections are used then the network is called recurrent.

The function f typically consists of many other functions and their compositionality is described by an directed acyclic graph. To set an example assume the functions f_1, f_2, f_3, then one possible composition is $f(x) = f_3(f_2(f_1(x)))$. The hierarchical structure of neural networks is characterized by layers and in this case we can say that each function is a layer. The number of the layers is the depth of the model, f_1 is the first layer, f_3 the last one and all the intermediate layers are called hidden. Typically the layers are represented by vectors and the dimensions of the vectors consist the width of each layer.

The functions that are used in deep feedforward networks can be grouped into two main categories linear and nonlinear. Linear ones such as logistic regression or linear regression, they are efficient but the model cannot learn complex interactions among input features. This problem can be addressed by applying a nonlinear transformation $\phi(x)$ on the input data. Choosing the right function ϕ is not always easy. In deep learning the exact function is supposed to be learned during training of the model. Given a model $y = f(x; \theta, w) = \phi(x; \theta)^T w$ the goal is to learn the parameters θ in order to adjust a generic function ϕ and then map the transformed features via the weights w. The generic function is also known as the activation function and it is usually chosen by a human, except for the cases of hyper-parameter optimization where this design choice is automated.

After designing a deep feedforward network, it has to be trained in order to fit the data distribution. The most popular training method is backpropagation through gradient learning. The cost function is driving the process of training. The model is a parametric model that defines a distribution $p(y|x; \theta)$ and by using the principle of the maximum likelihood the cost function is usually the cross-entropy between the training data and the model's predictions. In order to avoid overfitting it is a common practice to include a regularization term in the cost function.

2.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are pioneered by LeCun et al. (1989). They take advantage of the topology of the data such as images and time series. The former ones are seen as 2D grids and the latter ones as 1D grids by taking samples at regular timeframes. The fundamental attribute of CNNs is the mathematical operation named convolution. The main difference against classic neural networks is that at least one of the layers performs the operation of convolution instead of a matrix multiplication.

Typically, the process employed in a convolutional neural network does not fully match the definition of convolution as defined in pure mathematics. In general convolution is an operation between two functions and is symbolized with an asterisk.

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\,da \tag{2.14}$$

In the context of deep learning the function x is referred as the input, w as the kernel and the output as the feature map. The discrete version of convolution formula is given as follows:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.15}$$

Discrete convolution can be thought of as matrix multiplication, except the matrix has some elements that must be equal to other entries. For univariate discrete convolution, for example, each row of the matrix must be equal to the row above shifted by one element, which results in a Toeplitz matrix. Another property of convolutional layers is that they lead to sparse features because the kernel size is much smaller than the cardinality of the input data.

Three critical properties of a machine learning system that are provided by the operation of convolution include sparse interactions, parameter sharing and equivariant representations (Goodfellow et al., 2016). Sparsity is achieved by using a kernel with much smaller size than the dimension of the input data. Sparse representations lead to reduced memory complexity and improves statistical efficiency. It also reduces the computational complexity as it outputs less features. Matrix multiplication takes mn parameters if there are

m inputs and n outputs, and the algorithms used in reality have $\mathcal{O}(mn)$ runtime. By limiting the connections of each output to k, then we get a sparse connectivity with $||\backslash$. Parameter sharing refers to reusability of some parameters of the model and is also known as tied weights, because it is the value of the weights that is reused multiple times. Each member of the kernel is used at every point of the input in a convolutional neural net. Because of the parameter sharing employed by the convolution operation, we learn only one set of parameters rather than a different set for each location. The computational complexity is not affected but the memory is significantly reduced when compared against the case of a fully connected network. Finally, parameter sharing leads to equivariance to translation. A function is equivariant if for any change to its input, the output is also changing. Hence, a function $f(x)$ is equivariant to g if $f(g(x)) = g(f(x))$. In the context of convolution, let g be a shifting function, then the convolution is equivariant to g. In time series data this can be interpreted as a snapshot of the produced features by the convolution, across timeline. On the other hand convolution is not guaranteed to be equivariant for all transformations. For example with regards to image transformations such as scaling or rotation, convolution is not equivariant.

A convolutional layer typically has three stages. Firstly, the layer executes many convolutions in parallel in order to generate a set of linear activations. Each linear activation is then passed via a nonlinear activation function, such as the rectified linear activation function (ReLU). The second stage is also known as the detector stage. In the third stage, we utilize a pooling function to further change the layer's output. In the deep learning literature there are many variations of convolutional layers. Most of them enforce spatial constrains as prior knowledge depending on the nature of the problem and the equivariance properties that are required to execute successfully an AI task.

The third stage of a convolutional layer is often considered a separate layer. Pooling in conjunction with convolution are considered infinitely strong priors. A strong prior in general has low entropy in contrast to weak priors such as a Gaussian distribution with high variance. An infinitely strong prior excludes some parameters by setting their probability equal to zero. One important realization is that convolution and pooling can result in underfitting. Convolution and pooling, like any prior, are only beneficial when the prior's assumptions are legitimate.

2.3.3 Pooling Operation

As it has been discussed previously pooling can be part of a convolutional layer or an independent one. It imposes spatial constraints by substituting specific regions of the output of a previous layer with a summary statistic of neighboring features. One popular pooling operation is max pooling introduced by Zhou and Chellappa (1988), which selects the feature with the maximum value. Other popular pooling operations are the average, the $L2$ norm and a weighted average based on the distance from a reference feature.

The goal of pooling is to leverage the invariance properties of a neural network so that its predictive performance is not heavily affected by small translations of the input data. Invariance to local translation can be a beneficial characteristic if we care more about if a feature exists than where it is located (Goodfellow et al., 2016). Pooling can be thought of as adding an infinitely strong prior that the function learned by the layer must be invariant to tiny translations. When this assumption is valid, it can significantly increase the network's statistical efficiency. Pooling usually subsumples its input by using a kernel with cardinality k and thus estimating summary statistics within this kernel. As a result the next layer will have to process k less features. This has a great impact not only on computational complexity but also on memory consumption, as some layers such as fully connected ones their number of parameters depends on their input.

Pooling can also handle inputs of varying sizes, which is essential in many real-world applications. For instance, in image recognition where the input data can be images of different sizes, the input size of a classifier has to be fixed. This issue is addressed by altering the size of an offset between pooling areas, ensuring that the classification layer always receives the same number of summary statistics regardless of input size. The network's final pooling layer may be configured to output four sets of summary statistics, one for each quadrant of an image, independent of image size.

From the theoretical point of view, pooling operations have been studied by Boureau et al. (2010), aiming to answer the question which pooling function should we choose given an AI task. Other modern approaches of pooling include dynamically pool features together utilizing a clustering algorithm and producing different sets of pooling areas for each input data (Boureau et al., 2011). Another method is to learn a single pooling structure and then apply it to all images (Jia et al., 2012).

2.3.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are specialized for sequential data. One fundamental idea that constituted in the successful implementation of RNNs is sharing parameters among various components of a neural network. Because of parameter sharing, the model can be extended and applied to examples of other forms such as different lengths and generalized across them. We couldn't generalize to sequence lengths not encountered during training if we had distinct parameters for each value of the time index, and we couldn't share statistical strength across different sequence lengths and places in time if we had separate parameters for each value of the time index. This type of sharing is especially crucial when a certain piece of information can occur at various points in the sequence. For instance consider two phrases that provide the same information but in a different syntax e.g. "Coronaviruses are a group of related RNA viruses that cause diseases in mammals and birds" and "Mammals and birds are infected by a group of related RNA viruses named coronaviruses". If the task is to answer the question what type of virus is coronavirus, we should develop a model that can provide the right answer which is "RNA" regardless of the index of that word in the sentence. The benefit of RNNs against classic feedforward neural networks is that they can capture and remember the information while processing the sentence sequentially.

For convenience, we will refer to RNNs as acting on a sequence of vectors $x(t)$, with the time step index t spanning from 1 to t. In reality, recurrent networks typically operate on minibatches of such sequences, with each member of the minibatch having a different sequence length. RNNs can also be used in two dimensions over spatial data such as pictures, and when applied to time-related data, the network can contain connections that run backward in time, as long as the entire sequence is seen before it is fed into the network.

There are three basic types of recurrent architectures (Goodfellow et al., 2016). The first one regards recurrent neural networks that produce an output at each time step and maintain recurring connections between hidden units. The second type includes networks that generate an output at each time step and have recurrent connections from the output at one time step to the hidden units at the next time step. Finally there are recurrent architectures that have recurrent connections between hidden units and read an entire sequence before producing a single output. Next, an instance of the first type of recurrent neural network is described as a reference.

Assume that the activation function that is used is the hyperbolic tangent function. Also assume that the output is discrete such as predicting words of a text. A reasonable way to describe discrete variables is to consider the output $\mathbf{o}$ to be the unnormalized log probabilities of each discrete variable's potential values. The softmax operation can then be applied as a post-processing step to create a vector $\widehat{\mathbf{y}}$ of normalized probabilities over the output. The initial state is $\mathbf{h}^{(0)}$ and for each timestep we have the following updates:

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \tag{2.16}$$

$$\mathbf{h}^{(t)} = tanh(\mathbf{a}^{(t)}) \tag{2.17}$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)} \tag{2.18}$$

$$\widehat{\mathbf{y}}^{(t)} = softmax(\mathbf{o}^{(t)}) \tag{2.19}$$

, where the parameters are the bias $\mathbf{b}$ and $\mathbf{c}$ and the weight matrices $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{V}$. This is an example of a recurrent network that translates an input sequence to the same length output sequence. The total loss for a given sequence of x values paired with a sequence of y values is just the sum of the losses across all time steps. It is an expensive process to compute the gradient of this loss function with respect to the parameters. The gradient computation takes place in two main steps. Firstly it includes a forward propagation pass following the order of the sequence and then it performs a backward propagation pass moving backwards in the computation graph. Due to the fact that the forward propagation graph is sequential, each time step can only be computed after the preceding one, the runtime is $\mathcal{O}(\tau)$ and cannot be decreased by using parallelization. States computed in the forward pass must be retained until reused in the backward pass, hence the memory cost is also $\mathcal{O}(\tau)$. This algorithm is called back-propagation through time (BPTT).

One fundamental challenge in recurrent neural networks is that gradients propagated over multiple layers either vanish or explode and the model cannot converge to a solution. Even if we assume that the parameters are such that the recurrent network is stable storing memories, with gradients not exploding, the difficulty with long-term dependencies arises from the exponentially smaller weights assigned to long-term interactions. This happens because of the the multiplication of many Jacobians, compared to short-term interactions. The problem of exploding and vanishing gradients has been studied by many researchers and there is a rich literature where it is analyzed in great detail (Hochreiter, 1991; Doya, 1993; Bengio et al., 1994; Pascanu et al., 2013). It has been introduced independently by Hochreiter (1991) and Bengio et al. (1993). One could think that the problem can be avoided merely by remaining in a parameter space region where the gradients do not vanish or burst. Unfortunately, in order to store memories that are resilient to minor perturbations, the RNN must enter a region of parameter space where gradients disappear Bengio et al. (1993, 1994). When the model can describe long-term dependencies, the gradient of a long-term interaction has an exponentially smaller size than the gradient of a short-term interaction. This does not mean that learning long-term dependencies is impossible, but that it may take a very long time because the signal regarding these dependencies will tend to be hidden by the smallest fluctuations coming from short-term dependencies Bengio et al. (1994). Bengio et al. (1994) showed that as the span of the dependencies that must be captured increases, gradient-based optimization becomes increasingly difficult. The probability of successful training of a traditional RNN via SGD rapidly approaches 0 for sequences of size no longer than 20.

Two of the most successful recurrent neural architectures are the long short-term memory (LSTM) and the gated recurrent unit (GRU). These two architectures belong to the same type of recurrent nets called gated RNNs. Gated RNNs are built on the idea of establishing paths through time with derivatives that do not vanish or explode. LSTM architecture is based on the fundamental concept of self-loops that produce paths for the gradient to flow without vanishing and was proposed by Hochreiter and Schmidhuber (1997). The time scale of integration can be modified dynamically by having the weight of this self-loop gated in the sense that it can be controlled by another hidden unit. In this situation, we imply that even with a fixed-parameter LSTM, the time scale of integration might change depending on the input sequence, because the time constants are output by the model itself. LSTM architecture has thrived in many applications that involve sequential data such as unconstrained handwriting recognition (Graves et al., 2008), speech recognition (Graves et al., 2013, 2014), handwriting generation (Graves et al., 2013), machine translation (Sutskever et al., 2014), image captioning (Kiros et al., 2014; Vinyals et al., 2015b; Xu et al., 2015), and parsing (Vinyals et al., 2015a). GRU architecture has been proposed in an effort to reduce the complexity of LSTM architecture (Cho et al., 2014a; Chung et al., 2014, 2015; Chrupała et al., 2015; Jozefowicz et al.,

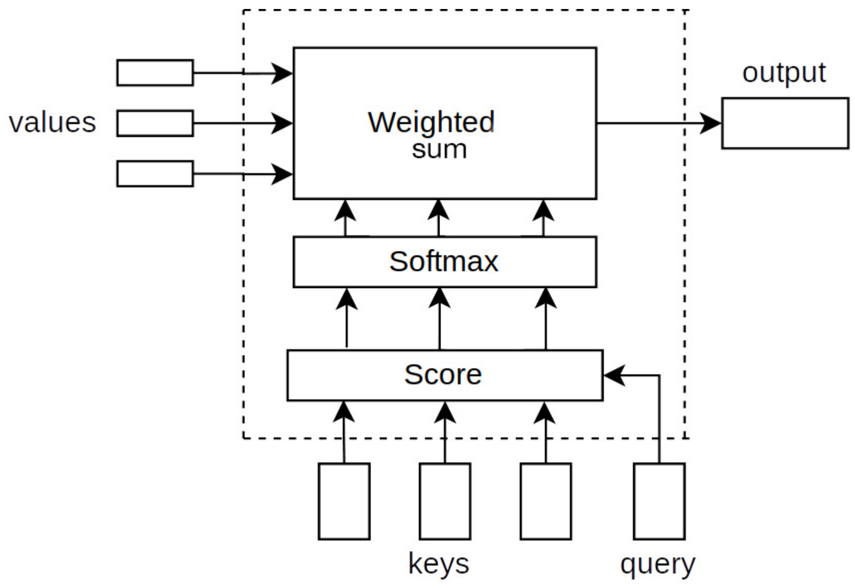


Figure 2.3.1: Inside Attention mechanism.

2015). The main difference between GRU and LSTM is that in the former architecture a single gating unit controls both the forgetting component and the choice to update the state unit.

2.3.5 Attention Mechanism

The extraction of input-output relations is a common task in machine learning and pattern recognition, with uses in image captioning, machine translation etc. Sequence to sequence models (seq2seq) consist a go-to approach regarding the Deep Learning techniques. The original seq2seq model, as proposed by Sutskever et al. (2014), contains two major components; the encoder and the decoder. Essentially these components are two RNNs. The role of the encoder is the compression of the sequence input into a vector of fixed length, known as the *context*. The intuition is that this vector suppresses the most important information of the source input. Given the same context vector, the decoder is capable to re-construct the source sequence. The drawback of this architecture is that it fails to process very long sequences, due to the fixed length of the context vector.

To improve the efficiency of seq2seq architecture, Bahdanau et al. (2015) proposed Attention. This mechanism gives the decoder the power to concentrate on the parts of the input that matter the most, in relation to the corresponding output. At each time step of the decoder, Attention calculates the relations between the entire input sequence and the decoder output. These calculations create an alignment vector, that contains the score between the input's sequence and the decoder's output at the corresponding step. The resulting context vector is a combination of both the alignment vector and the encoder's output.

Considering how the scores and alignments are calculated, the most popular types of attention are the Additive (Bahdanau et al., 2015) and the Multiplicative/Dot (Luong et al., 2015). In a different setting called Self-Attention (Cheng et al., 2016), the attention mechanism is applied on the same sequence, in order to relate different parts of it. Self-Attention can integrate either Bahdanau's or Luong's scoring methods. As inputs to an Attention layer three kinds of vectors are given; a query, a key and a value. The layer output is calculated as described bellow. A summary of the steps is shown in Fig.**??**.

To begin with, the similarity between a query (q) and a key (k_i) is calculated, estimating for each query-key pair, a score (a_i).

$$a_i = score(q, k_i) \tag{2.20}$$

Next, a softmax function is used to normalize the scores in order to sum up to one. The attention weights are obtained as follows.

$$b_i = \frac{exp(a_i)}{\sum_j \exp(a_j)} \tag{2.21}$$

The final output is the weighted sum of the values (v):

$$output = \sum_{i=1}^{n} b_i v_i \tag{2.22}$$

Between Additive and Dot attention mechanisms the scoring function differs. Specifically, the dot scores are given by the dot product of keys and queries. On the other hand, the additive scoring function is a non-linear sum.

2.4 Information Theoretic Methods in Deep Learning

Deep learning has thrived in many AI tasks surpassing human performance. Yet, there is no complete theory that explains the learning dynamics of deep neural networks or overaparameterized machine learning models. There are many unanswered questions that classic statistical learning theory cannot explain. Some of these mysteries include the double descent phenomenon (Nakkiran et al., 2021), the fact that overparameterized models generalize so well (Brutzkus and Globerson, 2019), the role of the depth of deep neural networks in learning (Kawaguchi et al., 2019). Escaping saddle points during optimization (Jin et al., 2017) and others. A formal mathematical theory is needed in order to be able to design deep neural networks efficiently when given a task and a dataset, rather than going through the process of trial and error. Information theory is a very promising mathematical tool that could bridge the gap between theory and practice. This section discusses the role of information theory in deep learning and presents recent advancements.

2.4.1 Quantities of Information

Let a continuous random variable X and its probability density function (PDF) $f(x)$. Then Shannon's differential entropy is given by the following formula:

$$H(X) = -\int_{\mathcal{X}} f(x) \log f(x) dx = \mathbb{E}(-\log f(x)) \tag{2.23}$$

Given a second continuous random variable Y and their joint probability density function $f(x, y)$ we have:

$$H(X, Y) = -\int_{\mathcal{X}} \int_{\mathcal{Y}} f(x, y) \log f(x, y) dx dy \tag{2.24}$$

In information theory it is very important to measure the common information between two random variables. The metric to measure the common information is the mutual information and is given by the following formula:

$$I(X; Y) = \int_{\mathcal{X}} \int_{\mathcal{Y}} f(x, y) \log \left(\frac{f(x, y)}{f(x) f(y)} \right) dx dy \tag{2.25}$$

Mutual information has the property of being symmetric. It is equal to zero when the two random variables encapsulate different information and thus they are independent.

Differential entropy comes with some shortcomings which make it unsuitable as a measure of entropy. Two desired properties that it doesn't fulfill are that it can be negative and that it is not invariant under change of variables (Cover and Thomas, 2012). These shortcomings are attributed to the fact that differential/continuous entropy was firstly defined by analogy just replacing the summation with an integral, whereas the correct way should be by taking the limits of the axiomatic definition of entropy. In order to measure the dissimilarity between two random variables, relative entropy is introduced. It is also known as Kullback-Leibler divergence (KL divergence or D_{KL}) and its formula is given below.

$$D_{KL}(f(x)\|g(x)) = \int_{\mathcal{X}} f(x)\log\left(\frac{f(x)}{g(x)}\right)dx. \tag{2.26}$$

Mutual information can also be expressed in terms of KL divergence of the joint distribution of two random variables $f(x, y)$ and the product of their marginal distributions according to the following formula:

$$I(X;Y) = D_{KL}(f(x,y)||f(x)f(y)) \tag{2.27}$$

In practice the probability distribution of random variables is not known and the estimation of MI is not trivial. One way of estimation of the MI is using non-parametric density estimation methods such as kernel density estimator (KDE) (Parzen, 1962). Such methods usually approximate the marginal and joint probability distributions and then MI can be calculated. KDE are not efficient in high dimensions though. Other methods that exist are based on nearest neighbor and graph properties (Kraskov et al., 2004). Noshad et al. (2019) recently proposed a scalable solution named ensemble dependency graph estimator (EDGE). Despite the efficiency of EDGE and other similar approaches the drawback is that they are not differentiable and hence they cannot be used for gradient based optimization. Belghazi et al. (2018) propose a variational approach of density estimation using neural networks and overcoming the problem of differentiability. A detailed theoretical work on the benefits and drawbacks of modern denstity estimators is presented by McAllester and Stratos (2020).

2.4.2 Information Theoretic Learning Principles

Information theory has played a significant role in machine learning. Information measures are extensively used as objective functions in deep learning. Mean squared error (MSE) or cross-entropy are two of the most popular loss functions when training neural networks. Despite the wide usage of these functions, it has been shown that the trained models result in poor generalization performance. It is worth mentioning that they can fit even random labeled data, meaning that the learned representations don't have the properties of optimal representations (Jiang et al., 2018; Zhang et al., 2021a; Feng et al., 2021). A robust criterion against outliers and out-of-distribution data is the maximum correntropy criterion (MCC). Utilizing MCC the model is trained by maximizing the correntropy between the labels y and the model's predictions $\widehat{y}$ as follows (Liu et al., 2007):

$$f^*_{MCC} = \underset{f\in\mathcal{F}}{\arg\max} V_\sigma(\hat{y}, y) = \underset{f\in\mathcal{F}}{\arg\max} \mathbb{E}\left(G_\sigma(e)\right), \tag{2.28}$$

where f^* is the optimal model, $\mathcal{F}$ stands for models hypothesis space, $e = y - \hat{y}$ is the prediction residual, $\mathbb{E}$ refers to mathematical expectation, and $V_\sigma(\hat{y}, y) = \mathbb{E}\left(G_\sigma(e)\right)$ denotes the correntropy between $\hat{y}$ and y, with $G_\sigma(e)$ being the Gaussian kernel function with width σ. Erdogmus and Principe (2002) propose the minimum error entropy (MEE), where the objective is to minimize the entropy of the prediction residual e. MEE has shown strong generalization and robustness to data perturbations when they are used for training neural networks (Yu et al., 2021). Xu et al. (2019) propose a new loss function, called determinant based mutual information (LDMI). This method is robust to label noise by utilizing a generalized mutual information which has the properties of being information-monotone and relatively invariant. The improvement of the performance of machine learning models on out-of-distribution data, attracted the attention of researchers to develop more information-theoretic metrics for robust deep learning models. Greenfeld and Shalit (2020) utilize the Hilbert Schmidt Independence Criterion (HSIC) (Gretton et al., 2007) and prove that deep learning models are robust to covariate shift if and only if the distribution of the prediction residuals is statistically independent of the distribution of the data. A most recent interesting approach is the relation of such measures with causal representation learning and the emergence of a new concept called independent mechanism analysis (IMA) (Gresele et al., 2021).

Another concept from information theory that has been an integral part of classic and modern machine learning models is the principle of maximum entropy also known as MaxEnt or InfoMax (Burg, 1975). InfoMax has been introduced to machine learning by Linsker (1988). A very successful application of InfoMax in neural networks is the seminal work of Bell and Sejnowski (1995) for the problem of blind source separation. InfoMax has been used in many other domains such as computer vision and natural language processing with the objective to maximize the mutual information between the input data and the latent representation in hidden layers (Oord et al., 2018; Hjelm et al., 2018; Kong et al., 2019). It has also been an integral part in moder deep learning architectures such as generative adversarial networks (GANs) (Chen et al., 2016) and graph neural networks (GNNs) (Velickovic et al., 2019). Being used as a design principle, InfoMax has also influenced the way that deep architectures are trained. Instead of the conventional way where training happens through backpropagation and all the layers are trained simultaneously, new methods suggest training each layer independently (Hu and Principe, 2021; Duan et al., 2020). The main drawback of InfoMax is that it can introduce a lot of noise or useless information regarding the target task y. This issue is addressed by the information bottleneck theory.

Information bottleneck (IB) method was firstly introduced by Tishby et al. (2000). It is a generalization of rate distortion theory. Given two random variables X and Y, the method describes how X can be compressed into another variable T retaining as much information about Y as possible. Unfortunately, there is no analytical solution to the original formulation and most solutions are approximations. One approximation is derived by assuming that X and Y are jointly Gaussian distributed random variables and the methodology is known as Gaussian information bottleneck (Chechik et al., 2005). This work is extended by Rey et al. (2014) by introducing a sparse variant of IB that compresses the original data to maintain information in only a few selected dimensions. Information bottleneck is a unified framework that is applicable in many information theoretic learning systems. In the domain of statistical learning theory it was firstly introduced by Shamir et al. (2010) and later on it was introduced to deep learning by Tishby and Zaslavsky (2015) contributing to both theoretical and practical aspects of neural networks. In the latter work the authors show that the optimal architecture of a deep neural network is related to the compression of the input layer with respect to the output. This is depicted in a plot, called the Information Plane, which is the plane of the mutual information values that each layer preserves on the input and output variables. An example of a plot of the information plane during training of a neural network is depicted in 2.4.1

Information Plane has been recently used to reveal more details about the dynamics of training a neural network Shwartz-Ziv and Tishby (2017). A meticulous analysis of the entropic dynamics of neural networks is presented by Saxe et al. (2019), taking into account the Information Bottleneck and arguing that the idea of a diffusion and a compression phase during training is not generic. The authors show that this statement holds only for the special case of double-sided saturating nonlinearities like tanh and not for ReLU or linear functions. A rigorous mathematical description of the exact conditions that the phases of diffusion and compression are present in the training of a neural network is still missing. Next, the original formulation of information bottleneck as well as other variations are presented.

Let two random vectors X and Y, then the goal of information bottleneck is to find a new random vector T which is a compression of X and at the same time retains as much as possible information about Y. This is a constraint optimization problem and is defined as follows:

$$\min_{P(T|X)} I(X;T) - \beta I(T;Y) \tag{2.29}$$

,where β is a parameter that balances the amount of compression of X and the amount of information

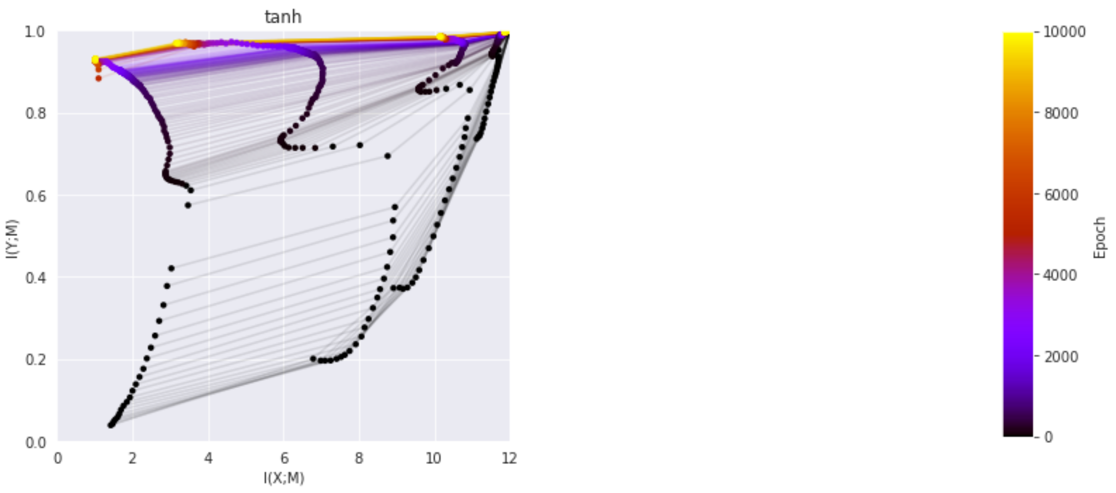


Figure 2.4.1: Illustration of an example of an information plane. The activation function of the hidden layers is tanh.

from Y. The optimal conditional distribution $P(T|X)$ is the solution of the problem. In general there is no known analytical solution, but for the case of discrete random vectors X and Y the Blahut-Arimoto algorithm can be utilized (Tishby et al., 2000). As it has been mentioned previously, if we assume that the joint probability distribution of X and Y is a Gaussian the problem can be simplified significantly.

$$(X, Y) \sim \mathcal{N}\left(0, \begin{pmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{XY}^\top & \Sigma_Y \end{pmatrix}\right) \tag{2.30}$$

Then we have a special case of information bottleneck, called Gaussian information bottleneck (Chechik et al., 2005) and the latent random vector T is also Gaussian and more specifically a linear projection of X with an additive Gaussian noise that is independent of X.

$$T = AX + \epsilon, \epsilon \sim N(0, \Sigma_\epsilon) \tag{2.31}$$

Therefore:

$$T \sim N(0, A\Sigma_X A^\intercal + \Sigma_\epsilon \tag{2.32}$$

Then the IB optimization problem is described as follows:

$$\min_{A,\Sigma_\epsilon} I(X; AX + \epsilon) - \beta I(AX + \epsilon; Y) \tag{2.33}$$

Given that for Gaussian random variables with n dimensions and Σ_X $\Sigma_{X|Y}$ the covariance matrices of X and $X|Y$ accordingly, the mutual information is described as follows:

$$I(X;Y) = H(X) - H(X|Y) = \frac{1}{2}\log((2\pi e)^n|\Sigma_X|) - \frac{1}{2}\log((2\pi e)^n|\Sigma_{X|Y}| \tag{2.34}$$

The main benefit of the Gaussian IB is that it has an analytical solution, based on the assumption that $T - X - Y$. Let β_0 a fixed value for β, then equation 2.33 is optimized by $_\epsilon = I$ and A depends on Σ_X and eigenvalues of $\Sigma_{X|Y}\Sigma_Y^{-1}$. The Gaussian IB consists the basis for another variation of IB that ensures sparsity of the compression and is called sparse Gaussian IB. Sparsity is accomplished by enforcing the projection matrix A to be diagonal. Rey et al. (2014) prove that since $\log|A\Sigma A^\top + I| = \log|\Sigma A^\top A + I|$ for any positive definite Σ and symmetric A, equation 2.33 is simplified to minimization over diagonal matrices with positive entries $D = A^\top A = diag(a_1^2, ..., a_n^2) = diag(d_1, ..., d_n)$ and is given by:

$$\min_{D=diag(d_1,...,d_n)} I(X; AX + \epsilon) - \beta I(AX + \epsilon; Y) \tag{2.35}$$

The computational complexity to find solutions of the IB method lead to variational approaches that approximate a lower bound. Alemi et al. (2016) propose the deep variational information bottleneck (DVIB). The authors parameterize the conditional distributions $P(T|X)$ and $P(Y|T)$ with neural networks and the mutual informations from equation 2.29 are:

$$\begin{aligned} I(X;T) &= D_{KL}(P(T \mid X)P(X) \| P(T)P(X)) \\ &= \int P_{\Phi}(T \mid X)P(X) \log \tfrac{P(T|X)}{P(T)} \mathrm{d}x \, \mathrm{d}t \\ &= \mathbb{E}_{P(X)} D_{KL}(P(T \mid X) \| P(T)) \end{aligned} \tag{2.36}$$

$$\begin{aligned} I(T;Y) &= D_{KL}\left(\left[\int P(T \mid Y, X)P(Y, X)\mathrm{d}x\right] \| P(T)P(Y)\right) \\ &= \int P(T \mid X, Y)P(X, Y) \log \tfrac{P(Y|T)P(T)}{P(T)P(Y)} \mathrm{d}t \, \mathrm{d}x \, \mathrm{d}y \\ &= \mathbb{E}_{P(X,Y)} \left[\int P(T \mid X, Y) \log P(Y \mid T)\mathrm{d}t\right] \\ &\quad - \mathbb{E}_{P(X,Y)} \left[\log P(Y) \int P(T \mid X, Y)\mathrm{d}t\right] \\ &= \mathbb{E}_{P(X,Y)} \mathbb{E}_{P(T|X,Y)} \log P(Y \mid T) + H(Y), \\ &= \mathbb{E}_{P(X,Y)} \mathbb{E}_{P(T|X)} \log P(Y \mid T) + H(Y) \end{aligned} \tag{2.37}$$

The last equality in equation 2.37 is based on the first assumption of the IB framework $T - X - Y$ and hence $P(T|X, Y) = P(T|X)$. The conditional distribution $P(Y|T)$ is parameterized by a neural decoder with parameters θ and is estimated by using the reparameterization trick (Kingma et al., 2015a). The latent vector T is estimated by approximating the $P(T|X)$ using a neural encoder with parameters ϕ. The authors of DVIB method also state that the entropy $H(Y)$ in equation 2.37 can be missed as Y is fixed.

Wieczorek et al. (2018) argue that missing the entropy $H(Y)$ in equation 2.37 has some shortcomings because the IB solution becomes no more invariant to monotonic transformations of X or Y. Indeed, this invariance is a property attributed to the invariance of the quantity of mutual information. For instance for an invertible function f we have $I(X;T) = I(f(X);T)$. Other issues regarding the marginal conditional distributions in DVIB are discussed by (Wieczorek et al., 2018). The deep copula information bottleneck is proposed as an alternative to DVIB (Wieczorek et al., 2018). The data are subject to a copula augmentation such as $\widehat{X} = \Phi^{-1}(\widehat{F}(X))$, with Φ being a Gaussian and $\widehat{F}$ an empirical cumulative distribution function. The method is also illustrated in figure 2.4.2. The copula augmentation further helps to encapsulate the method that is proposed by Rey et al. (2014) and produce a sparse latent representation T. Hence, deep copula information bottleneck demonstrates better disentanglement than the deep variational information bottleneck method.

DVIB has many similarities with variational autoencoders (VAEs) and it has been shown that both methods approximate the same lower bound. For the case of DVIB the posterior $P(Y|T)$ can be approximated by $Q(Y|T)$ and by using the Kullback-Leibler divergence $D_{KL}(Q(Y|T)||P(Y|T)) \geq 0$ equation 2.37 gives (Alemi et al., 2016):

$$I(T;Y) \geq \mathbb{E}_{P(X,Y)} \mathbb{E} \log Q(Y|T) + H(Y) \tag{2.38}$$

Researchers have proposed several bounds in order to estimate the mutual information and construct better latent representations. Alemi et al. (2018) establish variational lower and upper bounds on the mutual information between the input and the latent variable, then use these bounds to construct a rate-distortion curve that encapsulates the trade-off between compression and reconstruction accuracy. The authors demonstrate the existence of a family of models with similar ELBO but distinct quantitative and qualitative properties. Alemi and Fischer (2018) then extend this bound to the scenario where it is sample-independent, enabling comparisons between VAE and generative adversarial networks. Painsky and Tishby (2017) bound the mutual information terms in the information bottleneck using a Gaussian relaxation and then they compare their newly developed method to canonical correlation analysis.

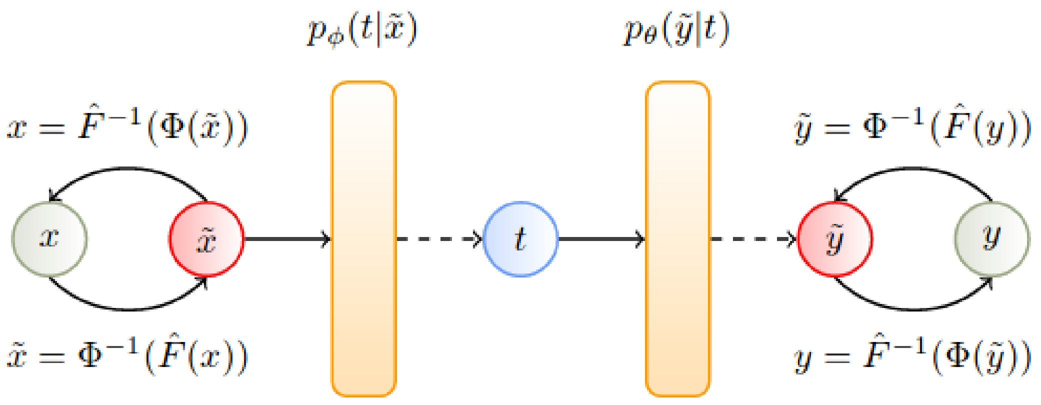

$p_\phi(t|\tilde{x})$
$p_\theta(\tilde{y}|t)$
$x = \hat{F}^{-1}(\Phi(\tilde{x}))$
$\tilde{y} = \Phi^{-1}(\hat{F}(y))$
x
$\tilde{x}$
t
$\tilde{y}$
y
$\tilde{x} = \Phi^{-1}(\hat{F}(x))$
$y = \hat{F}^{-1}(\Phi(\tilde{y}))$

Figure 2.4.2: Deep information bottleneck with the copula augmentation. Green circles represent random variables, whereas orange rectangles represent deep neural networks that parameterize the random variables. The blue circle represents latent random variables, whereas the red circle represents copula transformed random variables. (Wieczorek et al., 2018)

3 Time Series Problems

"Time is the coin of your life. It is the only coin you have, and only you can determine how it will be spent. Be careful lest you let other people spend it for you."
— Carl Sandburg

The research directions of this thesis are driven by real world applications. Following a task oriented approach we aim to magnify the impact of our scientific contributions. Our ideas are inspired by applications of ambient intelligence in a smart home with a focus on energy efficiency and eldercare. This chapter reviews some fundamental time series problems including blind source separation, classification and regression tasks with application in the domains of energy, speech and health. Firstly, a survey regarding the problem of non-intrusive load monitoring is presented. Then the importance of the problem of speech commands classification is discussed, emphasizing the necessity to develop a better human to machine communication interface. Finally, the rise of the internet-of-things and the plethora of sensors is an opportunity for the emergence of the field of predictive health.

3.1 Non-Intrusive Load Monitoring

3.1.1 *Introduction*

Climate change has been one of the toughest challenges for humanity. Global warming, acid rains, air and water pollution, depletion, disruption of our natural environment are only a small number of climate change threats. One of the major factors to face these problems is energy efficiency. Global energy demand has increased by 16 times in the twentieth century, whereas the population has increased fourfold (Kamat, 2007). The governments of many countries have realized the importance of climate change and have made provisions to limit the annual carbon emissions and urban waste by 2050 (climate change act, 2008). In this direction, traditional fossil fuels expected to be replaced by non-conventional energy sources, whereas current electricity infrastructure will be transformed to a smart grid.

According to estimates, 40% of the carbon dioxide emissions in the USA relates to energy consumption that comes from electric power. Residential and commercial buildings account for about 40% of total energy consumption, 20% of which can be saved by applying efficiency improvements Carrie Armel et al. (2013). The significance of these savings is highlighted by the fact that approximately 10–15% of the electricity consumed by US homes requires 200 billion kWh per annum. The equivalent energy production of 16 nuclear power plants or 81.3 million tons of coal (Froehlich et al., 2011).

Reducing electricity consumption in buildings can significantly drop energy wastage. According to studies, energy monitoring and direct feedback to the users, such as personalized recommendations or real-time consumption on appliance level, are extremely valuable. Electricity bills could be reduced and residents would be aware of their house energy profile (Darby, 2006).

A key consideration for an efficient and accurate energy monitoring in a domestic building is the problem of power disaggregation. The benefits of providing analytical energy consumption of residential buildings are not restricted to the residents. Valuable appliance data can empower research and development, help redesigning household appliances and improve building operational efficiency. Utilities and policies will be improved by providing more accurate energy consumption forecasting, more efficient economic models, reformed funding allocations, better energy building evaluation, smart grid optimization etc. (Carrie Armel et al., 2013).

The problem of breaking down the total power signal to several appliance level signals, using non-intrusive methods, was firstly introduced by Hart (1992) as non-intrusive load monitoring (NILM). Since then, NILM has been the preferred method for power disaggregation in contrast to other pervasive methods, where appliance level data has to be collected. The reasons that researchers and engineers prefer the NILM method are both economic and practical. This means that NILM research is not only focused on theoretical models, but also on the deployment of these systems in the real world. A large-scale deployment favors NILM over ILM, because it offers lower costs, there is no need for multiple sensor configuration and installation is much simpler. The only advantage of intrusive methods is the high accuracy in measuring energy consumption of specific appliances.

3.1.1.1 Power Disaggregation

The purpose of power disaggregation is to break the total power consumption down into its components. In a domestic building, the resultant power is the outcome of the power consumption of each electrical device. Thus, the problem is to identify how much power each appliance consumes. The superimposition of the power of N devices in a time period T, can be defined as:

$$P(t) = P_{noise}(t) + \sum_{i=1}^{N} p_i(t), t \epsilon \{1, T\} \tag{3.1}$$

where p_i is the power of each appliance and P_{noise} is the power of unknown signal, which is considered noise.

In the literature there are various problem formulations. The most popular one aims to estimate each p_i for $i = 1, 2, ..., N$ devices, given only the total power $P(t)$. A more practical and business oriented approach of the problem would be described as follows: "estimate the cost of electricity, consumed by the oven in the time frame of a month" (Dong et al., 2014). The former approach is the principal goal and some formal definitions are presented below.

Given a discrete sequence of observed aggregate power readings $x = x_1, x_2, ..., x_T$ mine the sequence of appliance power demands $w^{(n)} = w_1^{(n)}, w_2^{(n)}, ..., w_n^{(n)}$, where n is one of N appliances. Alternatively, this problem can be represented as the determination of appliance states $z^{(n)} = z_1^{(n)}, z_2^{(n)}, ..., z_n^{(n)}$, if a mapping between states and power demands is known. Each appliance state corresponds to an operation of approximately constant power draw (e.g. 'on', 'off' or 'standby') and t represents one of T discrete time measurements (Batra et al., 2014b; Parson et al., 2011, 2012)

A similar definition from Batra et al. (2014a) is:

The aim of energy disaggregation is to provide estimates, $\hat{y}_t^{(n)}$, of the actual power demand, $y_t^{(n)}$, of each appliance n at time t, from household aggregate power readings, $\overline{y}_t$. Most NILM algorithms model appliances using a set of discrete states such as off, on, intermediate, etc. We use $x_t^{(n)} \epsilon Z > 0$ to represent the ground truth state, and $\hat{x}_t^{(n)}$ to represent the appliance state estimated by a disaggregation algorithm.

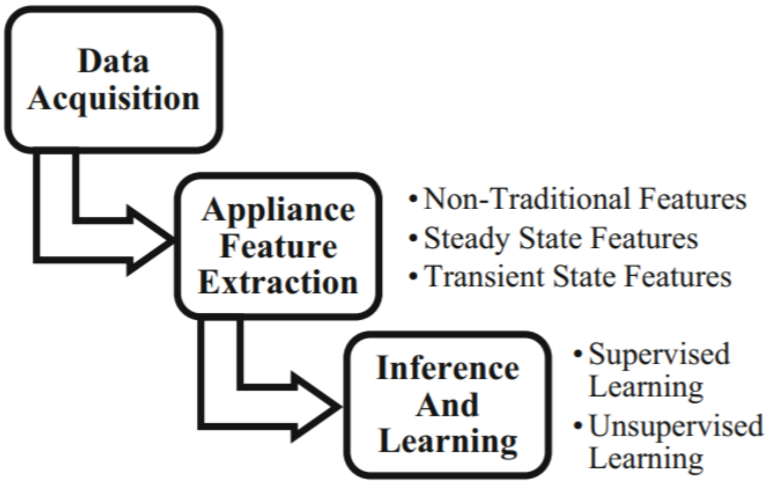


Figure 3.1.1: The main three steps of NILM framework.

3.1.1.2 NILM Framework

In a NILM system, there is a smart meter connected to the central electrical equipment of a residence and collects energy consumption data. The records include the total consumption of the residence. Then, energy disaggregation models can estimate the consumption of each appliance that is working in the house. The three basic steps of the whole procedure are described by the NILM framework (Carrie Armel et al., 2013; Burbano, 2015) and are depicted in Fig. 3.1.1.

The first step of the NILM framework is the data acquisition one. It refers to the methodology the energy records are collected. In this step the hardware that is used to take the measurements plays a significant role. One of the most important features of the sensors is the sampling rate which affects the performance of the disaggregation algorithms later on. The equipment is usually smart meters which are becoming very popular in many countries. This equipment has low cost and is easy to install. As far as the properties of the recorded data are concerned, the most important ones are the type of power, the power level resolution and the sampling frequency. The types of power are real and reactive and smart meters can record one of them or both. The second property is the smallest measurement of power that the sensors can detect. The last property can be classified into two main categories low frequency which is up to 1kHz and high frequency ranging from 10 to 100MHz (Carrie Armel et al., 2013). These properties play an important role on the performance of the disaggregation models and the privacy of the residents. The fundamental limits of a NILM system are explored in depth by Dong et al. (2014). The authors derive an upper bound on the probability of distinguishing scenarios given a NILM system. The upper bound depends on the characteristics of the collected energy data.

The second step of the NILM framework is the appliance feature extraction. The hierarchical taxonomy of these features is depicted in Fig. 3.1.2 and is described in details by Zoha et al. (2012). According to the authors, the group of steady state features require low sampling frequency and consist a lower cost solution. A system that is using transient state features tends to be more expensive because it requires specific hardware. The performance is in general improved when appliances with overlapping steady state features are involved. Taking into account the existing hardware in the market a system using steady state features is more feasible. There is another category of features, called Non-Traditional. Non-Traditional are considered the features that cannot be grouped into the other two traditional categories. These include more complex features such as combining traditional ones, contextual features, behavioural or indicators of the appliances.

Kim et al. (2011) propose a conditional factorial hidden semi-Markov model that utilizes non-traditional features. These features include time of day and day of week, based on the fact that appliance usage has temporal patterns. Wytock and Kolter (2014) suggest a contextual supervised technique with radial basis functions over temperature and hour of day. According to the above mentioned research, temporal features show strong performance boost of energy disaggregation algorithms. An unsupervised approach that takes into account device interactions is proposed by Aiad and Lee (2016). Low power quality issues can lead to disturbance of harmonics and interference currents, due to the operation of other devices. Embedding these interactions in a factorial Hidden Markov Model, shows improved disaggregation accuracy for the devices involved in mutual interactions. Lange and Bergés (2016) utilize a neural decoder to extract binary sub-components. The sub-components are used to infer the power consumption of the appliances.

The third step of the NILM pipeline regards inference and learning. Given the extracted features of the previous step, the model disaggregates the original signal into its components. The technology that is used to model the disaggregator are based on two main approaches: optimization and machine learning. In the literature there are many optimization algorithms tackling the problem of power disaggregation (Baranski and J, 2003; Liang et al., 2010; Baranski and J, 2003). The computational complexity of the problem restricts a practical implementation of optimization approaches to only a small number of appliances. A large number of devices and a complex environment makes the task of signature matching very hard.

Modern approaches are approximate solutions based on machine learning methods. The ideal machine learning based NILM system is an unsupervised algorithm that can identify the number of electrical appliances on its own and thus minimum input is required from the user (Zeifman and Roth, 2011). However, unsupervised learning is very challenging and supervised solutions perform better and are more practical, given that there are enough labeled data. Supervised NILM solutions include methods such as neural networks (Ruzzelli et al., 2010; Srinivasan et al., 2005), support vector machines (SVM) (Kato et al., 2009; yuan Lin et al., 2010), hidden Markov model (HMM) (Zia et al., 2011), Bayesian approaches (Marchiori et al., 2011; Zhong et al., 2015), clustering Hart (1992); Laughman et al. (2003) and combination of various methods (Chang et al., 2011; Lai et al., 2013; Liang et al., 2010).

Most modern approaches have been based on hidden Markov models, until the revolution of deep neural networks. Since 2012 deep neural networks have shown unprecedented performance on many fields spanning computer vision (Krizhevsky and Hinton, 2009), speech recognition (Deng et al., 2013), natural language processing (Collobert and Weston, 2008) and others. Deep learning solutions in the domain of NILM have outperformed HMM based solutions and new neural architectures are yet to be explored.

3.1.1.3 Complexity

Power disaggregation was firstly introduced by Hart (1992), as a combinatorial optimization problem with known number of appliances. It is NP-complete and for large number of appliances it is computationally intractable. The assumption of known appliances a priori is not true in most cases and even if it is known it can change. However, for practical reasons the problem is usually constrained to a certain number of electric loads.

Apart from the number of devices in a house, it is shown that the complexity depends on other factors as well. Egarter et al. (2015b) take the following factors into account:

- The number of devices.
- The switching frequency of energy states of a device.
- The number of operation states.

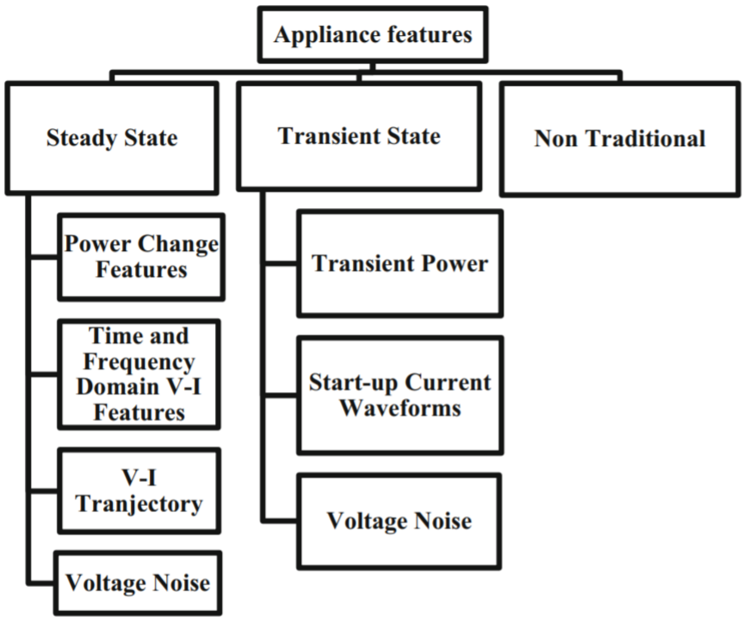


Figure 3.1.2: Hierarchical taxonomy of appliance features.

- How similar energy footprint the devices have.
- Any additional noise or appliances that are not known.

Using the standard computational complexity as a measure to compare different power disaggregation systems is not adequate. Shannon's Entropy is an alternative but it doesn't cover the case where there are many similar energy states among the devices. Kolmogorov's complexity is a powerful mathematical tool, but practically there is no typical method to estimate it. A novel complexity measure is introduced by Egarter et al. (2015a) meeting the following criteria:

- The complexity of the disaggregation problem is measured without dependencies on the algorithm.
- It considers the number of appliances, the number of energy states and the similarities between states and appliances.
- It takes into account the usage of electric devices through time.
- It is simple and comprehensive.
- It is specific to the problem of power disaggregation, enabling a comparison among different solutions.

The new complexity measure is based on the similarity factor of energy states among some electric appliances. The measure is defined according to the overlapping coefficient as follows (Egarter et al., 2015a):

$$C_k = \sum_{j=1}^{M} OVL(f_{P_k}, f_{P_j}), k\epsilon[1, M] \tag{3.2}$$

or

$$C_k = \sum_{j=1}^{M} \int_0^{P_M} \min(f_{P_k}(p), f_{P_j}(p))dp, k\epsilon[1, M] \tag{3.3}$$

where C_k is the disaggregation complexity for power state k, P_k is the reference power value, k is the reference power state and M is the number of combinations of power states. Equation 3.3 refers to a specific power state k against a set of M power state combinations. In a time series the disaggregation complexity will be the average complexity given a predefined time frame. Let a time series set of T power samples, then the complexity C_t for each sample t is computed against all the M different appliance state combinations. Then the average complexity is computed. The equation below describes the disaggregation complexity in a time series:

$$C_k = \frac{1}{T}\sum_{t=1}^{T} C_t = \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{M} OVL(f_{P_t}, f_{P_k}), k\epsilon[1, M] \tag{3.4}$$

where T is the observed time frame, C_t the disaggregation complexity for the power state at time t.

3.1.1.4 System Requirements

Deploying a NILM system in the real world is not trivial, because of the complexity of the problem and unpredictable environmental conditions. In order to keep the quality of such a service high for all the end users, it has to meet certain requirements. Zeifman (2012) describe the criteria that will make a NILM system robust and applicable as a product.

The first requirement, named "feature selection" regards the feature selection and is based on the fact that most smart meters support a sampling rate of 1Hz. This is the first step of the NILM framework and the low frequency offers a low cost solution. The second requirement is called "accuracy" and refers to the performance of the actual disaggregation algorithm. The minimum accuracy for an acceptable user experience should be around 80% - 90%. Furthermore, for a seamless and smooth user experience the installation and the setup of the system should be straightforward. The ideal NILM system should minimize the configuration steps and should automatically identify any new devices. This requirement is known as "no training". The fourth requirement, called "real-time capabilities" concerns the latency of the disaggregation algorithm. The system should be able to provide real-time feedback about the current energy status of the residence. The fifth requirement, named "scalability" is about the scalability of the NILM system with respect to the number of appliances. The performance of the algorithms should be stable in a complex environment where there are more than twenty appliances. Finally, the variability of the electric appliances in the house that the disaggregation algorithm can recognise should include all the four different types. The term "Appliance types" is used and includes on/off, finite-state, variable power and permanent consumer.

The NILM research community uses Zeifman's requirements in order to evaluate NILM systems. However some clarifications have to be made with regards to the terminology. The term "feature selection" from the software perspective, it refers to the process of feature extraction and selection from the aggregated data. From the hardware perspective, the features correspond to the specifications of the equipment that gathers the data. Hardware and software are related though as the hardware specs affect the disaggregation model capabilities. The next term that creates some confusion is the term "accuracy". In Zeifman's requirements "accuracy" is used without specifying a metric and the minimum threshold is an indication. Using the right metric to measure the performance of a NILM system is a subject of research and there is a rich literature proposing various metrics. Finally, "no training" should not be confused with the training of machine learning algorithms. It refers to the configuration of a new installation and the effort that is required from the user. For example a user could select the appliances that will be recognised by the system and that would require to configure the system by switching them on and off and sync with the software of the system.

In addition to Zeifman's requirements, the adoption of machine learning methodologies raises some new concerns. Kelly and Knottenbelt (2015a) emphasize the ability of machine learning based NILM solutions to generalize to data that have not been used for training. Each house in the real world corresponds to a different distribution and a machine learning model should generalize its performance on unseen houses. This thesis evaluates NILM algorithms considering their generalization capabilities as well. For this reason generalization metrics and respective acceptance thresholds should be proposed.

Another aspect of a NILM system which is frequently passed over by NILM researchers is the user's privacy. Indeed, sensitive information could be disclosed via a NILM system and consequently the capabilities and the limitations of a NILM system should be acknowledged a priory. Dong et al. (2014) analyze the limitations of a NILM system and describe what information can be disclosed by an energy disaggregation algorithm. Greveler et al. (2012) prove that power signal from a house can reveal information about the resident's actions inside the house e.g. what TV program they watch. Information leakage without the user's written permission can put the resident's privacy in danger. Privacy should be added as a requirement for NILM systems so that the users are aware of what information a company can extract from the data that they are given. Additionally to privacy, transmission and storage security is also essential and can be achieved by standard protocols and specific encrypted hardware. Privacy is a relative new concern in the sense that in the past years no user would imagine the capabilities of the current technological advancements. Therefore, limiting the capabilities of a NILM system is essential although it could degrade the performance of disaggregation. This can be made more clear by the following example. Assume that a user doesn't have any problem revealing the information when a TV is on or off, but she would like to keep the information about what program she watches as private information.

The aforementioned properties constitute qualitative criteria that can be used to evaluate a NILM system. Unfortunately these qualitative criteria are not sufficient to compare different NILM systems. There is a need to map them to quantitative metrics so that a straightforward comparison is feasible.

3.1.1.5 Performance Evaluation Criteria

A solid benchmark framework for NILM systems is a disputable challenge in the domain. Researchers have been evaluating their solutions on different datasets, with different criteria and using different metrics. Consequently, a direct comparison between different approaches is not possible.

Liang et al. (2010) inquire into the challenge of comparing different NILM systems as follows. The authors define accuracy measures such as detection, disaggregation and overall accuracy. Appliance-based accuracy is deduced. Similarity and complementary ratio are also defined, in order to quantify the divergence between two electrical signatures and the correlation between two solutions, respectively.

The previous assessment tools are not always successfully applied, especially when the event detection step is not part of a NILM solution. Kim et al. (2011) address the problems of using the metric of accuracy for evaluating NILM solutions, attributing the reasons to the fact that an appliance specific event is negligible in a reasonable period of evaluation. For example predicting that an appliance is not operating the given time frame will end up with a high accuracy, although the model might be missing the small operating periods. F1 score is proposed to overcome this problem. It is the harmonic mean of precision and recall. These two measures are redefined by Kim et al. (2011), considering accurate true positive and inaccurate true positive results, depending on whether the predictions exceed a threshold of distance from ground truth.

Another classification metric which is very popular in the domain of pattern recognition is the receiver operating characteristic (ROC) curve. Zeifman and Roth (2011) firstly introduced ROC curve as an evaluation method for power disaggregation algorithms. More experiments have to be conducted though, in order to validate if that metric is helpful for this problem.

According to Bonfigli et al. (2015) the evaluation metrics of NILM systems can be split into two main categories. The first category is based on the comparison between the observed aggregate power signal and the reconstructed signal after disaggregation. These metrics include: root-mean-square error (RMSE), mean average error (MAE) and disaggregation percentage (D). The respective equations are:

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T} \left|\overline{y}_t - \hat{\overline{y}}_t\right|^2} \tag{3.5}$$

$$MAE = \frac{1}{T}\sum_{t=1}^{T} \left|\overline{y}_t - \hat{\overline{y}}_t\right| \tag{3.6}$$

$$D = \frac{\sum_{i=1}^{k} E_i}{E_{tot}} \tag{3.7}$$

where $\overline{y_t}$ is the observed aggregate signal, $\hat{\overline{y}}_t$ is the reconstructed signal after disaggregation, E_{tot} is the total energy of the observed aggregate signal, K is the number of appliances signals and T is the number of samples.

The second category describes how effectively the disaggregated signal signatures are assigned to appliance signatures and include: total energy correctly assigned (TECA), disaggregation error (DE), precision (P), recall (R), accuracy (Acc) and F-measure (f1). They are defined as follows:

$$TECA = 1 - \frac{\sum_{t=1}^{T}\sum_{i=1}^{k}\left|\hat{y}_t^{(i)} - y_t^{(i)}\right|}{2\sum_{t=1}^{T}\overline{y}_t} \tag{3.8}$$

$$DE = \frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{k}\left|\hat{y}_t^{(i)} - y_t^{(i)}\right|^2 \tag{3.9}$$

where $\hat{y}_t^{(i)}$ is the separated appliance signal, $y_t^{(i)}$ is the original appliance signal, $\overline{y}_t$ is the observed aggregate signal, K is the number of appliances signals and T is the number of samples.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.10}$$

$$P = \frac{TP}{TP + FP} \tag{3.11}$$

$$R = \frac{TP}{TP + FN} \tag{3.12}$$

$$F1 = 2\frac{P \cdot R}{P + R} \tag{3.13}$$

where TP is the true positive that the appliances was working, FP is the false positive that the appliance was working, TN is the true negative and FN is the false negative.

The work of Bonfigli et al. (2015) focuses on unsupervised methods, while the same metrics are used for supervised ones. In spite of the large number of evaluation metrics, a fair comparison of state-of-the-art NILM algorithms is still very difficult. New experimental results should include the same metrics that are used by previous experiments otherwise they should replicate older experiments and adjust the metrics that they prefer. In order to facilitate a fair and reproducible comparison of NILM systems Batra et al. (2014a) have designed an open source toolkit named Nonintrusive Load Monitoring Toolkit (NILMTK). The main features of NILMTK are: easy dataset parsing, efficient loading of data to RAM, preprocessing methods, statistical characteristics of the datasets, plotting, a couple of disaggregation algorithms and common accuracy metrics.

Beckel et al. (2014) introduce a new energy dataset named ECO and a novel evaluation framework, named NILM-Eval. ECO dataset provides quality and quantity of both aggregate and appliance specific power data, including real and reactive power. NILM-Eval framework helps running benchmarking experiments and compare different NILM algorithms by tuning multiple parameters. These include a variety of datasets, diversified houses or time periods, particular data features and distinctive algorithm configurations. The metrics, that are used by NILM-Eval, are RMSE, F1 score, precision and recall.

Finally, a different version of F1 score is proposed by Makonin and Popowich (2015), called finite-state-f-score (FS-fscore). FS-fscore is calculated using precision and recall, which are redefined including a partial penalization measure. This is called inaccurate portion of true positives (inacc). The inacc is given by the following equation:

$$inacc = \sum_{t=1}^{T} \frac{\left|\hat{x}_t^{(m)} - x_t^{(m)}\right|}{K^{(m)}} \tag{3.14}$$

where $\hat{x}_t^{(m)}$ is the estimated state from appliance m at time t, $x_t^{(m)}$ is the ground truth state and $K^{(m)}$ is the number of states for appliance m. Precision, recall and FS-fscore are given by:

$$P = \frac{TP - inacc}{TP + FP} \tag{3.15}$$

$$R = \frac{TP - inacc}{TP + FN} \tag{3.16}$$

$$FS - fcore = 2\frac{P \cdot R}{P + R} \tag{3.17}$$

where TP is true-positives, FP is false-positives and FN false-negatives.

3.1.2 Modern Approaches in NILM

Energy consumption data depend on time by nature and NILM researchers have been borrowing ideas from other time series problems. Hidden Markov models were considered state-of-the-art in speech recognition and other domains that include sequential data and thus they have been used in NILM as well. Despite the impressive results and the research efforts to overcome scalability problems, HMMs have been proved to be impractical for NP-hard problems like NILM. HMM based solutions are restricted in relatively small discrete state space, the algorithmic complexity for inference is intractable and the state space can easily grow exponentially. This exponential space complexity is worse, when extending the model in context window (Lipton et al., 2015). In this thesis, the most recent HMM models in NILM are reviewed, presenting the aforementioned problems. Furthermore, the great success of deep neural networks in speech recognition and natural language processing, lead many NILM researchers to build neural networks for the problem of power disaggregation. This section presents the first neural network based approaches for NILM and compares them against state-of-the-art HMM based solutions.

3.1.2.1 Particle Filter-Based Load Disaggregation (PALDi)

Egarter et al. (2015b) proposed a method, named Particle Filter-Based Load Disaggregation (PALDi). The devices that were used belong to Type-I (On/Off), Type-II (Finite State Machines) and combinations of them. Appliances' load signatures and superimposition of them were modeled with HMM and factorial HMM, respectively. Inference was estimated by Particle Filtering (PF). For the evaluation, both synthetic and real data were used and the preferred metrics were: normalized root-mean-square error (RMSE) and accuracy of classification, as presented in Eq 3.10.

Five different scenarios were implemented on synthetic data and three on real data from REDD dataset (Kolter and Johnson, 2011) The three scenarios on real data were three variations of PALDi: with noise adaptation, without noise adaptation and resetting posterior estimation. The number of particles was the same for all three cases, Np = 100. The variety of scenarios enhance the impressive results of total accuracy of 90%. However, the fact that the proposed algorithm was not compared directly with other approaches and was not tested on different databases, might indicate a case of overfitting.

3.1.2.2 FHMM Exploiting Context-Based Features

Paradiso et al. (2016) proposed a new power disaggregation method, which uses Factorial Hidden Markov Models (FHMM) and exploits context-based features. The context information consists of the user presence and the power consumption patterns of appliances. The experiments used real home data from Tracebase dataset, sampled at low frequency. The proposed solution is cost effective and easily applicable, because sampling requirements are already met from existing smart meters. Any extra sensors, to gather the contextual features, are widely available and can have low cost.

Data from Tracebase was preprocessed and the devices that were tested are: coffee maker, LCD TV, microwave oven, pc, refrigerator and washing machine. Each appliance behavior was approximated by a few states and power states were extracted by using clustering analysis. The adopted clustering method was Gaussian Mixture Model (GMM). Next, extensive experiments were run, testing each context-based scenario and comparing the results against the Additive Factorial Approximate MAP (AFAMAP) (Kolter and Jaakkola, 2012). Finally, both types of contextual information were combined and evaluated.

AFAMAP is based on FHMM, has the same scalability issues and is susceptible to local optima. The main advantage of the model is that the result fits very well with the total aggregate signal. Kolter and Jaakkola (2012) combined this algorithm with a model called difference FHMM. The final model is computationally tractable, is robust to noise and overcomes problems of local optima. Paradiso et al. (2016) compared their model against the basic additive model and the results are very encouraging. It would be interesting to see a comparison against the combined model as well.

The basic scenarios of the experiments were: user presence single interval conditioning, user presence double interval conditioning, usage statistics conditioning and a combination of the last two. the metrics that were used for the evaluation are precision and F-Measure. On average the scenario with combined context data showed the best results, whereas each scenario performed better than the basic AFAMAP algorithm. The suggested approach is not directly compared with other solutions different from AFAMAP, but the results prove that context information can be very valuable in a NILM system.

3.1.2.3 FHMM with Device Interactions

Aiad and Lee (2016) suggested an unsupervised disaggregation model, taking into account the interactions between devices. The device interactions were modeled using Factorial Hidden Markov Model and inference was performed by the Viterbi algorithm. The model was tested on the well-known REDD dataset and was compared to the standard FHMM. The results were significantly improved in the case of device interactions, whereas no substantial improvement was observed in the absence of interactions.

According to this methodology, a state of an appliance was described by three values: the initial, average and final power consumption. In order to simulate any pulsations, a state was also represented as a random variable with normal distribution. Only a small fraction of the power consumption was used to model the states and estimate any mutual device interactions. Then, power disaggregation took place using the Viterbi algorithm. Finally, the results were compared to real data. The total energy, which is correctly assigned, was estimated by the metric of accuracy. The data of the experiments come from house 2 in REDD dataset.

A direct comparison was shown against the standard FHMM where device interactions were ignored. The results are encouraging as the proposed solution showed the same or better performance for all devices. However, still the average accuracy, over 7 devices, was less than 70%. These devices include: microwave, outlet 1, outlet 2, refrigerator, washer dryer, lights and stove.

3.1.2.4 Sparse Viterbi Algorithm

Makonin and Popowich (2015) proposed a new algorithm, tackling the efficiency problem of Viterbi algorithm. This approach was based on super-state Hidden Markov Model and a different version of Viterbi algorithm. A super-state was defined as an HMM that describes the overall power state of a set of appliances. Each appliance could be ON or OFF and during operation could have an operation state. Each combination of the devices' states represented a unique state of the house.

The main advantage was that exact inference was feasible in computationally efficient time, by calculating sparse matrices with large number of super-states. Disaggregation could also run in real time, even on an embedded processor with limited capabilities.

The methodology started with separation of data in two categories, priors and testing. For the priors, it was necessary to know the sub-meter readings of the devices that would be disaggregated. Then a Model Builder was used to create a super state HMM. Firstly, each load was mapped to a probability mass function (PMF) by using the priors. Secondly, each PMF was quantized finding each load's states and the peak value of the state. Thirdly, the load's states were combined and constituted the super-state HMM. The super-state and the smart meter data were introduced to the Sparse Viterbi Algorithm, finding the state with the maximum probability. The key of the algorithm was that it ignored zero-probability terms. Finally load consumption was estimated by decoding the quantized state and finding the peak of the respective PMF.

Regarding the data during the experiments, REDD and AMPds (Makonin et al., 2013) were selected, because they support low frequency data. The evaluation criteria were accuracy, FS-fscore and consumption estimation accuracy. For more robust results, tenfold cross-validation process was applied.

It is worth mentioning that this algorithm was directly compared to related work, Kolter and Johnson (2011) and Johnson and Willsky (2013), where factorial Hidden Markov Model (FHMM) and Hierarchical Dirichlet Process Hidden semiMarkov Model (HDP-HSMM) were used respectively. The former one is the standard FHMM and it has been the basis when comparing NILM systems. The latter one is a method which overcomes two restrictions of Hidden Markov Models. The first restriction is that HMMs operate using discrete states and the distributions of state duration consist a graph that doesn't describe adequately real-world data. The second one is that hidden states are set a priori defining model's complexity in a nonBayesian way. HDP-HSMM is based on Hierarchical Dirichlet Process HMM (HDP-HMM) and takes advantage of semi-Markovian properties.

Johnson and Willsky claim that their method is unsupervised. HDP-HSMM is a Bayesian approach and the graphical model encodes prior appliance information. Therefore, there is no need for manual labelling. On the other hand, other researchers (Makonin and Popowich, 2015; Parson et al., 2014) argue that the method cannot be considered unsupervised in the real world because it requires prior knowledge for each appliance. The model must know the number and the type of the appliances in the house. Finally, this Bayesian approach has not been tested in complex environments with more than 15 appliances.

To conclude, both the basic FHMM and the HDP-HSMM are more complex models than the proposed sparse Viterbi algorithm. The comparison showed that HMM sparsity not only reduced complexity, but also presented major improvement in terms of accuracy, 13.3% better than HDP-HMM and 48.3% better than

FHMM on average.

3.1.2.5 The Neural Energy Decoder

Lange and Bergés (2016) examined a method using a neural decoder, in order to extract features from high frequency data. In summary, the aggregated power signal was broken into sub-components in an unsupervised way. Then, they were combined to shape appliance profiles. This was the first attempt of unsupervised feature extraction, by using a deep neural decoder.

The architecture of the deep neural network had a linear output layer and a binary activation function in last layer. The total number of layers was six and the framework that was used is the python package Keras. The inputs of the network were the real and imaginary parts of the Discrete Fourier Transform of the current signal and the target outputs were active and reactive power. The selected optimizer was stochastic gradient descent. The data came from Phase B of the BLUED (Filip et al., 2011) dataset and the network was trained on all data. After training, the last layer of the decoder was removed, and the network could infer the binary subcomponents. Then, the subcomponents were used as features to predict which devices were on or off. For this purpose, two different algorithms were tested: greedy search and logistic regression. A naive energy estimation was also examined. According to the results, logistic regression performed better F1 score than greedy search, with values more than 0.90 for the majority of the devices.

The main goal was to face three basic drawbacks of existing disaggregation solutions: computational efficiency, data transmission limitations and prior knowledge of electrical devices. The proposed algorithm overcomes these pitfalls, but also a direct comparison against other algorithms is needed.

3.1.2.6 Deep Neural Networks Applied to Energy Disaggregation

Kelly and Knottenbelt (2015a) focused on solving the problem of power disaggregation by means of deep neural networks. In this scope, the authors proposed three different approaches: a) a solution using a specific type of recurrent neural network, named “long short-term memory” (LSTM), b) a solution reducing noise with denoising autoencoders and c) a regression algorithm forecasting the start time, end time and average power demand for each device.

The source of the data was UK-DALE dataset (Kelly and Knottenbelt, 2015b). Both real and synthetic energy data were used for training. This approach of training on a mixture of real and synthetic data showed better generalization, when neural nets were tested against unseen data. The step of validation and testing included only real energy data, for more realistic results. NILMTK was used to preprocess data. The algorithms were written in Python using Pandas, Numpy and Lasagne. The target devices included washing machine, kettle, fridge, dish washer and microwave. The metrics that were used are seven: F1 score, precision, recall, accuracy, relative error in total energy, proportion of total energy correctly assigned and mean absolute error.

Regarding training, the first step was to find all the activations of the target device. Then, the algorithm decided randomly to include one of the activations in the timeframe that will be the target of the net during training. Below, the three architectures that were tested, are presented.

The first, of the three neural networks, had six layers in total. An input layer with length which depended on the duration of the device. The second layer was a 1D convolution layer that played the role of extracting features from the signal. The third and fourth layers were bidirectional LSTM. The last two were fully connected with the last being the output of the network. The recurrent neural network was trained with the method of backpropagation through time.

The second neural network was a denoising autoencoder and was used to reduce noise. The network was trained having a noisy signal as input and the clean signal as target, learning in that way to remove noise. The input was the total aggregate power signal and the target was the appliance's load. The architecture of this network consisted of an input layer, a 1D convolutional layer, three fully connected and one convolutional as the output.

The third architecture aimed to predict the start time, the end time and mean power demand of a target device. The network consisted of the input with length depending on the duration of the appliance, two 1D convolutional layers and five fully connected layers. The last one had three outputs, one for each of the target values.

The proposed algorithms were compared with combinatorial optimization and factorial Hidden Markov Model, using implementations of the tool NILMTK (Batra et al., 2014a). In general, denoising autoencoder and regression neural network showed better results than the two algorithms from the benchmark, except for some specific cases. For seen houses, CO and FHMM outperformed Denoising Autoencoder on the metric of relative error in total energy. Regression neural network was outperformed by CO and FHMM only when disaggregating microwave. It is worth mentioning that the two neural networks performed strong generalization capabilities, when data came from a new unseen house. The LSTM network outperformed CO and FHMM only on two-state devices. On more complex devices, such as dish washer and washing machine, LSTM was worse than the other two algorithms.

Further research is recommended with emphasis on optimization of neural networks, understanding LSTMs inferior performance and direct comparison against state of the art algorithms.

3.1.2.7 Deep Recurrent LSTM Network

Mauch and Yang (2015) presented another architecture of deep recurrent LSTM network, in order to test if this type of network is possible to overcome the known problems of previous NILM approaches. Such problems include disaggregation of various appliance types, automatic feature extraction from low frequency data, generalization of a solution to other buildings and unseen devices, extensibility of the approach to continuous time and computational tractability.

According to the suggested methodology, the experiments used synthetic power signals, by summing up sub metered data. The data source was the well-known REDD open dataset. The house, that was used both for training and testing, was house 1, whereas house 2 was used to test the generalization of the algorithm. The target devices included fridge, microwave and dishwasher. The first two appliances were considered as ON/OFF and the third one as multistate. The authors also had the intention to test this algorithm on variable load devices, but unfortunately the selected database doesn't include such devices. Another worth mentioning characteristic of the group of the chosen devices is that the fridge had a periodicity, in contrast to the other two.

The proposed solution needed one network for each device in a house. As a result, in these experiments three networks were used, one for each of the three target appliances. Each network had the following architecture: one input layer with ten units, two bidirectional recurrent layers with 140 units each and one output layer. Each network was trained until the validation error didn't decrease any more or for a maximum number of 100 epochs. The evaluation metrics that were used were estimated energy, consumed energy, NRMS for active periods, precision, recall and F1 score.

The proposed algorithm was not compared directly with other solutions, used only synthetic data and depended on sub metered signals. Nevertheless, the experiments showed encouraging results and lead to the following conclusions: supervised disaggregation was feasible with the proposed architecture, the LSTM architecture worked well for appliances showing some periodicity, the system worked with low frequency data, there was no need for event detection and feature extraction and the trained networks of the fridge and dishwasher could generalize efficiently when tested on house 2.

3.1.2.8 Sequence-to-point Learning with Neural Networks

Zhang et al. (2018a) proposed a deep learning solution for the problem of single-channel blind source separation with application in NILM. The method is called sequence-to-point (seq2point) learning, because it uses a window as input and a single point as target. The proposed solution is a deep convolutional neural network (CNN), which also learns the signature of the appliances in a house.

The seq2point solution differs from a sequence-to-sequence (seq2seq) solution regarding the target output. Instead of predicting a window of appliance power consumption, it predicts the midpoint of that window. The intuition behind the midpoint prediction is that this point is related not only to past values but also to future ones.

The data sources of the experiments come from UK-DALE and REDD. The devices that were tested are: kettle, microwave, fridge, dish washer and washing machine. Kettle wasn't tested on REDD database because it is not included. Regarding UK-DALE, houses 1, 3, 4, and 5 were used for training and house 2 was used for testing. In the case of REDD, houses 2 to 6 were used for training, and house 1 for testing.

The implementation of the model was done in Python, using Tensorflow framework. The architecture of the neural network has an input layer with length 599. Then there are 5 convolution layers with activation function ReLU. Next is a dense layer with 1024 units and activation function ReLU. The final layer is the output of the neural network. There are two versions of this layer, one representing the seq2seq solution and one representing the seq2point solution. For the former case, the layer has 599 units, whereas for the latter one it has only one unit. The activation is Linear for both cases.

Both seq2seq and seq2point versions of the proposed architecture are compared to AFHMM (Kolter and Jaakkola, 2012) and the seq2seq autoencoder architecture (Kelly and Knottenbelt, 2015a). The autoencoder is referred as seq2seq(Kelly), to differentiate it from the proposed seq2seq solution. The evaluation metrics that were used are: mean absolute error (MAE) and normalized signal aggregator error (SAE). MAE is more suitable for the error in power at each time step, whereas SAE is more suitable for the total error in a timeframe. The four models were tested using UK-DALE dataset. Overall the seq2point model gave the best results. It is very impressive that, compared to seq2seq(Kelly), it reduces MAE by 84% and SAE by 92%. Both versions of the CNN achieved much better results than the other two models. Next, only the two versions of the CNN were compared, by using REDD dataset. The results show that the seq2point is superior to seq2seq in most devices with substantial difference. It is worth noting that the tests, where seq2seq had lower error, the difference was negligible (less than 1 watt).

The deep convolution neural network did not only show state of the art performance, but also seems to understand the data. It extracts meaningful features, regardless of the version (seq2seq or seq2point). The authors show systematically that the model learns features that previously were being extracted manually. Such features are: change points, typical usage duration and power levels of appliances. They can be illustrated by plotting the last convolution layer of the proposed neural network.

3.1.3 From Qualitative to Quantitative Comparison

Evaluating and comparing NILM model is not a straightforward task. The challenges to objectively evaluate a power disaggregation algorithm and to find the most suitable for production are manifold. This section analyses these challenges and suggests possible solutions. Firstly, the current qualitative comparative methods are highlighted and next they are mapped into qualitative ones.

3.1.3.1 Qualitative Analysis

The ten different disaggregation models that were described in previous section are evaluated and compared. For that the NILM system requirements, that were presented in previous section, are considered. These include the Zeifman's requirements as well as "generalization" and "privacy". Generalization has been one of the most important aspects in artificial intelligence. Yet it is open problem in machine learning research as a mathematical theory of how overparameterized models generalize is missing (Zhang et al., 2021b). In NILM a generalization metric is mandatory for an objective qualitative measure but is so far missing and will be defined in the next section. In this qualitative comparison the qualitative metric "accuracy" will be used for both seen and unseen houses. Privacy regards sensitive data and information that the user doesn't want to disclose, but can be extracted by a NILM system. A summary of the criteria that the models under comparison meet, is presented in table 3.1.1.

Unfortunately NILM researchers have been ignoring privacy. Therefore, no system has been investigated in terms of "privacy" and a conclusion cannot be extracted. As far as "generalization" is concerned, HMM based solutions have not been tested extensively on unseen houses. Deep neural networks are known for their capabilities to generalize and seem to be perform better. In terms of "scalability" deep learning solutions again outperform HMM ones. Intractability is one of the biggest problems with HMMs and only FHMM exploiting context-based features overcomes it. "Feature selection" and "real-time" capabilities are the requirements the majority of the suggested systems meet successfully. Regarding "accuracy" the conclusion is that no solution meets the lower threshold of 80%. The criteria "no training" and "appliance types" are very difficult to be met. Despite the researchers' efforts to develop unsupervised NILM models, there is plenty room for improvement. Furthermore, there are not enough experiments that utilize all the appliance types, specifically missing variable power appliances. Most of the devices under examination are on/off and multistate ones.

Particle Filter-Based Load Disaggregation (PALDi) Given the fact that a direct numerical comparison against other NILM systems is very complicated, the authors have evaluated PALDi algorithm against Zeifman's requirements. Three of them are fully met. The active power is collected with 1 sec, resolution reported accuracy is around 90% and operational running time is computationally efficient. On the other hand, "no training" requirement is partially met. Although there is no need for training during operation, the algorithm requires to know the used devices in the house and cannot recognize new or unseen devices. The other two requirements, "scalability" and "various appliance types", are not met. Regarding the former, the algorithm's complexity depends on two parameters, the number of particles and the number of appliances. HMM models are, in general, computationally impractical for more than 18 devices and according to the tests, particle filtering shows proportionally better results in the case of more particles. Consequently, PALDi is not scalable. For the appliance types, there is no evidence that the algorithm will present equal results for types such as variable power and permanent consumer devices. Finally, no conclusion can be made for "generalization" and "privacy".

FHMM exploiting context-based features This NILM system fulfills four requirements. "Feature selection", because active power is the basic feature sampled with low frequency. "No training", because it identifies the power profile states of a specific appliance in an unsupervised way, using Gaussian Mixture Model. "Real-time capabilities", because inference can be done in real-time. "Scalability", because it is based on the work of Kolter and Jaakkola (2012), where the proposed method scales almost linearly in the space of HMM states. On the contrary, there is no reported accuracy, thus no conclusion can be made. Only precision and F-Measure results are compared against a basic AFAMAP algorithm with 13 and 14% improvement respectively. Additionally, the proposed method was mainly tested using finite state machines such as coffee maker, LCD TV, microwave oven, refrigerator and washing machine. Therefore, the variety

	Feature selection	Accuracy	No training	Real-time	Scalability	Appliance types	Generalization	Privacy
Egarter et al. (2015b)	✓	✓	✗	✓	✗	-	-	-
Paradiso et al. (2016)	✓	-	✓	✓	✓	-	-	-
Aiad and Lee (2016)	✓	✗	✓	✓	✗	✗	-	-
Makonin and Popowich (2015)	✓	✓	✗	✓	✗	-	-	-
Lange and Bergés (2016)	✗	-	✗	-	✓	-	-	-
Kelly and Knottenbelt (2015a) (AE)	✓	✓	✗	✓	✓	-	✓	-
Kelly and Knottenbelt (2015a) (Conv)	✓	✓	✗	✓	✓	-	✓	-
Kelly and Knottenbelt (2015a) (LSTM)	✓	✗	✗	✓	✓	-	✗	-
Mauch and Yang (2015)	✓	-	✗	✓	✓	-	✓	-
Zhang et al. (2018a)	✓	✓	✗	-	✓	-	✓	-

Table 3.1.1: NILM models with the qualitative requirements they meet. (✓) met, (✗) not met, (-) no conclusion

of devices used in the test is not sufficient to meet this requirement and further experiments need to be run. Concerning “generalization” and “privacy” no conclusion can be drawn.

FHMM with device interactions The proposed approach meets successfully three of the qualitative criteria. First of all, it uses low frequency data. Secondly, it is unsupervised, although manual labeling is required at the end of the process. Thirdly, inference can be done in real-time. The requirements that are not met are “accuracy”, “scalability” and “appliance types”. Regarding “generalization” and “privacy”, unfortunately no conclusion can be made.

Sparse Viterbi algorithm The authors of this paper state that the proposed algorithm fulfils four requirements. The algorithm was successfully tested on low frequency data, average accuracy exceeds the minimum threshold of 80% and inference can be completed in under one millisecond. The fourth requirement is “various appliances types”. According to the authors, it is met considering a more relaxed version of the requirement. In this case and for the purpose of approximate estimation, continuously variable devices can be represented by one more general state. However, someone could argue that mainly finite state machine devices were used during evaluation and eventually the algorithm doesn’t fulfill this requirement. Additionally, there is no proof that all variable state devices can be represented by one general state. More experiments with more appliances could possibly solve this ambiguity. For the criteria of “no training” and “scalability”, they are not met. The requirement of “no training” is not met because of the process of selecting priors. Regarding “scalability”, the system demonstrated the ability to disaggregate efficiently up to 18 loads. This is a notable improvement, because previous methods were restricted to 6–9 loads. However, the problem of exponential time and space complexity remains unsolvable, when the target devices are more than 18. As far as “generalization” and “privacy” are concerned, the system cannot be evaluated, because of lack of further information and experiments.

The neural energy decoder This system meets only one requirement, “scalability”. There are two reasons explaining why the system is scalable. Firstly, after the neural decoder is trained, inference is computationally very cheap. Secondly, high frequency data will be transformed in binary subcomponents, which require much less space. This transformation of high frequency data makes it possible to exploit more information, but at the same time fails the requirement of “feature selection”. The other requirement that is not met is “no training”. Although the binary subcomponents can be extracted in an unsupervised way, the inference process is supervised. The system will need extra training to identify unseen devices. Regarding “accuracy” no conclusion can be made, because the metric of disaggregation performance is F1 score. In the same way, there is no evaluation outcome concerning “near real-time capabilities”. The characteristics of the system are promising, because once the decoder is trained, both feature extraction and inference can be done very efficiently. Nevertheless, the system is tested off line and the need for temporal randomization of the dataset can be an obstacle during online testing. Finally, there are not enough experiments or information to evaluate the algorithm against “various appliance types”, “generalization” and “privacy”.

Deep neural networks applied to energy disaggregation In this paper, there are three different architectures of deep neural networks: denoising autoencoder, deep convolutional neural network for regression and RNN LSTM with convolution input layer. The first two architectures, according to the experimental results, show similar behavior. They fulfill five of the eight qualitative system requirements. On the other hand, the third architecture doesn’t perform equally and meets only three requirements. In more details, the “feature requirement” is met by all three architectures, due to the low frequency data. According to the results, all but LSTM based architectures show average accuracy above 80%, outperforming the basic algorithms of CO and FHMM. Training is supervised for all the suggested networks and requires knowledge of the device signature profile. Therefore, they all fail to meet the requirement of “no training”. In contrast, they are all considered capable of performing in real time, because after training, inference is computationally efficient. For the same reason, neural networks are suitable for scalability. Regarding "generalization" the denoising autoencoder and the deep convolutional network have shown impressive results. Finally, there is no conclusion about "privacy" and "appliance types". The devices were considered to have only two states ON or OFF. Consequently, further research is needed to examine if the proposed methods are suitable for other appliance types.

Deep recurrent LSTM network The proposed recurrent neural network meets four requirements: "feature selection", "real-time capabilities", "scalability" and "generalization". Indeed, the experiments used low frequency data, neural networks, theoretically, can perform inference computationally efficient and the results showed satisfactory generalization. Regarding "no training", it is supervised and requires knowledge of the devices in the house. Unfortunately, no conclusion can be made for "accuracy". It is not used as a metric, but the other metrics showed encouraging results. Similarly, there is no conclusion for "various appliance types" and "privacy". Only two device types are tested and more complicated experiments are required.

Deep convolution neural network (seq2point) Two versions of this architecture are presented in the paper. The seq2point version will be analyzed here because the results are by far the best. Also since the basic architecture is the same, equivalent results should be expected for the seq2seq version. The requirements that are clearly met are: "feature selection", "scalability" and "generalization". The reasons are that the selected databases have low frequency data, inference is computationally efficient and the model was successfully tested on unseen houses. "Accuracy" is not clear if it is met because of the different metrics that are used. This is also a deficit of the system requirements, where minimum acceptance values of other metrics should also be considered. However, the authors directly compare their proposed solution to Kelly's autoencoder. The results are impressive with a reduction of 84% in MAE and 92% in SAE. The model achieved better results than the other systems, for all devices. Considering the impressive results and their consistency, leads to the outcome that accuracy criterion should be met. Regarding the "real-time capabilities", there is an argument of how much of the future data is used as input, when there is seq2point learning. Considering that there are 599 inputs, the future data are 299. Also, the readings are recorded every 1–6 seconds. As a result, inference takes place with a lag of 299–1794 s or 5–30min. A real-time system should respond to any changes in a few seconds. On the other hand, for the case of seq2seq the predictions are clearly real-time. Consequently, deciding if the seq2point can give real-time results, depends on the sampling rate and how much of the future data is needed. To answer such questions more experiments must be done with exact specification of real-time requirements e.g. how many seconds of lag is accepted for a NILM system. The requirement of "no training" is not met, as the proposed solution is supervised. Finally regarding "privacy" and "appliance types" no direct conclusion can be made.

3.1.3.2 Quantitative Analysis

Table 3.1.1 shows the results from the suggested algorithms, the databases that were used and the number of target devices. For comparison reasons, the table presents the three most common performance metrics. These are: Acc 3.10, F1 score 3.13 and TECA 3.8. To avoid any confusion it is worth mentioning that in bibliography TECA is also called Acc (Aiad and Lee, 2016; Kolter and Johnson, 2011) or Est Acc (Estimated Accuracy) (Makonin and Popowich, 2015). For each metric the mean and the standard deviation are calculated. The calculations are based on results taken from the respective papers. For PALDi system the Table VII was used from the respective paper of Egarter et al. (2015a), for FHMM with device interactions Table 5 was taken from the paper of Aiad and Lee (2016), for Sparse Viterbi algorithm Table V of Makonin and Popowich (2015),for The Neural Energy Decoder Table I from Lange and Bergés (2016), for the systems described by Kelly and Knottenbelt (2015a) Figure 3 and Figure 4, and finally for RNN LSTM the source of the data were Table 1 and Table 2 from the paper of Mauch and Yang (2015). The results from the seq2point CNN are not presented in Table 2 because different metrics are used. Moreover, Zhang et al. (2018a) presented a direct comparison to other models as discussed later.

At the same table, appliance set complexity is also presented, to make comparison more objective. The calculation of complexity is based on equation 3.3 and is implemented in Python, using the NILMTK for parsing the databases [1]. The complexity of the appliance set, that Neural Energy Decoder used, is not calculated, because BLUED dataset is not integrated with NILMTK toolkit.

As it is noticed, the complexity of each NILM environment differs a lot and makes a direct comparison very difficult. For example, the RNN LSTM system described by Mauch and Yang (2015), has been tested on an environment with maximum complexity 1.84 and mean complexity 1.2 for house 1. The mean f1 score is 0.78. On the other hand, the Deep Convolutional Neural Network is tested on environments with mean and maximum complexities around 4.7 and 10, whereas the mean f1 score is around 0.55. Based on these numbers, it is obvious that a direct comparison is unfair. Even if two systems are compared by using results from the same house e.g. Redd House 1, the complexity can be different, because each experiment might use different appliance sets. The more the appliances and the states of each appliance, the higher the disaggregation complexity.

The appliance set complexity doesn't depend on the disaggregation algorithm. It characterizes how difficult it is, to solve the problem of disaggregation for a given appliance set. This means that if the results were based on datasets with similar complexity, a comparison among all the approaches would be easier. As an example, let's compare the results of the PALDi system and the Deep Convolution Neural Network. According to the metric of Acc the latter one seems to outperform the former. Both have similar mean complexity, which is not a surprise, because they both recognize the activity of the same number of devices.

Another difficulty that can emerge from comparing these NILM systems is that they are using different metrics. To set an example, regarding the three common metrics Acc, TECA and f1 score, PALDi uses only Acc, whereas The Neural Energy Decoder uses only f1 score. A different metric could also be the result of how each solution is solving the problem of power disaggregation. TECA, for example, is used when a system predicts the power consumption of a device, whereas Acc and f1 score are used to decide if an appliance is ON or OFF or to classify its state. In addition, researchers do not always disclose all the details of the experimental results. As it can be noticed from Table 2, some useful information is missed. The standard deviation, for example, could give an overview of the performance of a system for the various devices.

To conclude, a quantitative analysis of various NILM systems is not feasible. The variety of methodologies, datasets, metrics and ways of presenting results make this procedure difficult.

3.1.3.3 Mapping quality requirements to quantity metrics

Power disaggregation is an old and yet unsolved problem. Comparing different NILM systems is essential to solve it, but still not possible. Each of the qualitative and quantitative evaluations, has proved to be inadequate to distinguish the best system. This means that maybe there should be a connection between the two methodologies

Table 3.1.3 shows that researchers' efforts focused on measuring the performance of algorithms in terms of disaggregation accuracy. Now, there are several metrics, making a comparison even more confusing. On the other hand, there is plenty of room for improvement on how to quantify the rest of the qualitative requirements. On these grounds, some simple evaluation measurements are proposed, while further studies

[1]The authors would like to thank Odysseas Krystalakos for his contribution in developing the software and running the experiments for the appliance set complexity. The source code is available at: https://github.com/christofernal/power-disaggregation-complexity.

Systems	Datasource	Complexity	Avg & std of Acc	TECA	Avg & std of F1	N
Egarter et al. (2015b) Noise-adapt	REDD House 1	M = 23.34 m = 4.39	m = 0.73 std = 0.24	-	-	6
Egarter et al. (2015b) No noise-adapt	REDD House 1	M = 23.34 m = 4.39	m = 0.72 std = 0.24	-	-	6
Egarter et al. (2015b) Resetting	REDD House 1	M = 23.34 m = 4.39	m = 0.90 std = 0.10	-	-	6
Paradiso et al. (2016)	Tracebase	-	M = 10.83 m = 6.17	-	Not detailed results	6
Aiad and Lee (2016)	REDD house 2	M = 32.37 m = 19.46	-	m = 0.68 std = 0.11	-	7
Makonin	REDD house 1	M = 6.09 m = 2.83	-	m = 0.95	-	5
and	REDD house 2	M = 4.04 m = 1.97	-	m = 0.94	-	5
Popowich	REDD house 3	M = 5.49 m = 2.10	-	m = 0.90	-	5
(2015)	REDD house 6	M = 1.90 m = 1.19	-	m = 0.98	-	5
	REDD house all houses	-	-	avg = 0.94	-	5
Lange and Bergés (2016)	BLUED phase B	-	-	-	m = 0.92 std = 0.08	13
Kelly and	UK-DALE seen house	M = 10.10 m = 4.75	m = 0.90 std = 0.08	-	m = 0.55 std = 0.18	6
Knotten-	UK-DALE unseen house	M = 10.10 m = 4.75	m = 0.92 std = 0.06	-	m = 0.52 std = 0.32	6
belt Kelly and	UK-DALE seen house	M = 10.10 m = 4.75	m = 0.94 std = 0.07	-	m = 0.58 std = 0.15	6
(2015a) Knotten-	UK-DALE unseen house	M = 10.10 m = 4.75	m = 0.92 std = 0.06	-	m = 0.54 std = 0.25	6
(AE) belt Kelly and	UK-DALE seen house	M = 10.10 m = 4.75	m = 0.68 std = 0.29	-	m = 0.39 std = 0.28	6
(2015a) Knotten-	UK-DALE unseen house	M = 10.10 m = 4.75	m = 0.66 std = 0.33	-	m = 0.38 std = 0.37	6
(Rect) belt Mauch	REDD house 1	M = 1.84 m = 1.20	-	-	m = 0.78 std = 0.12	3
(2015a) and Yang	REDD house 2	M = 1.84 m = 1.20	-	-	m = 0.56 std = 0.43	3
(LSTM) (2015)						

Table 3.1.2: Numerical results, where: N = number of target devices, M = max, m = mean, avg = average, std = standard deviation

Qualitative requirements	Quantitative metrics
Feature selection	Sampling rate
Accuracy	Root-mean-square error (RMSE), mean average error (MAE), disaggregation percentage (D), total energy correctly assigned (TECA), disaggregation error (DE), precision (P), recall (R), accuracy (Acc), F-measure (f1), finite state-f-score (FS-fscore)
No training	Three-state variable for no training (NTR)
Real-time capabilities	Algorithm computational complexity
Scalability	Algorithm computational complexity
Appliance types	Four appliance types standard deviation (FAT_σ)
Generalization	Generalization over unseen houses ($GoUH_\sigma$)
Privacy	Upper bound on the probability of distinguishing scenarios (UBPDS)

Table 3.1.3: Mapping of qualitative requirements and quantitative metrics

and experiments are encouraged

A new metric called NTR (no training) is proposed for the requirement of "no training", which ensures the minimum user involvement. Ideally, the algorithm should be able to identify any new devices with no further user interaction. In practice, there are three distinct categories regarding user interaction, thus NTR values can be the following: a) no user interaction (NUI) where the system can recognize unseen devices, b) light user interaction (LUI), in which case the system doesn't need training, but knowing the used devices in the house is necessary, c) heavy user interaction (HUI), when the system requires training for each new device and consequently the user has to reconfigure the system. From another perspective, the three states also represent how much the algorithm knows about its environment: unknown environment for NUI, partial known environment for LUI and known environment for HUI.

$$NTR = \begin{cases} NUI & \text{no user interaction} \\ LUI & \text{light user interaction} \\ HUI & \text{high user interaction} \end{cases} \tag{3.18}$$

The second metric which is proposed, measures the performance of power disaggregation on four different appliance types: on/off, finite-state, variable power and permanent consumer. A naive approach, to quantify this requirement, is to count how many of the four types can be recognized e.g. 1/4, 3/4 etc. A more sophisticated approach is to calculate the standard deviation of the accuracies of each of the four appliance types. The new metric is called FATσ (four appliance types standard deviation) and is described by the following formula:

$$FAT_\sigma = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(Acc_i - FAT_\mu)^2} \tag{3.19}$$

where $FATm = \frac{1}{4}\sum_{i=1}^{4} Acc_i$ and Acc_i the accuracy of each appliance type given by equation 3.10. Instead of Acc other metrics of "accuracy" can be used such as F1 score.

As far as the requirement of generalization is concerned, a metric similar to FATσ is proposed, representing how well the algorithm generalizes on unseen houses. It is defined as the standard deviation of the total disaggregation accuracy of the system for various houses. The smaller the value, the better the algorithm can generalize. The following formula defines the metric of GoUH (generalization over unseen houses):

$$GoUH = \sqrt{\frac{1}{H}\sum_{h=1}^{H} Acc_h - \mu)^2} \tag{3.20}$$

where H is the number of different houses, Acc_h is the average disaggregation accuracy on house h, and m is the mean accuracy over all houses. Instead of Acc as defined in equation 3.10, other metrics such as F1 score are also accepted. Based on Ziefman's requirement for a minimum accuracy of 80%, a maximum threshold could also be defined regarding GoUH with value $GoUHmax = Acc0.8$.

Finally, it is equally important to quantify privacy. Ignoring the limitations of power disaggregation, could lead to leakage of private information without user's agreement. In order to protect privacy of occupants, Sankar et al. (2013) propose a theoretical framework. Another approach would be to use a distance metric (Katos et al., 2011) finding the correlation of scenarios, given the features and power data that will be used by the algorithm. A deep analysis of private prediction problems, mainly based on hidden Markov models, is presented in related work by (Polat et al., 2010). Privacy is a multilateral problem, requiring knowledge of laws and definition of policies. Theoretical models, will only be useful after setting clear rules and policies on what violates an individual's privacy. Next an example of a metric which probably best fits NILM privacy is presented and is based on an upper bound on the probability of distinguishing private scenarios (Dong et al., 2014). Assume a NILM system which can identify N scenarios. Also assume that m out of N of these scenarios are considered private information. According to Dong et al. (2014), there is an upper bound probability of distinguishing successfully m private scenarios. The lower this upper bound is, the better privacy this NILM system provides. This metric can be calculated as follows (Dong et al., 2014):

$$UBPDS = \sum_{i=1}^{m} P(\hat{u}_{MAP}(y) = u_i | u = v_i) p(u = v_i) \tag{3.21}$$

where MAP is given by:

$$\hat{u}_{MAP}(y) = argmax_{v \in V} P(G(u, .) = y | u = v) p(u = v) \tag{3.22}$$

and V is a finite set of inputs representing scenarios, $\hat{u}$ is an estimator, G is the distribution of the power consumption, u is the input representing the scenario that will be distinguished, y is the observed signal.

3.2 Audio Classification Applications

3.2.1 Introduction

The Internet of Audio Things (IoAuT) is an emerging sub-field of the Internet of Things (IoT) and has attracted many researchers from different disciplines. It lies in the intersection of Internet of Things, sound recognition, machine learning and human-computer interaction (Turchet et al., 2020).

From the humans point of view, speech is the intrinsic way of communication. Yet, most machine-to-human user interfaces have been restricted to very limited voice interactions or other methods such as touchscreens. IoAuT addresses the interdisciplinary challenges that need to be solved in order to pave the way for new opportunities and applications. Turchet et al. (2020) present a taxonomy of these challenges based on the following classes: connectivity, interoperability and standardization, machine analysis of audio content, data collection and representation of audio content, edge computing, synchronization, privacy and security and Audio Things design. In the context of machine analysis of audio content and edge computing, the main challenges we face are large volumes of data, presence of noise, limitations of computing resources and latency during inference.

IoAuT will be generating a huge amount of audio data which will be hard to clean and annotate. Learning from noisy data is a common problem for many machine learning tasks. Even when a model is trained with noisy data, there is no guarantee it will be robust when deployed in the real-world (Martin-Morato et al., 2018; Bharitkar, 2019; Esmaeilpour et al., 2020). Furthermore, IoT devices have limited storage, which makes the deployment of a state-of-the-art neural network prohibitive. Modern deep neural networks are usually trained and tested using very powerful GPUs and thus they are not suitable to run on embedded devices with restricted computational and energy resources. Finally, in order to make an IoAuT application to meet the requirements of the end users, it has to perform under low bandwidth connectivity and in real-time.

The goal of this research is to develop efficient deep learning solutions that are robust to nuances in audio classification applications. The tasks for evaluation of the proposed models are speaker recognition and speech commands identification. The former one regards the recognition of five different speakers using the popular dataset "Speaker Recognition Dataset Prominent Leaders Speeches". The latter one aims to identify ten voice commands from the popular dataset "Google Speech Commands" that is provided by Google.

3.2.2 Speech Commands Classification

Recently, deep learning approaches have demonstrated superior performance than classic machine learning systems in various audio classification tasks. Until now there has been a race on achieving state-of-the-art performance in terms of accuracy on specific tasks. This lead to the development of huge neural networks with millions or billions of parameters that are prohibitive for resource-constrained and real-time systems. In this context, researchers now put their effort on improving the efficiency of deep neural networks.

Recent interest in deploying speech recognition models on the edge has lead to new work on ASR model compression McGraw et al. (2016) and other sound recognition tasks. Coucke et al. Coucke et al. (2019) developed a model utilizing dilated convolution layers, allowing to train deeper neural networks that fit in embedded devices. It is worth noting that the dataset that they created, named "Hey Snips" is public with utterances recorded by over 2.2K speakers. Kusupati et al. Kusupati et al. (2018) proposed a novel recurrent neural network (RNN) architecture named FastGRNN, which includes low-rank, sparse and quantized matrices. This architecture results in accurate models that can be up to 35x smaller than state-of-the-art RNNs. FastGRNN was tested on a variety of datasets and tasks including speech, images and text. The scope of this research was to build models that can be deployed to IoT devices efficiently. Zeng et al. Zeng and Xiao (2019) proposed a neural network architecture called DenseNet-BiLSTM for the task of keyword spotting (KWS) using the Google Speech Command dataset. Their main contribution was the combination of a new version of DenseNet named DenseNet-Speech and BiLSTM. The former component of the architecture captures local features whereas maintaining speech time series information. The latter one learns time series features. Solovyev et al. Solovyev et al. (2020) used different representations of sound such as Wave frames, Spectrograms, Mel-Spectograms and MFCCs and designed several neural network architectures based on convolutional layers. Two of their best performing networks had very similar architecture with VGG Simonyan and Zisserman (2014) and ResNet He et al. (2016). The models were evaluated on the Google Speech Command dataset, showing very strong results with accuracy over 90%.

3.2.3 Robust Speaker Recognition

The goal of speaker recognition is to identify speakers based on their vocal characteristics (Poddar et al., 2018). This problem is also known as automatic speaker verification (ASV). Deep learning methods have considerably increased ASV performance in recent years. Variani et al. (2014) suggested the d-vector using multiple fully-connected neural network layers. Snyder et al. (2018) proposed X-vectors based on the Time-delayed neural networks (TDNN). Yet, recognizing speakers under noisy acoustic conditions remains a challenging task.

Prior studies (Leglaive et al., 2019; Horaud, 2020) used to preprocess the audio data and remove noise, in order to restore their original state. Some other algorithms (Jang et al., 2017; Farahani et al., 2006) try to extract features from uncorrupted speech, while others (Nahma et al., 2019; Yao et al., 2016) try to evaluate speech quality by calculating signal-to-noise ratio (SNR). ASV studies have employed speech enhancement, however this has often been done on an individual basis. This could lead to the learned features or augmented speech signals not being able to meet the requirements for speaker recognition and verification. As a result, it is imperative that both speech enhancement and the speaker processing model be optimized simultaneously. Shon et al. (2019) combined speech enhancement and speaker processing into one framework. To remove noise-corrupted characteristics, the speech enhancement module generated a ratio mask and multiplied it element-wise by the original spectrogram to verify the speaker. However, during training the voice enhancement network, the speaker verification module was pretrained and locked. As a result, the optimization of the two modules was not done simultaneously.

The employment of attention processes as well as collaborative optimization has become increasingly common for speaker identification and verification (rahman Chowdhury et al., 2018; Zhu et al., 2018; India et al., 2019; An et al., 2019). This is due to the fact that distinct input features might be assigned different weights by a neuronal attention process. As a result, the relevant information may be highlighted, and the noise produced by irrelevant information can be reduced.

With the current ASV systems, speech biometrics (Shetty, 2015; Loshin, 2016) are being employed in various areas such as banking and surveillance (Kinnunen and Li, 2010; Campbell, 1997). Modern ASV systems have good performance with lengthy enough voice data. However, because of the high volume of speech needed for training and assessment, as well as the wide range of intersession variability, its wider practical application has been curtailed. The amount of speech data that can be processed by a speaker recognition system in the real world is limited. Assuming that appropriate speech data can be obtained for training purposes, it may not always be possible to obtain the same for verification. Data required for enrollment is unlikely to be obtained even when speaker recognition systems are being used for forensic purposes (Mandasari et al., 2011). The typical speech length in access control instances is only a few seconds (Larcher et al., 2014). As a result, research efforts are needed to provide consistent ASV performance under conditions of short duration.

The quest to create an ASV system that can be used in the actual world has progressed significantly over time. An extensive review on this progress can be found in (Jayanna and Prasanna, 2009), where technical issues as well as the problem of limited duration are discussed. Modern ASV research approaches are best suited for large-scale training and testing scenarios involving a lot of speech. It is possible that ASV approaches, which are best suited for a huge amount of data, are not acceptable for lesser amounts of speech. The constraints of brief utterances should be taken into account while conducting future research.

3.3 AI Tasks for Electronic Health Records

Increasing volumes of data are being generated in the medical industry these days. It is possible to derive useful information about a patient's health by monitoring a variety of physiological signals emitted by various organs. A complex endeavor, the study of physiological signals necessitates the employment of specialized methods like machine learning. This procedure has a number of ramifications and issues, especially when applied on data acquired from medical examinations of patients, which is predominantly a time-series (Aljawarneh et al., 2019).

In the realm of precision healthcare, precise and personalized predictions, preventions, and interventions are all aimed towards improving patient care. There has been considerable progress toward personalized predictions in cardiovascular medicine, radiology, neurology, dermatology, and ophthalmology, just to name a few, thanks to recent advancements in deep learning. For example, Ardila et al. (2019) show that DL can accurately predict the risk of lung cancer from a patient's CT scans. Poplin et al. (2018) demonstrate that DL can accurately predict a variety of cardiovascular risk factors from a single retinal fundus photograph. Along with advancements in deep learning algorithms, a significant contributor to this success has been the huge increase in the number of multimodal biomedical data, including but not limited to mega cohorts such as the UK Biobank (Sudlow et al., 2015) and routinely collected health data such as electronic health records (EHR) (Shickel et al., 2017).

The adoption of EHR systems has accelerated in recent years; the percentage of hospitals in the United States and the United Kingdom that have implemented EHR systems now surpasses 84 percent and 94 percent, respectively (Parasrampuria and Henry, 2019). As a result, the EHR systems of a large medical organization are now likely to collect data from millions of individuals over the course of several years. Each individual's EHR can integrate data from a variety of sources and hence comprise "concepts" such as diagnoses, interventions, lab tests, and clinical narratives. Each instance of a concept may represent a single or numerous data points; for example, a single hospitalization may create thousands of data points for an individual, whereas a diagnosis may represent a single data point. This makes large-scale EHRs an unmatched source of information and data for training data-hungry machine learning models (Li et al., 2020b).

The traditional EHR research pipeline represents individuals as multidimensional vectors, often called features. This method depends heavily on domain expertise of humans. Recent advances in deep learning, have developed models that can learn representations from raw or minimally processed data, with little or no expert input. This is accomplished via a hierarchical structure of layers, each of which employs a large number of simple linear and nonlinear operations to map their corresponding inputs to a representation. The progression from layer to layer is expected to result in a final representation that is useful for a given task. Liang et al. (2014) demonstrated that deep neural networks outperform SVM and decision trees mixed with manual feature engineering on a variety of prediction tasks and datasets. When predicting the risk of suicide from individuals' EHR, Tran et al. (2015) advocated the use of restricted Boltzmann machines (RBM) for learning a distributed representation of EHR, which was proven to outperform human feature extraction. These early research works on the application of DL to EHR failed to account for the nuances of EHR data. In an attempt to address this, Nguyen et al. (2017) developed Deepr, a convolutional neural network (CNN) model for predicting the likelihood of readmission. The authors viewed one's medical history as a series of concepts and added a specific word between each set of successive visits to indicate the time difference. Choi et al. (2016b) proposed a shallow recurrent neural network (RNN) model to forecast the diagnoses and medications that are likely to be encountered in the next visit. Both approaches used some form of embedding to convert non-numerical medical notions to an algebraic space in which the sequence models could operate.

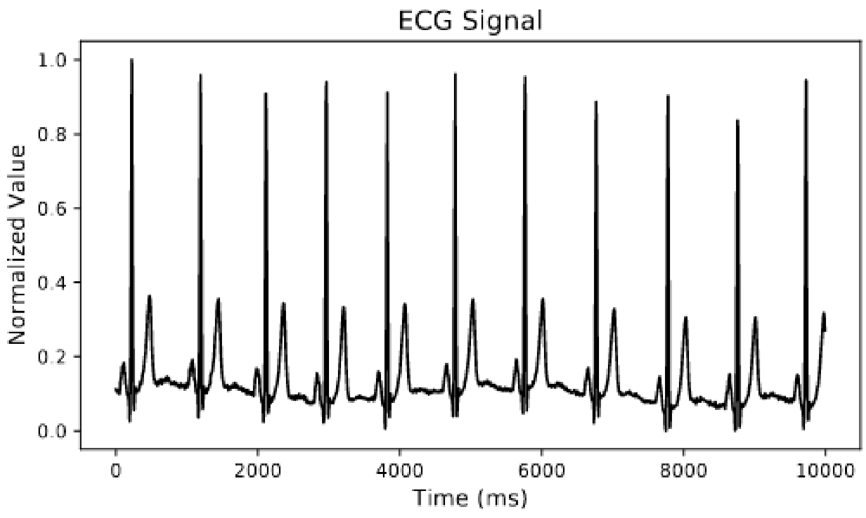


Figure 3.3.1: An example of an ECG signal.

One of the next enhancements made to EHR aimed at capturing long-term dependencies between events. A few representative examples include key diagnoses such as diabetes that can remain a risk factor throughout a person's life, even decades after their first occurrence. Pham et al. (2016) developed DeepCare, a Long Short-Term Memory (LSTM) architecture with an attention mechanism that outperformed traditional ML approaches, LSTM, and plain RNN in applications such as diabetes prediction. Choi et al. (2016a) established a model based on reverse-time attention mechanism to consolidate prior impactful visits using an end-to-end RNN model, named RETAIN, for the prediction of heart failure. RETAIN beat the majority of models at the time of its publication and served as a good starting point for medical deep learning research (Ayala Solares et al., 2020).

3.3.1 Analysis of ECG Signals

Cardiologists and medical practitioners routinely utilize electrocardiograms (ECGs) to assess cardiac health. The fundamental issue with manual analysis of ECG signals, as with many other time-series data, is the difficulty in recognizing and categorizing various waveforms and morphologies in the signal. An example of an ECG signal is shown in figure 3.3.1. For a human, this task takes a long time and is prone to errors. It should be noted that effective diagnosis of cardiovascular disorders is critical, since they constitute the cause of death for approximately one-third of all deaths worldwide (Kachuee et al., 2018). Millions of people, for example, suffer from irregular heartbeats, which can be fatal in some situations. As a result, precise and low-cost arrhythmic heartbeat diagnosis is extremely desirable.

To overcome the issues posed by manual analysis of ECG signals, numerous studies in the literature investigated the use of machine learning approaches to reliably detect anomalies in the signal (Esmaili et al., 2017; Dastjerdi et al., 2017). The majority of these approaches include a preprocessing phase to prepare the signal. Then, handcrafted features are derived from these signals, which are typically statistical summarizations of signal windows, and used in further analysis for the ultimate classification task. Support Vector Machines, multi-layer perceptrons, decision trees, and other traditional machine learning algorithms for ECG interpretation are used in the inference engine (Inan et al., 2006; Sayadi et al., 2010; Kachuee et al., 2017).

According to recent machine learning studies, automatic feature extraction and representation methods are more scalable and capable of making more accurate predictions than these manual features. They give an adequate representation of the signal. Deep learning frameworks allow the computer to learn the qualities that are most suited to the specific purpose for which it was designed (Kiranyaz et al., 2016; Jin and Dong, 2016). This method provides a more precise representation of the ECG signal, allowing the computer to compete with a human cardiologist in signal analysis. Deep learning algorithms, on the other hand, comprise a staggering number of variables that necessitate huge amounts of data to be taught.

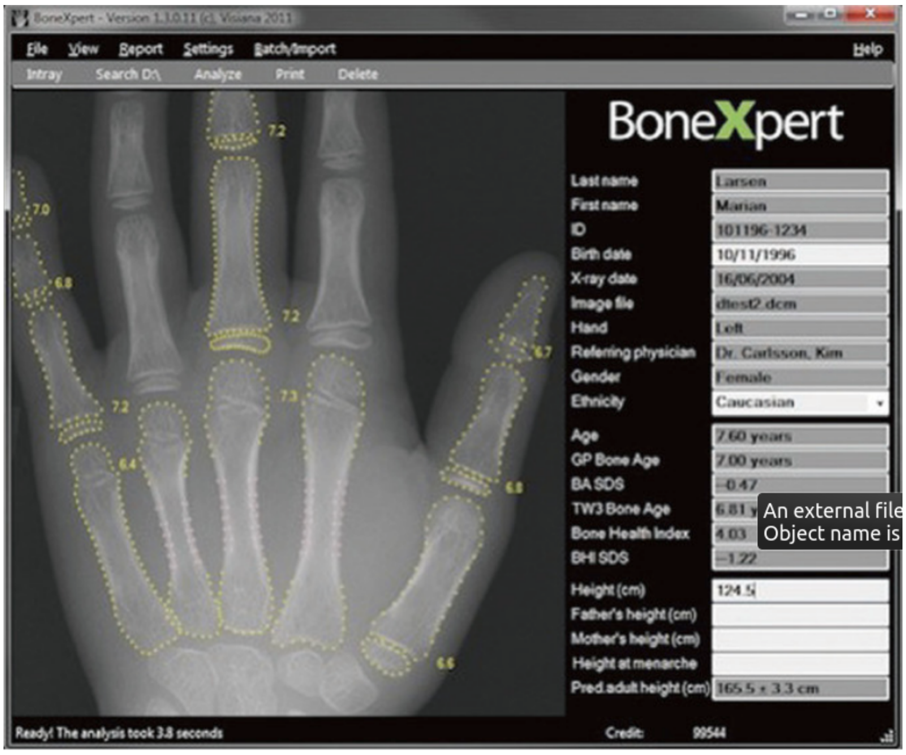
3.3.2 Bone Age Assessment

The non-dominant hand's bone growth is typically used to estimate a patient's age based on a radiograph. Ascertaining if a child's bones are developing at an acceptable rate and tracking the effects of certain drugs on bone growth are the primary purposes of the bone density test. So far, this task has been carried out manually with the use of an atlas based system such as Greulich and Pyle (GP) (Garn, 1959) or a bone scoring method like Tanner and Whitehouse (TW) (Tanner, 1983). Atlas-based techniques such as GP compare the query image to a collection of representative hand radiographs taken from people of various ages. Bones are categorized into one of several phases, which are then used to estimate an individual's age. Manual methods are time-consuming and often incorrect. An automated system for determining bone age was previously discussed. These either try to reproduce the TW or GP approaches (Niemeijer et al., 2003), or they build regression models for chronological age (Thodberg et al., 2008).

As AI and machine learning models are being used in radiology for a wide range of purposes, it is critical to comprehend how these models arrive at their conclusions. An understanding of these models helps to create trust in the software, as well as help in quality assurance. There are a multitude of ways to generate the saliency maps, also known as attention heat maps, that are used in classification and regression applications. The interpretation of AI models remains a challenge despite the fact that the field is actively researching it. Grad-CAM5 or comparable approaches are being used in the majority of investigations. The placement of an object, such as a dog in a background for a classification job, can be easily identified using this technique. For evaluating a regression model, however, where the majority of areas in the picture include information relevant to decision-making, this technique does not suffice. Heat maps can be created by plotting the gradient, or partial derivative (PD), of an outcome variable with respect to the intensity in each input image pixel and displaying this gradient in the form of a graph (Wang, 2021).

Machine learning has recently been used to identify the age of hand bones from an X-ray radiograph. The task of determining the age of a hand bone is a nice illustration of saliency map-ping research. Multiple characteristics, including the centers of ossification, size, form, and texture of individual bones, have been studied extensively in relation to the skeletal maturation of hand bones. Hand bone age is traditionally determined by radiologists by calculating the age or maturity stage of each bones and then combining the results to reach a final bone age. This is a time-consuming process that is complicated by the wide range of opinions expressed by the raters. Hand bone age determination has been automated and assisted by computer program development efforts, largely relying on traits familiar to radiologists. An example of such a software is BoneXpert which is utilizing AI algorithms for bone age assessment (Thodberg et al., 2009). BoneXpert is an AI system that analyzes a left-hand radiograph and calculates bone age based on 13 bones using feature extraction techniques. An instance of the tool is depicted in figure 3.3.2. An AI model in radiology, in general, and for hand bone age estimation, in particular, have yet to demonstrate the ability to optimally incorporate all relevant information, demanding further exploration into these models.

Artificial Intelligence (AI) may be considered a massive revolution with an unstoppable trend in its development. The focus of AI research has switched from improving system performance to figuring out how to best use this intelligence to improve human performance. Instead of competing with doctors, AI in medical imaging is projected to operate as a helper who can lighten their workload. Artificial intelligence-



Last name	Larsen
First name	Marian
ID	101196-1234
Birth date	10/11/1996
X-ray date	16/06/2004
Image file	dtest2.dcm
Hand	Left
Referring physician	Dr. Carlsson, Kim
Gender	Female
Ethnicity	Caucasian
Age	7.60 years
GP Bone Age	7.00 years
BA SDS	−0.47
TW3 Bone Age	6.81 y
Bone Health Index	4.03
BHI SDS	−1.22
Height (cm)	124.5
Father's height (cm)	
Mother's height (cm)	
Height at menarche	
Pred.adult height (cm)	165.5 ± 3.3 cm

An external file
Object name is

Ready! The analysis took 3.8 seconds

Credit: 99544

Figure 3.3.2: Automated bone age assessment by BoneXpert.

based automated bone age assessments can help alleviate the stress on radiologists who deal with many pictures. It can also greatly reduce the subjectiveness and inter- and intra-observer variability associated with traditional bone age assessment methods.

4 Machine Learning Solutions for NILM

"The way to succeed is to double your failure rate."
— Thomas J. Watson

This chapter presents novel machine learning methods for the problem of power disaggregation. The problem is faced as a single regression approach, where one model estimates the power that is consumed by a target device. The models that are proposed are mainly based on deep neural networks. The chapter is organized as follows. Firstly, related work is summarized, emphasizing previous deep learning approaches. Then, six novel deep learning solutions, that have been developed in the scope of this thesis, are presented. For each solution the experimental results and the conclusions are described.

4.1 Related Work

The electrical demand of a modern household is constantly changing with the introduction of new types of devices (smart appliances, smartphones, tablets) and new energy sources (solar cells, batteries). Given that modern buildings are already monitored by smart meters, there is a need to retrieve information related to the energy demand of a building. Energy disaggregation allows the user to extract power consumption data, turn on/off events and behavioral patterns of the residents, by measuring only the total consumption.

Energy disaggregation is the process of extracting the energy consumption of individual devices, by using a single meter that collects the aggregated energy. Using non-intrusive load monitoring techniques, there is no need to install meters for each device. Energy disaggregation was first proposed as Non-Intrusive Appliance Load Monitoring (NALM or NILM) by Hart (1992). This seminal work introduced a NILM solution based on combinatorial optimization. The main idea was to find the optimal states of the monitored appliances so that the sum of power consumption would be the same as the meter reading. This approach works only for appliances that have a finite number of states, during which the power consumption remains the same. This means that devices with continuously variable consumption cannot be monitored. Common examples are computers, electric drills and light dimmers.

Machine learning has thrived in many domains because these models can detect complex patterns. As it has been shown in previous chapter, machine learning approaches are popular in the domain of NILM as well. Usually, machine learning models are appliance specific. After a model is trained it can be deployed in the real world to monitor the consumption of the target appliance in unknown houses.

During the past decade, the most popular techniques used in NILM were Bayesian models (Marchiori et al., 2011), hidden Markov models (Egarter et al., 2015a; Parson et al., 2014; Paradiso et al., 2016) and artificial neural networks (Paradiso et al., 2013; Kelly and Knottenbelt, 2015a; Lange and Bergés, 2016; Mauch and Yang, 2015; Zhang et al., 2018a).

Kelly and Knottenbelt (2015a), describe three different architectures of artificial neural networks: a Recurrent Neural Network (RNN), a Denoising Autoencoder and a Convolutional Regressor. All three networks are trained, using the UK-DALE (Kelly and Knottenbelt, 2015b) dataset, to infer the electricity load of a specific appliance, given the total energy consumption. The performance of the proposed neural networks outperformed Hart's combinatorial optimization algorithm and other methods based on factorial hidden Markov models.

Lange and Bergés (2016) follow a different approach with the aim to decompose the aggregated signal using artificial neural networks. The model is trained to compute the active and reactive power using the current reading. The last layer of the network has binary outputs. The binary values are considered additive subcomponents of the total power draw. Then, utilizing a combinatorial optimization method, the algorithm finds which of these subcomponents correspond to each appliance. In summary, the proposed method compresses the aggregated signal into a binary representation without using any ground truth data.

Since recurrent neural networks have shown superior results in sequential data and given the two aforementioned studies, NILM research community put efforts on discovering new recurrent neural architectures. Mauch and Yang (2015) designed a deep recurrent network using Long-Short Term Memory (LSTM) layers. This solution is a point-to-point architecture because the model estimates the load of a device at a certain time point and having as input the total energy consumption of the same time point. In the same fashion Kim et al. (2016) compare the performance of Gated Recurrent Units (GRU) against traditional recurrent networks for the problem of NILM. According to the experiments, GRU networks seem to achieve better results.

In contrast to previous works that focused on recurrent architectures, Zhang et al. (2018a) present a deep neural network based on convolutional layers. The proposed method surpasses all the previous ones showcasing state-of-the-art results. One of the reasons that the model is so successful is that the model uses a window of aggregate data to estimate the power of the target appliance at the midpoint of the same window. The length of the window is 600 samples which, given a 6s sampling rate, corresponds to 1 hour of data. The method is called sequence2point because it takes as an input a sequence of mains power and predicts a point of meter power.

4.2 A Benchmark Framework for NILM

This section addresses the problem of reproducibility of NILM experiments and the difficulties to compare different NILM models. The plethora of datasets, especially when the data are not publicly available, the different environmental properties such as the sampling rate, the different dates and length of training and testing NILM models are some of the obstacles that NILM researchers face. A unified framework that addresses these issues is mandatory for the development of more advanced NILM algorithms. A novel benchmark framework for NILM experiments, which was firstly introduced by Symeonidis et al. (2019), is described in this section.

The problem of the availability of the dataset that is used for experiments can be addressed by preferring public datasets. In the literature there are many options of public datasets for the problem of power disaggregation. Two very popular datasets and widely used by the NILM research commnunity are Reference Energy Disaggregation Data Set (REDD) Kolter and Johnson (2011) and UK-DALE Kelly and Knottenbelt (2015b). They both support low frequency energy data from domestic buildings and include several houses and electric appliances. According to the scientific literature, most of the NILM experiments target the following devices: fridge, kettle, microwave, washing machine and dishwasher. These appliances are usually available in the majority of the houses. They also cover different types of devices e.g. kettle belongs to the on/off category whereas dishwasher is a multi state one. As far as the sampling rate is concerned, it has been previously described that most of the smart meters support low frequency sampling rate. In order to conduct research that is impactful in the real world researchers are advised to adhere to low

Taxonomy	Description
Category 1	Single building NILM
Category 2	Single building learning and generalization on the same dataset
Category 3	Multi building learning and generalization on the same dataset
Category 4	Multi building learning and generalization on different dataset

Table 4.2.1: Benchmark framwork for NILM systems (Symeonidis et al., 2019).

frequency data. This is a direction that the majority of NILM researchers are already following.

The biggest problem of NILM research is the standardization of the evaluation and testing methodology. Next, the problem is faced by introducing four categories of experiments with the scope to test NILM algorithms under different conditions. The four categories of experiments are "single building NILM", "single building learning and generalization on the same dataset", "multi building learning and generalization on the same dataset" and "generalization on different dataset". The benchmark framework is briefly depicted by table 4.2.1.

The first category of experiments is called "single building NILM" and is the simplest one. Training and testing of the algorithm under evaluation takes place on the same building. Of course the data are split into seen and unseen, but the important thing is that they come from the same house. The goal is to evaluate the performance of a model on an environment that is similar to the one it was trained on. A poor performance in this type of experiment indicates that the model is weak.

The second category of experiments named "single building learning and generalization on the same dataset" regards the ability of a NILM model to generalize not just on unseen data but also out-of-distribution data. Training data come from a house of a specific dataset and testing data belong to different houses of the same dataset. Learning takes place on one house and testing shows if the model is overfitting. The different houses can have different devices or energy footprint but share the same properties of the same electricity grid.

The third category of experiments is known as "multi building learning and generalization on the same dataset". This category examines the ability of a model to learn from more than one houses that come from the same dataset. Testing occurs on data from an unseen house. The goal is twofold. Firstly, to assess the generalization ability of the model. Secondly, to evaluate the ability of the model to learn from multiple houses. The intuition is that the model will eventually learn from a diversified dataset and will achieve better performance on new unseen data. The tricky part is that a large variance of the data distribution might confuse the learning model.

Finally, the fourth category is called "generalization on different dataset". It is an extension of the previous category with the difference that testing takes place using a different dataset such as a dataset from another country. Therefore testing happens on a completely different electricity grid that might have different properties e.g. more noise. In addition, the houses come from a geographically different location which means different weather conditions and consequently different energy patterns. This is considered the most difficult testing scenario because there are many unknown factors.

The framework, that has been described so far, defines the details of the experimental setup when evaluating NILM systems. It will help researchers and engineers to develop robust models which can be deployed with confidence in the real world. Moreover it standardizes the environmental conditions for an objective and fair comparison of various NILM solutions revealing their weaknesses and their strengths.

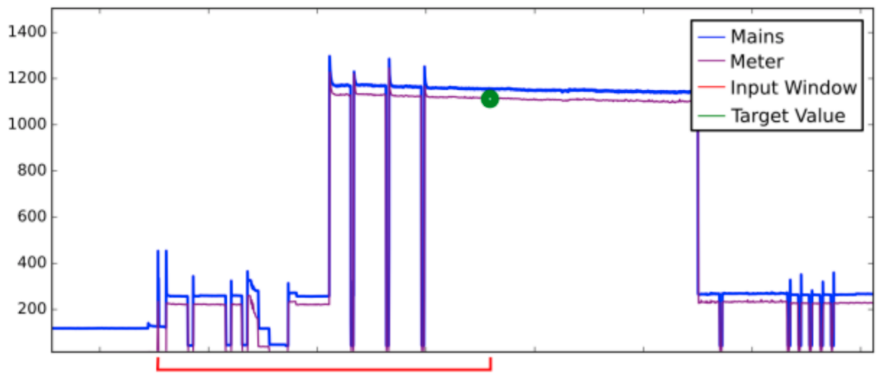


Figure 4.3.1: Example of the look back window approach.

4.3 Online GRU

At the beginning this research focused on recurrent neural networks because they proved to perform well with sequential data and there is previous work by Kelly and Knottenbelt (2015a) and Mauch and Yang (2015) that approached the problem of energy disaggregation in a similar way. The aforementioned papers both use a single point of the aggregate time series to predict the power consumption of the appliance at the same point. In other words, in order to predict the power draw of an appliance at time t, the model gets as input the aggregate draw at t. The method that is proposed in the scope of this thesis looks at a window of past aggregate data and infers the consumption at a single point. A window of length w means that the model will receive the time frame $[t - w, t]$ to estimate the consumption of the target appliance at time t. This is illustrated in Figure 4.3.1. The benefit of this approach is that the algorithm gets information within a time frame rather than just a single point without looking at future values. Therefore the look back window method can be used for real-time disaggregation solutions and is also called online.

Two variations of previous neural networks are developed and one original is proposed. The original architecture is called online GRU. All of the architectures use the window approach and are described in details next. The development is done using Keras with Tensorflow backend on GPUs. The toolkit NILMTK (Batra et al., 2014a) is used for the data loading and preprocessing. It is found experimentally that the Adam algorithm worked best for training. The loss was measured using mean squared error.

The first architecture is a recurrent neural network with a window and is based on the LSTM network of Kelly and Knottenbelt (2015a). The architecture is the same but with an input vector size of w instead of 1 value. The details of the network are depicted in Figure 4.3.2. Networks that use LSTM neurons suffer from high computational cost. First of all, each LSTM cell performs several mathematical operations before they produce their output. This makes them computationally heavy for both training and inference. Moreover, they have an internal memory cell which raises the memory demands of the model. Therefore, this kind of network may be unsuitable as NILM systems may have to run on low-cost embedded or mobile devices.

Sequence2point (Zhang et al., 2018a) is using a context window of energy data and predicts the consumption of the target appliance for the point that is in the middle of the input window. Since it has demonstrated state-of-the-art results it consists a strong baseline for comparison. The drawback of the original method is that it uses an input window of 1 hour to infer a single point. Moreover, the second half of the window consists of time points after the target consumption time point. Therefore the model is not suitable for online disaggregation, where the user expects real-time feedback. This issue is easily addressed by shrinking the input window to a time frame of 10-20 minutes. Another modification of the model is the addition of dropout layers between the convolutional layers to avoid overfitting. The architecture is

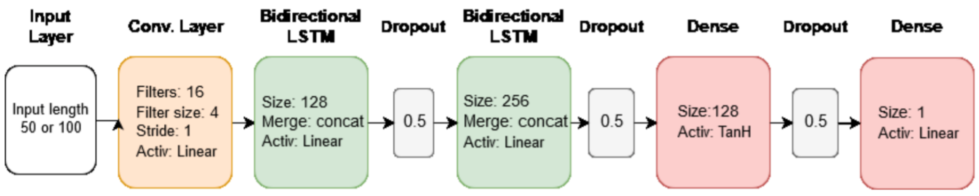


Figure 4.3.2: Architecture of the LSTM network with sliding window.

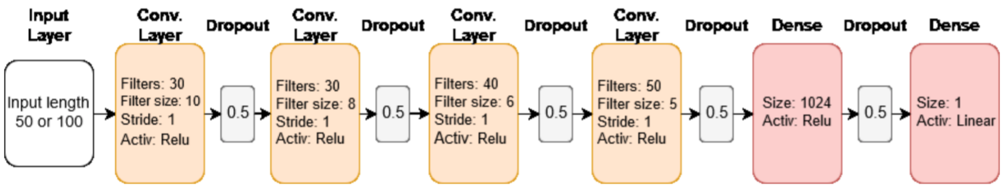


Figure 4.3.3: Architecture of the short sequence2point network.

depicted in Figure 4.3.3. Sequence2point is one of the fastest to train and test despite it has more layers with more neurons. This is due to the fact that convolutional networks are computationally light and take better advantage of parallel computing than their recurrent counterparts. This happens because neurons in a recurrent layer receive the output of every previous neuron of its layer as input and makes many sequential computations.

In an effort to design an improved recurrent neural network that addresses the drawbacks of the LSTM architecture, another one named Gated Recurrent Units (GRU) is selected. The goal is to reduce the computational complexity of the original LSTM architecture of Kelly and Knottenbelt (2015a), without trading off any performance. GRU architecture doesn't have an internal memory, thus reducing both computational and memory complexity. Previous work that compare the two recurrent architectures shows that GRU achieves similar performance with LSTM (Chung et al., 2014). Therefore the first step designing the proposed model is to replace LSTM units with GRU ones. The second step is to reduce the number of neurons per layer. As Kelly and Knottenbelt (2015a) mention, their networks are not optimized. The final architecture has recurrent layers with half neurons whereas the accuracy of the network remains the same. The total number of learnable parameters is dropped by 60%. In order to avoid overfitting, the method of dropout is used. The dropout technique proved to be useful especially for the cases where training data come from many buildings. The proposed architecture, named online GRU, is illustrated in Figure 4.3.4

All the models are trained using the UK-DALE dataset (Kelly and Knottenbelt, 2015b). The training and testing data come from different buildings as shown in Table 4.3.1. Kelly and Knottenbelt (2015a) follow the same environmental setup, except that they also used synthetic data during training. Testing takes place on unseen houses from UK-DALE and corresponds to one week data again similar with the work of Kelly and Knottenbelt (2015a). The inputs and targets are normalized using a maximum consumption value

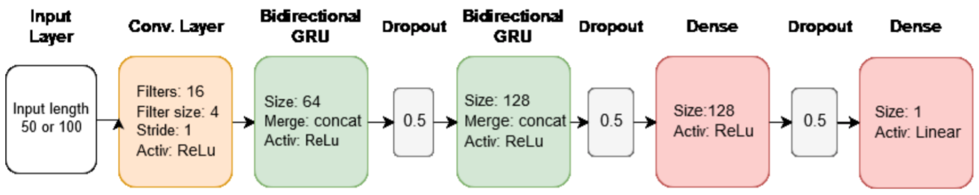


Figure 4.3.4: Architecture of the online GRU neural network.

Appliance	Training	Testing
Dish Washer	1, 2	5
Fridge	1, 2, 4	5
Kettle	1, 2, 3, 4	5
Microwave	1, 2	5
Washing Machine	1, 5	2

Table 4.3.1: Buildings used for training and testing.

Appliance	LSTM	OnlineGRU	Sequence2Point
Dish Washer	50	50	100
Fridge	50	50	50
Kettle	50	50	100
Microwave	50	50	50
Washing Machine	100	100	200

Table 4.3.2: Sliding window sizes (samples) for each device and network.

per device. For sequence2point the mean is subtracted and divided by the standard deviation according to the work of Zhang et al. (2018a). The selected values of max power, power threshold, mean power and standard deviation are listed in Table 4.3.3.

In an effort to compare the three proposed architectures with the ones presented by Kelly and Knottenbelt (2015a) all the models are tested on the same data using the same metrics. There are two categories of metrics: Precision, Recall, Accuracy and F1 Score that measure the ability of the network to detect on/off events and Mean Absolute Error and Relative Error in Total Energy that measure the ability to infer the correct power consumption value.

The length of the input window is defined specifically for each algorithm and device. At first, all windows were set to a default length of 50 samples (5 minutes). Then, through experiments it is found that some appliances perform better for different sizes. The window lengths that are selected for each pair of appliance and model are listed in Table 4.3.2.

In the same way as Kelly and Knottenbelt (2015a) and Zhang et al. (2018a) did, it is assumed that an appliance is turned on when its consumption is above a threshold that is provided by UK-DALE dataset. All thresholds are presented in Table 4.3.3. The evaluation metrics that are used are recall 3.12, precision 3.11, $F1$ score 3.13, accuracy 3.10, MAE 3.6 and relative error in total energy (RETE) which is described by 4.1.

$$RETE = \frac{|E' - E|}{max(E', E)} \tag{4.1}$$

The results produced by the experiments along with the results of the RNN from Neural NILM (Kelly and Knottenbelt, 2015a) can be seen in Figure 4.3.6. An example of the outputs of the proposed GRU architecture and the short sequence2point is depicted in Figure 4.3.5, where the ground and the disaggregation estimation of these models is illustrated. In general, there is no model clearly superior to the others. The results are mixed, with some models achieving better results depending on the target appliances and the metric.

The Neural NILM RNN performs very well on detecting events of two-state appliances such as fridge and kettle. However, it fails to predict the power consumption, which becomes obvious from the Relative

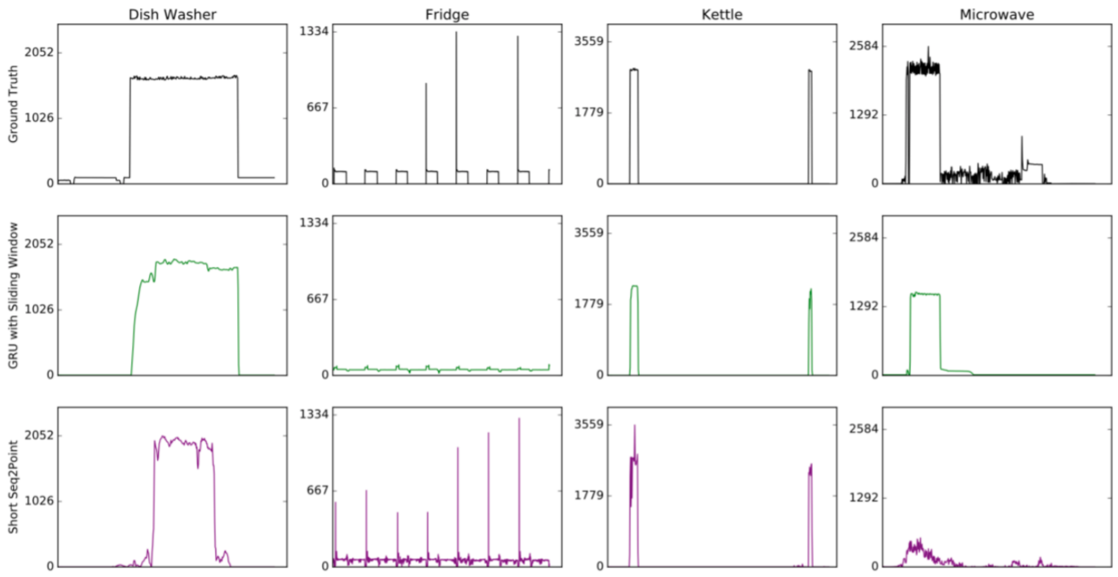


Figure 4.3.5: Example outputs of the GRU and the Short sequence2point networks.

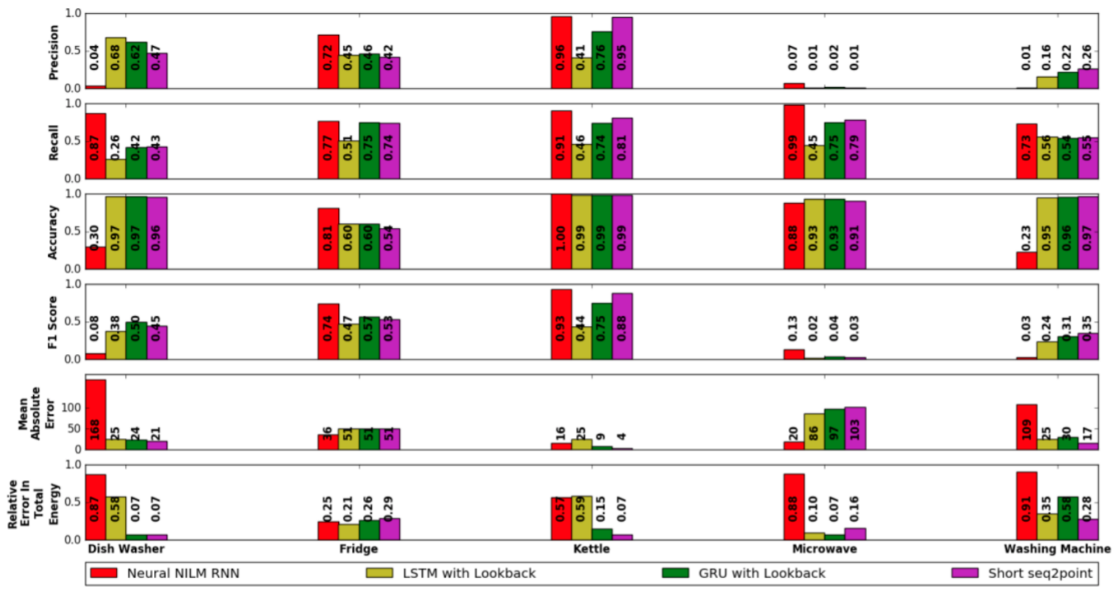


Figure 4.3.6: Experimental results including the look back window versions of LSTM and sequence2point, the proposed online GRU architecture and the original LSTM by Kelly and Knottenbelt (2015a).

Appliance	Max Power	On Power Threshold	Mean Power	SD of Power
Dish Washer	3000	10	700	1000
Fridge	200	50	200	400
Kettle	3000	2000	700	1000
Microwave	3000	200	500	800
Washing Machine	2500	20	400	700

Table 4.3.3: Appliance features used for normalization and state detection.

Error in Total Energy of the kettle. The networks that get as input the sliding window achieve better results on the multi-state devices like dishwasher and washing machine. This suggests that using information about the consumption in the previous and/or following time points, is useful for the network to recognize the behavior of the appliance.

The experimental results of the GRU network show that it performs the same or better than the LSTM architecture on all five appliances. The performance of the two networks is similar for washing machine. For the rest of the appliances the GRU network is superior. Taking into account that LSTM neurons are computationally more expensive and require more memory, it is concluded that GRU networks are better suited for the task of power disaggregation.

The short sequence2point network seems to perform almost the same with the GRU network. Kettle is an exception, where seq2point achieves clearly better scores on all metrics. Judging from the experimental results, sequence2point and online GRU are well suited for real-time disaggregation, where the user needs to receive results with minimum latency.

Microwave is the appliance that no model demonstrated sufficient performance. Microwave is a multi state appliance and its ON state energy behaviour varies with power consumption ranging from 1300 Watt to 2800 Watt and the appliance duration can be from 20 seconds to 10 minutes.

The three models that have been presented in this section are deep architectures and undoubtedly could be further optimized. Another aspect that has a large impact on the performance of a disaggregator is the length of the sliding window. In this section the default window size includes 50 samples which is equivalent to 5 mins. Washing machine shows better performance with a larger window of 100 samples regardless of the model. However, a larger window doesn't always mean better performance. This indicates that the sliding window size in dependent on the behavior of the device and should be optimized accordingly. Furthermore, according to the experiments, different neural architectures tend to favor certain types of appliances. The rolling window approach works better for the majority of the appliances, except for two-state appliances where the original LSTM architecture, with one point as input, performs better. More experiments should be conducted in order to discover the pattern that makes certain neural architectures to perform better for specific device types. Finally, the generalization of the disaggregators should be tested more extensively, including not only unseen houses from the same dataset but from other ones as well. Withing a dataset e.g. UK-DALE, the houses have appliances with similar behaviour regarding their ON state consumption. In order to execute experiments that are closer to real-world conditions, more datasets should be taken into consideration.

4.4 Stacking Neural Networks

In previous section it is demonstrated that neural networks present state-of-the-art performance in NILM. In machine learning an effective methodology to combine several models is stacking. This section studies the possibility of combining several neural network based disaggregators with the method of stacking. The goal is to achieve better results than each model individually.

The environmental setup of the experiments is based on the benchmark framework that is introduced by Symeonidis et al. (2019) and described in 3. The two datasets are REDD Kolter and Johnson (2011) and UK-DALE Kelly and Knottenbelt (2015b). The four categories of the benchmark framework are configured as follows. The first category with regards to single building NILM, includes training and testing on house 1 of UK-DALE. The test set is defined as the year following April 2016, while the rest of the data are available for training. House 1 has the most available data, thus making it suitable for both training and testing. The second category, characterized as single building learning and generalization on same dataset, includes house 1 of UK-DALE as the training set. The rest of the houses where the target appliance is present compose the test sets. The first house has a good amount of data for learning, while the rest have a few months each, so they are better for testing. The third category concerns multi building learning and generalization on same dataset. The experiments used for this category are defined for the UK-DALE dataset and follow the same structure as the ones in the study of Kelly and Knottenbelt (2015a), for tests on unseen buildings. This way a comparison with existing studies becomes easier and more direct. Finally, the hardest scenario which evaluates the generalization capability of an algorithm on different datasets, includes training data from UK-DALE and test data from REDD. The former one has buildings in the UK, while the latter one is for buildings in USA. The differences between these two can be apparent, making them a suitable choice for this category.

In line with recent research highlights in NILM, five deep neural networks are selected as base models to explore ensemble techniques. The selected networks are Denoising Auto-Encoder (DAE)(Kelly and Knottenbelt, 2015a), Recurrent Neural Network (RNN)(Kelly and Knottenbelt, 2015a), Short sequence2point (SS2P)(Zhang et al., 2018a), GRU without using a sliding window (GRU)(Krystalakos et al., 2018) and window GRU which uses a sliding window (WGRU)Krystalakos et al. (2018). The aforementioned architectures have shown promising results in NILM but their performance depends on the target appliance. A summary of these architectures is presented as follows. DAE includes a convolutional layer, 3 fully connected layers and one final convolutional layer as the output. Dropout takes place between the layers. GRU has 2 convolutional, 2 bidirectional GRU and 2 fully connected layers. RNN includes 1 convolutional, 2 bidirectional LSTM and 2 fully connected layers. WGRU consists of 1 convolutional, 2 bidirectional GRU and 2 fully connected layers, with dropout between the last four layers. SS2P includes 5 convolutional layers and 2 fully connected ones, with dropout between them.

The meta regressors that are utilized are the following. Ada Boost with Decision Tree of depth 3, 25 estimators, learning rate 0.1 and ‘square’ loss (AB-3d). Ada Boost with Decision Tree, 30 estimators and learning rate 0.5 (AB-30). Ada Boost with Decision Tree, 15 estimators and learning rate 0.5 (AB-15). Multi Layer Perceptron with one hidden layer of 100 neurons (MLP). Decision Tree Regressor of depth 5, 15% min split ratio, 9% min leaf ratio (DT5). Simple Decision Tree Regressor (DT). Gradient Boosting Regressor with 25 estimators and learning rate 0.5 (GB).

The basic idea behind stacking is combining various models, in order to achieve better results than each of the base models individually. It is especially efficient when base learners have different weaknesses. Each algorithm learns a part of the problem and the final ensemble model adds the knowledge if the individual ones. The training set is split into 2 separate parts. The first part, usually much bigger than the other, is used to train each of the base models. After each model is trained, the second part is given as an input to each model to get their predictions. The predicted values are then combined into a matrix that will make up the training input of a different learner, called the meta-learner. The target output is the same as in the second part. In this way the stacked model learns from the predictions of the base models. Next, the stacked model is ready for inference. Inference follows a similar process. Each base model is given a copy of the input and then their predictions are given as an input to the meta-model to generate the final prediction. Stacking is especially efficient when base learners make different errors.

In this work, stacking takes place in a two step process. Firstly, the five neural networks are trained using part of the training data. Each trained model is given the rest of the training data along with the test data as input for prediction and the results are saved. These are the train and test set of the stacked model. In the second step of the methodology, the generated train set is scaled and used to fit a predefined meta regressor. Finally, the predictions of the test set are given as inputs to the meta-model in order to infer the final predictions.

The preferred sampling frequency is 6 seconds. Only real data were used, with no synthetic data generation. Code is written in Python. The implementation of the used networks is based on a previous study of Krystalakos et al. (2018), which were developed using Keras with Tensorflow backend on GPUs. NILMTK is also used for loading and preprocessing the data. Scikit-learn is used for training the meta regressors. The implementation of stacking and the five neural architectures, a detailed spreadsheet containing the defined experiments for each category, as well as baseline results can be found in the following repository `https://github.com/symeonick15/NILM-Stacking`. The metrics that are used for the evaluation are F1 3.13, Relative Error in Total Energy (RETE) 4.1 and Mean Absolute Error (MAE) 3.6.

In this section, some representative results are presented. A complete set of results can be found in the supplied repository. The highlighted results indicate that they are the best among those under evaluation including base models and ensemble ones. It is noteworthy that some results in F1 score are 'nan', which means that the predictions of the model were never above the activation threshold, leading to division by zero. This happens mostly on generalization tasks, proving the difficulty of disaggregating the signal of an unseen building. On the other hand stacking seems to have the advantage of overcoming this problem.

Table 4.4.1 shows the experimental results for Category 1 with target appliance fridge. Among base models, GRU achieves the best performance. Stacking with AB-3d improves mainly F1, while AB-30 shows strong results regarding RETE and MAE. AB-3d has short trees, so it cannot be as accurate, but manages to hit the activation threshold better. AB-30 has predictions closer to the ground truth values, as it is presented by figure 4.4.1, but seems a bit unstable in the sense that there are many unexpected spikes which should be continuous values. A smoothing technique could improve it. Generally stacking has good results, especially with AB-30, which enhances regression efficiency. Table 4.4.2 shows the respective results when disaggregating fridge in category 3. The best RETE is not improved, but it's better than 3 out of 5 of base models. MAE is reduced, while DT5 demonstrates very strong results with regards to F1. As above the short tree (DT5) is better suited for classification, while it is less prone to overfitting. In general, tree-based models seem to work better when disaggregating fridge, probably due to the simple, repetitive nature of its power consumption signal.

Table 4.4.3 summarizes the experimental results of Category 1 for the device of kettle. F1 is increased significantly with the application of stacking. Especially the meta regressor AB-15 improves a lot the metric MAE and has better RETE than three out of five of base models. However the best RETE is demonstrated by base models. Again trees are the best combiners. In category 2 though, stacking did not do so well. Probably because kettle is a relatively simple device and stacking overfits during training.

The results of category 3 for kettle are summarized at table 4.4.4. Among neural networks DAE and WGRU show the best performance. Regarding stacking, GB demonstrates the best RETE, about the same MAE and 2nd best F1 after WGRU. DT behaves similarly, with a bit worse F1 and RETE, but the best MAE overall. A simple tree is again among the best meta regressors, because of the simplicity of the device's behaviour. Also boosting is susceptible to overfitting, risking generalization capabilities.

As far as dishwasher is concerned, table 4.4.5 lists the results for the first category of experiments.

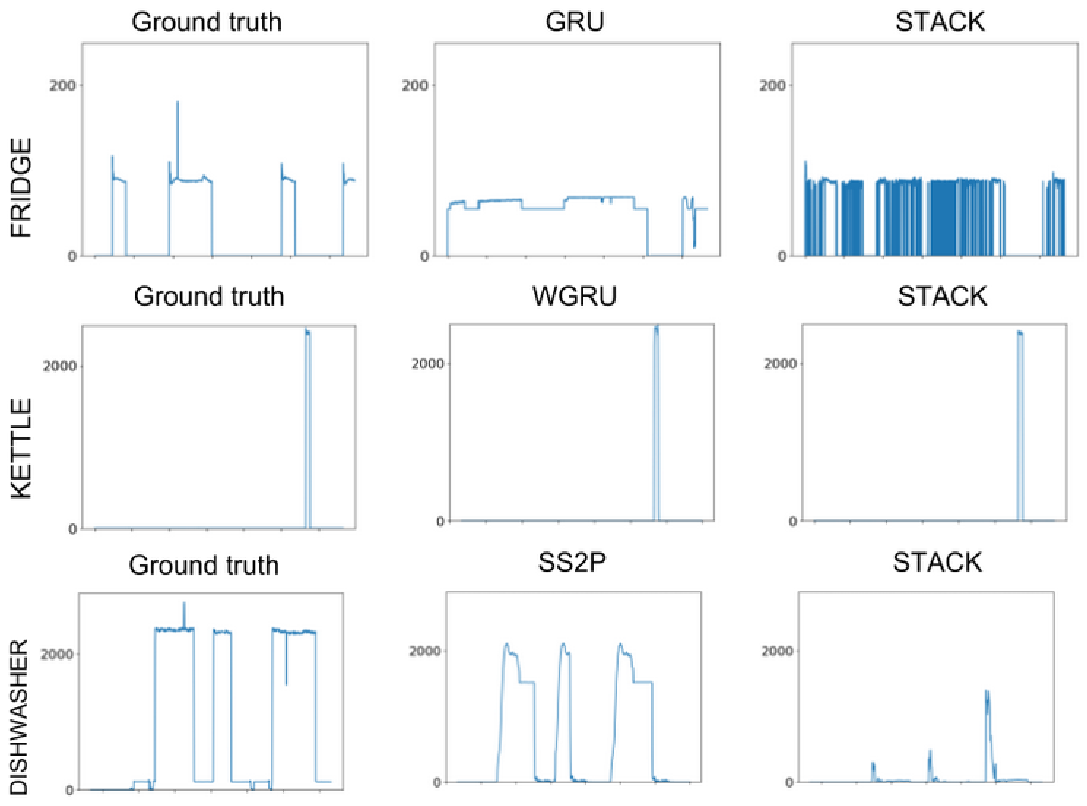


Figure 4.4.1: Sample signal plots including: original, predicted from ANN, predicted from Stacking.

Table 4.4.1: Fridge, Category 1, Train and test on UK-DALE house 1.

MODEL	F1	RETE	MAE
DAE	0.637	0.224	35.81
GRU	**0.673**	**0.130**	**34.03**
RNN	0.662	0.270	34.69
SS2P	0.651	0.215	35.23
WGRU	0.641	0.224	34.30
AB-3d	**0.674**	0.163	33.54
AB-30	0.635	**0.017**	**26.56**

Table 4.4.2: Fridge, Category 3, Train on houses 1, 2, 4 and test on 5 of UK-DALE.

MODEL	F1	RETE	MAE
DAE	0.514	0.176	**47.17**
GRU	nan	0.296	53.87
RNN	nan	0.318	53.92
SS2P	nan	**0.023**	49.18
WGRU	**0.569**	0.243	51.85
DT5	**0.641**	**0.221**	46.71
AB-30	0.525	0.225	**44.43**

Table 4.4.3: Kettle, Category 1, Train and test on UK-DALE house 1.

MODEL	F1	RETE	MAE
DAE	0.496	0.250	15.47
GRU	0.301	0.536	32.00
RNN	0.304	0.461	28.12
SS2P	0.322	**0.110**	19.95
WGRU	**0.582**	0.192	**10.78**
DT	**0.740**	**0.130**	**8.11**
AB-15	**0.804**	**0.161**	**6.37**

Table 4.4.4: Kettle, Category 3, Train on houses 1, 2, 3, 4 and test on 5 of UK-DALE.

MODEL	F1	RETE	MAE
DAE	0.504	**0.055**	10.71
GRU	0.104	0.383	24.48
RNN	0.250	0.419	43.58
SS2P	nan	0.593	11.02
WGRU	**0.663**	0.122	**10.16**
DT	0.566	0.098	**9.98**
GB	0.583	**0.033**	10.44

Table 4.4.5: Dishwasher, Category 1, Train and test on UK-DALE house 1.

MODEL	F1	RETE	MAE
DAE	0.109	**0.001**	43.61
GRU	0.471	0.140	37.06
RNN	0.467	0.072	38.54
SS2P	**0.550**	0.047	**31.01**
WGRU	0.468	0.332	31.22
MLP	**0.571**	0.195	29.46

Table 4.4.6: Washing machine, Category 3, Train on houses 1,5 and test on 2 of UK-DALE.

MODEL	F1	RETE	MAE
DAE	0.128	**0.566**	28.18
GRU	0.156	0.601	32.05
RNN	nan	0.679	33.21
SS2P	0.174	0.745	40.27
WGRU	**0.302**	**0.568**	**10.55**
AB-30	**0.324**	0.136	12.07

Stacking here seems to fail at the score of RETE. On the other hand MLP improves the best F1 and the best MAE. AB-3d had an even better MAE, but lost even more on the metrics of F1 and RETE. For this device the best meta regressor is MLP with 100 neurons, implying that a more complex model is required for this type of appliances, with multiple states and complicated behaviour. Regarding the second category of experiments of dishwasher the results are mixed: F1 is improved, while the other metrics vary, depending on the house and meta-regressor. In Categories 3 and 4 stacking is not successful. Results of washing machine category 3 are presented at table 4.4.6. Most meta-models showcase a large improvement of RETE. AB-30 increases the best F1 score, while keeping MAE low.

According to the experiments it is evident that some models are better in different categories of the benchmark framework. The advantage of stacking is that it can either improve the base models or find an intermediate solution with confidence. The tested stacked models show improved performance mostly for disaggregating simple devices e.g. fridge and kettle especially when train and test happens on the same house. Overall, tree based models are the best meta learners, while AdaBoosting can further enhance them with the risk of overfitting. This risk is made apparent in generalization experiments including the categories 2-4. The weakness of stacking is empirically shown to be the lack of generalization and poor results regarding complex target appliances. This can be attributed to the complex energy behaviour that is time dependant, along with the number of states and functions they have. Other meta regressors are advised to be evaluated such as more complex neural architectures. Another form of ensemble learning or stacking would also be interesting for future work, such as meta decision trees Todorovski and Džeroski (2003). Furthermore, due to the spiking outputs of the stacking methods that have been tested, applying a smoothing filter sounds promising.

4.5 Imaging Time Series

Transfer learning is a machine learning technique where a model acquires knowledge in one task and then reuses the same knowledge in another task. The second task doesn't necessary depend on data with the same distribution as in the first task Pan and Yang (2010); Dai et al. (2009). Transfer learning can be very handful when there is a shortage of data or there is a need for a model to generalize in different distributions. This section focuses on image representation of energy time series and the research goal is to transfer knowledge from a completely different domain. This approach, to the best of the authors' knowledge, has not been previously explored in the existing literature concerning NILM problems. It has been though applied to other time series classification tasks showing promising results.

The data that are used in this study involve low-frequency data, from the popular datasets UK-Dale Kelly and Knottenbelt (2015b)and REDD Kolter and Johnson (2011). For the transformation of energy time series to images the method proposed by Wang and Oates (2015) is used. This method is called Gramian Angular Field Matrices (GAF). Next, the image representation is inserted into a pretrained convolutional neural network, named VGG16 Simonyan and Zisserman (2014). VGG16 extracts features from the given images in the form of vectors. Finally, the output of the convolutional model is used to train a classification algorithm such as a decision tree. The classifier is used to predict whether the target device is working or not. Below previous approaches that utilize image representations of time series in the domain of NILM are presented.

De Baets et al. (2019) at their work represent the VI trajectory of appliances as images and train a siamese neural network that is used to extract a new feature space. This new representation is the input of the DBSCAN algorithm that is able to recognize appliances in a household that are left unlabeled. They use high-frequency data from the datasets PLAID Gao et al. (2014) and WHITED Kahl et al. (2016). Their approach seems to be successful in recognizing unknown appliances in a household. Wang and Oates (2015) propose a novel way of transforming time series into images, in order to test whether this new representation improves classification results. The images are constructed by transforming a time series to its polar coordinate representation. The data are scaled to one of the following intervals [-1,1] and [0,1]. The scaled time series $\tilde{X}$ can then be represented by its polar coordinates. This can be achieved by using the value of the time series and the time stamp to encode as the angular cosine and radius respectively. The aforementioned method is described by the equation below:

$$\begin{cases} \phi = arccos(\tilde{x_i}, -1 \leq \tilde{x_i} \leq 1, \tilde{x_i} \in \tilde{X}) \\ r = \frac{t_i}{N}, t_i \in N \end{cases} \tag{4.2}$$

Then, a Gramian Angular Summation/Difference Fields (GASF/GADF) matrix is constructed. This matrix encapsulates the correlation between elements of different timestamps. GASF and GADF are defined by the following equations respectively:

$$GASF = [cos(\phi_i + \phi_j)] = \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2} \tag{4.3}$$

$$GADF = [cos(\phi_i - \phi_j)] = \sqrt{I - \tilde{X}^2}' \cdot \tilde{X} - \tilde{X}' \cdot \sqrt{I - \tilde{X}^2} \tag{4.4}$$

The authors evaluate the above representation on several datasets. They use a tiled convolutional neural network for extracting features. For the classification task they use a denoising autoencoder. The results are promising as the Mean Squared Error metric is reduced by a maximum of 48% compared to approaches that use raw data.

Being inspired by the work of Wang and Oates (2015) the proposed method uses the GAF algorithm in the form of equation 4.4. The pipeline starts with the transformation of the total power consumption of a

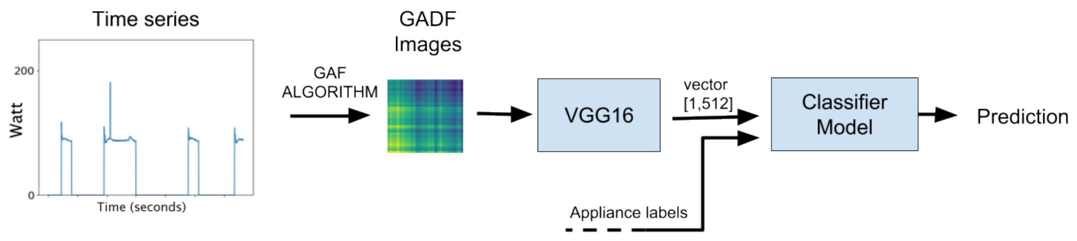


Figure 4.5.1: The pipeline of the proposed methodology that uses image representation of time series in order to identify a target appliance.

building to 64 dimensional vectors. The size of the vectors is equivalent to 6.4 minutes and is large enough to provide adequate information without introducing high latency. Then, the vectors are inserted to GAF algorithm. The output of GAF are png images with dimensions 100x100 pixels, which in turn are smoothed using PAA. The images are paired with target 64 dimensional labels that consist the ground truth energy consumption of the appliance fridge. The end-to-end process is summarized in figure 4.5.1.

VGG16 is an image classification convolutional neural network which is trained for 1000 classes using ImageNet Russakovsky et al. (2015). It is very popular in the domain of computer vision. Each image goes through the network which extracts embeddings from a new feature space that has 512 dimensions. These 512 dimensional features are the input of the classifier. The classifier is trained using the predefined labels that contain the information of whether the target device is working during the given time frame.

The classification models are selected from the library scikit-learn including: 1) Multi-Layered Perceptron (MLPC), with 50 hidden layers. 2) AdaBoost with Decision Tree learners (ABDTC), with a maximum depth of 2 and 1000 estimators, 3) AdaBoost Classifier (ABC), with its default parameters and 1000 estimators. The evaluation metric that is used is $F1$ score according to equation 3.13.

All experiments can be reproduced by cloning the implementation code from the following github repository: `https://github.com/LampriniKyrk/Imaging-NILM-time-series`. The hardware that is used for all of the model training and testing is an AMD Ryzen 5 1600x processor, an AMD R7 260x GPU and 8GB ddr4 RAM. Note that the image production takes a significantly long time, even when running in multiple threads, due to the bottleneck created by the hard disk.

The proposed method is compared against some popular NILM solutions including gated recurrent units network (GRU) Kim et al. (2016), recurrent neural network (RNN) Kelly and Knottenbelt (2015a), window GRU (WGRU)Krystalakos et al. (2018), short sequence2point (SS2P) Zhang et al. (2018a), and denoising autoencoder (DAE) Kelly and Knottenbelt (2015a). GRU consists of two convolutional layers, followed by two bidirectional GRU and two fully connected layers. RNN consists of one convolutional layer, followed by two bidirectional LSTM layers and two fully connected layers. WGRU consists of one convolutional layer and two bidirectional GRU layers who are followed by two fully connected layers. The last four layers have dropout between them. SS2P consists of a total of seven layers: five convolutional layers followed by two fully connected layers. All layers of the SS2P architecture have dropout between them. DAE consists of a convolutional layer, three fully connected layers followed by one convolutional output layer. All layers of the DAE architecture have dropout between them. The comparison between these architectures and the current method is direct using exactly the same time periods and houses for training and testing. The target is the appliance fridge by predicting on and off states.

In the first experiments category, all models are being trained on house 1 of the UK-Dale dataset and

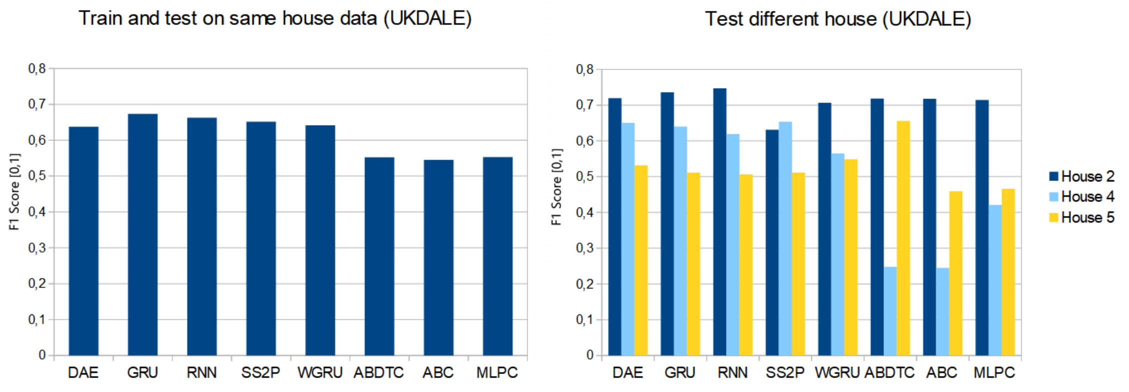


Figure 4.5.2: The bar chart on the left shows the results for all the models that are trained for the data of house 1 of UK-Dale and tested on data from the same house. The bar chart on the right shows the results for the same models with data from the unseen houses 2, 4 and 5 of the UK-Dale dataset.

the dates range from 1/4/2013 to 1/4/2014. The test also occurs for the data of the same house, and the time frame ranges from 1/4/2016 to 1/4/2017. As it can be seen in figure 4.5.2 the proposed approach gives comparable results with those from the ANN models. This experiment uses the F1 metric and evaluates how well the algorithm performs on data of the same house.

The second category of experiments uses the same trained models as above but the test occurs on different houses of the same dataset. In these experiments, it is evaluated how well the models generalize on unseen houses of the same dataset. The metric that is used is F1 score. The proposed approach seems to perform on par with the competitive models for houses 2 and 5. On house 5 the AdaBoost classifier with low depth decision trees (ABDTC) is superior to all of the neural networks. On house 4 the proposed methodology is not as accurate as the neural disaggregators, which could be due to the fact that house 1 and house 4 differ greatly with regards to the number of appliances. House 1 has 54 appliances, while house 4 has only 6.

In the third experiment category, the models are trained with data from multiple houses of the UK-Dale dataset. The houses used for the training data are 1, 2 and 4. From house 1 only the time frame from 1/4/2013 to 1/4/2014 is used for the training in order to avoid any memory issues and to make the experiments computationally lighter. This test evaluates if it is possible for a model to learn from multiple houses and be able to predict accurately on an unseen house. Some ANN models were not able to converge. The proposed method has a significantly lower F1 score from the best neural architecture, which is the window GRU.

The last experiment category tests how well a model trained in UK-Dale generalizes for the data of the REDD dataset. Note that UK-Dale is a UK based dataset while REDD is a US-based dataset. As it is shown in figure 4.5.3 most models perform well on most houses. The proposed approach is on par with the rest models on most houses and on house 3 it outperforms them.

To conclude transfer learning from generic image features to the domain of NILM is promising with experimental results being encouraging when compared against state-of-the-art NILM algorithms. The proposed methodology could be further investigated for regression and multi-label classification tasks. For future work, researchers are advised to try different imaging representations of time series as well as more complex classifiers.

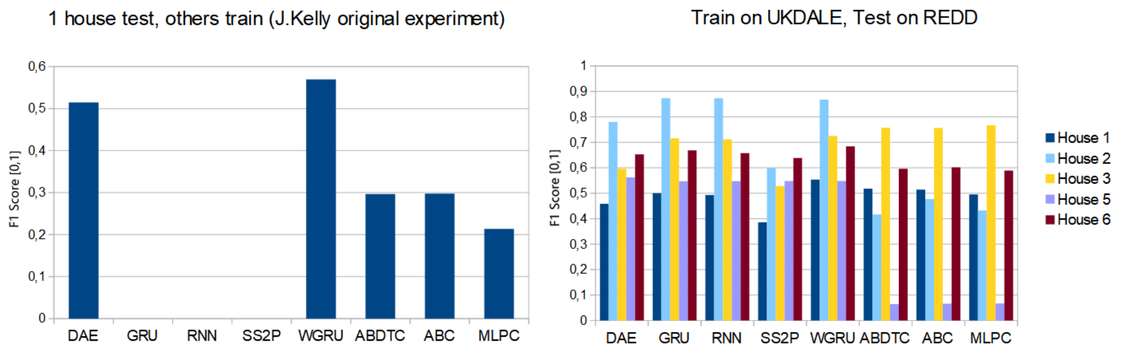


Figure 4.5.3: The bar chart on the left shows the results for all the models that are trained for the data of house 1, 2 and 4 of UK-Dale and tested on data from house 5 of UK-Dale. The bar chart on the right shows the results for the models that are trained for the data of house 1 of UK-Dale dataset and tested for the houses of the REDD dataset.

4.6 Self-Attentive Energy Disaggregation

One of the challenges in NILM research is the development of computationally light models with the aim to deploy them on the edge. Inspired by the introduction of window GRU (WGRU) (Krystalakos et al., 2018), which is the first effort to reduce the complexity of previous state-of-the-art models, in this section a novel lightweight architecture is developed.

WGRU Krystalakos et al. (2018) is composed of the following ANN layers: a convolutional layer, two Bidirectional GRU layers and one Dense layer before the output. Dropout technique Srivastava et al. (2014) is used between layers to overcome overfitting problems. In order to approximate the appliance power consumption at a single time step, a sliding window of past aggregate data points is used. The core element of the WGRU architecture, is the GRU layer, a variation of the LSTM recurrent layer.

Instead of using two GRU layers, the proposed model replaces the first GRU layer with the mechanism of attention. The proposed model is called self-attentive energy disaggregation (SAED) and, comparing to the WGRU, achieves up to 7.5 times faster training and up to 6.5 times faster inference. In terms of performance, there is a trivial trade-off which is explored thoroughly.

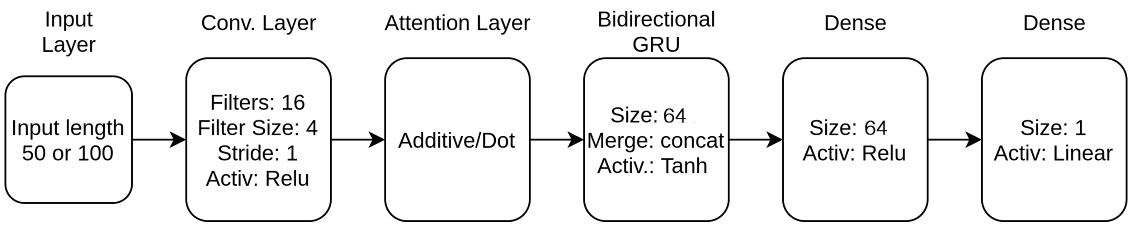


Figure 4.6.1: Architecture of the Attention model.

SAED consists of four different types of neural network layers. In order to extract new features from the input, the first layer is a 1D convolutional one. Convolution is time invariant and can learn local patterns found at certain positions of the sequence. Using the attention mechanism, the network learns to focus on the most important features. Next, the output of the attention layer is the input to a GRU layer, to recognize possible sequential patterns. Finally, there is a dense layer, functioning as a regressor and predicting the output. A graphical representation of the architecture is depicted in Fig.4.6.1. The attention layer operates

Model	Parameters	Size(MB)
WGRU	270k	3.3
Seq2Point	2600k	31.3
SAED-dot	40k	0.54
SAED-add	42k	0.54

Table 4.6.1: Learnable Parameters and Size of the models.

as a self-attention mechanism and it has two variations. The first variation is additive and the second dot attention, mentioned as SAED-add and SAED-dot respectively.

The proposed network is developed in Python 3.8 and Pytorch. NILMTK framework Batra et al. (2014a) is used for data loading. The selected optimization algorithm is Adam Kingma and Ba (2014) and the loss function is the MSE loss. The experiments are executed on a Nvidia GPU GTX-1060 6Gb.

As far as the experiments are concerned no artificial data are considered. The sampling rate is 6 seconds and the batch size 1024. Seven electric devices are selected for the experiments; dish washer (DW), fridge (FZ), kettle (KT), microwave (MW), washing machine (WM), television (TV) and computer (PC). The optimal size of the sliding windows depends on the device and on the algorithm (Krystalakos et al., 2018). In this work, time window is 50 samples for all the target appliances and models with the exception of washing machine, which has a window of 100 samples. The experiments are conducted comparing three different architectures; the proposed SAED architecture, the WGRU (Krystalakos et al., 2018) and the Seq2Point (Zhang et al., 2018a) as implemented by Krystalakos et al. (2018). The number of learnable parameters of all the models are presented in Table 4.6.1. SAED models have 65 and 6.5 times less parameters than Seq2Point and WGRU respectively, resulting to considerably smaller storage requirements in the real world. It should be noted that for the devices television and computer, the dropout ratio for the Seq2Point model is set to 25%. For all the models the training duration is 5 epochs. The environmental setup follows the benchmark framework by Symeonidis et al. (2019). Therefore the experiments are classified in four categories; single building NILM, single building learning and generalization on same dataset, multi building learning and generalization on same dataset and generalization to different dataset.

The first category of experiments regards training and testing on the same house at different time periods. Models with low performance in these experiments are considered weak. In the second category of experiments training and testing take place on different houses of the same dataset. These experiments serve the purpose of measuring the generalization capability of a model when it is tested on different houses. Different houses have different energy footprint that depends on multiple factors such as the different habits of the residents, the use of other electric devices and others. Similarities between measurements of the same data set are also expected. Properties like electricity grid, weather conditions and regionality are some possible factors. The third category concerns experiments where training data come from different buildings and testing is happens on unseen environment. The buildings belong to the same dataset. For the fourth category training includes different houses and testing takes place on houses of a different dataset. The purpose of the last two categories of experiments is to evaluate the capability of an algorithm to learn from a multiple sources. The fourth category is the hardest task, because testing regards unknown data from an unknown dataset. So far no model has been successful in the last two categories of experiments.

Due to the large number of devices employed, we classified them according to the data sets used for training and inference. The trials with the first group of devices used the UK-DALE Kelly and Knottenbelt (2015b) and REDD Kolter and Johnson (2011) data sets, whereas the experiments with the second group used the REFIT REFIT Firth et al. (2017) and UK-DALE data sets. It should be mentioned that UK-DALE

Device	Category1		Category2		Category3		Category4	
	Train	Test	Train	Test	Train	Test	Train	Test
DW	1	1	1	2,5	1,2	5	1,2	1,2,3,4,6
FZ	1	1	1	2,4,5	1,2,4	5	1,2,4	1,2,3,5,6
KT	1	1	1	2,3,4,5	1,2,3,4	5	-	-
MW	1	1	1	2,3,5	1,2	5	1,2	1,2,3,5
WM	1	1	1	2,4,5	1,5	2	1,5	1,2,3,4,5,6
TV	6	6	6	14,17,19	6,17	14,19	6,17	1,5
PC	6	6	6	16,17,19	6,17	16,19	6,17	5

Table 4.6.2: Buildings used for train and test. For the first 4 devices: In Categories 1-3, UK-DALE was used for training and testing. In Category 4, UK-DALE was used for training and REDD was used for testing. For the last 3 devices: In Categories 1-3, REFIT was used for training and testing. In Category 4, REFIT was used for training and UK-DALE was used for testing.

and REDD contain data on the power consumption of households in the UK and the USA, respectively, whereas REFIT contains data on the power consumption of 20 residential houses in the Uk, using a broader range of devices than UK-DALE.

Categories of experiments 1-3 for the first group of devices (dish washer, fridge, kettle, microwave, and washing machine) were performed on the UK-DALE data set, while category 4 inference was evaluated on the REDD data set. The fourth category of kettle experiments was not carried out due to a lack of data from kettle devices in the REDD data set. Training for categories 1 and 2 of experiments was carried out on UK-house DALE's 1 during the first nine months of 2013, with testing taking place in the last three months of the same year. The ratio of training versus inference data in category 3 and 4 experiments varies depending on the device.

For the devices of the first group (dish washer, fridge, kettle, microwave and washing machine), categories of experiments 1-3 are executed on the UK-DALE data set, while for category 4 testing takes place on the REDD data set. Due to the lack of kettle device data in the REDD data set, the fourth category of experiments on kettle is not conducted. Training for categories 1 and 2 of experiments takes place on house 1 of UK-DALE during the first 9 months of 2013 while the last 3 months of the same year are used for testing. Regarding the experiments of categories 3 and 4, the ratio of training versus testing data depends on the device.

For the remaining electric devices (television, computer) REFIT Firth et al. (2017) is used for training, while data from UK-DALE are used for testing. In the experiments on this device group, three months of REFIT data are used, while testing is performed on one month measurements. These experiments may shed light on how the models perform in the presence of sparse data. As a result, the models do not perform well in certain types of experiments. All the experiments are described in Table 4.6.2.

The following metrics are calculated for model evaluation and comparison: F1 score 3.13, Relative Error in Total Energy (RETE) 4.1, and Mean Absolute Error (MAE) 3.6. The ability of model to detect on/off energy states is evaluated with F1 score. MAE (measured in Watts) and RETE (dimensionless) evaluate the capability of the models to estimate the actual electric power consumption of the device.

To examine the suggested SAED model's generalization properties even further, the inclusion of additional metrics is necessary. According to Klemenjak et al. (2019) the number of observed and unseen installations in which a model is evaluated should be considered. As a result, the concept of generalization loss (G-loss) is introduced. The intuition is that there may be a change in the value of a metric between visible and invisible installations. This implies that the model's performance has changed when evaluated on previously undisclosed data. Whether the metric is used to evaluate event detection or power approximation, the G-loss is calculated as described in eq. 4.5 or eq. 4.6, where u stands for unseen and s for seen installations. For instance, a calculated G-loss of 15% on the F1 score indicates that the measured F1 score on the unseen house data is 15% lower than the measured F1 score on the seen house data where the training occurred. On the other hand, a 10% G-loss on MAE indicates that the error observed on unseen data is 10% larger than the error observed on seen building measurements.

The mean of all the G-losses obtained for unseen houses is called the mean generalization loss (MGL), which is used to indicate the overall performance loss. Additionally, accuracy on unseen homes (AUH) and error on unseen houses (EUH) can be calculated in order to evaluate an architecture's generalization properties. The above metrics are given by eq. 4.7-4.9, where N is the number of the unseen building.

$$G - loss = 100(1 - \frac{F1_u}{F1_s}) \tag{4.5}$$

$$G - loss = 100(\frac{MAE_u}{MAE_s} - 1) \tag{4.6}$$

$$MGL = \frac{1}{N}\sum_{i}^{N} G - loss_i \tag{4.7}$$

$$AUH = \frac{1}{N}\sum_{i}^{N} F1_{ui} \tag{4.8}$$

$$EUH = \frac{1}{N}\sum_{i}^{N} MAE_{ui} \tag{4.9}$$

A different aspect of generalization is suggested by D'Incecco et al. (2020). The idea is that extracted features from "complex" equipment used for training could be utilized to disaggregate appliances with "simpler" electric signatures. The primary benefit of this concept is that it accelerates training, requiring fewer computational resources. The authors present two scenarios for model knowledge transfer: appliance-based (ATL) and cross-domain (CTL). In CTL schema, a model is trained and tested on different data sets, in similar way as in the benchmark method described in Symeonidis et al. (2019). In ATL scenario, the model is trained and fine-tuned before the final inference. This article compares the models based on the ATL scenario.

Three degrees of comparison are used to evaluate the models. To begin, in terms of performance across the four experimental groups. Following that, we'll discuss generalization by calculating the generalization loss for unseen data. Additionally, on the possibility of latently acquired knowledge being transferred. Finally, the inference speed for various data sizes is calculated in order to compare the models' scalability. The most important results are summarized in Tables 4.6.3 – 4.6.5, where the mean training epoch time in seconds is noted as *time/ep* and the best values are highlighted. The complete python code and the produced results are contained in: *https://github.com/Virtsionis/SelfAttentiveEnergyDisaggregator*.

As shown in Table 4.6.3, SAED models perform on par with WGRU and Seq2Point for dish washer in category 1. In terms of training time per epoch, SAED is up to 7.1 times faster than WGRU, but nearly as fast as Seq2Point. In category 2, SAED-dot is the clear winner, with similar metric values to the SAED-add model but nearly half the training time per epoch compared to the WGRU. In category 3 of the same device, the SAED models perform similarly to the Seq2Point, while the WGRU outperforms it in the error metrics. SAED-dot is the fastest in terms of speed. In category 4, SAED-add achieves a higher F1 score and MAE, while having the lowest RETE. The overall conclusion is that SAED outperforms the WGRU and the Seq2point on the dish washer, with faster training and greater performance in categories 2 and 4. For

Device	Cat.	Train	Test	Model	F1s	RETE	MAE	time/ep
DW	1	1	1	WGRU	**0.33**	**0.17**	13.22	550
				Seq2Point	0.31	0.35	15.44	79
				SAED-dot	0.28	0.31	13.03	77
				SAED-add	0.25	**0.17**	**12.03**	141
DW	2	1	2	WGRU	0.26	0.77	37.47	550
				Seq2Point	0.35	0.83	41.77	79
				SAED-dot	**0.63**	**0.62**	**33.48**	77
				SAED-add	0.6	0.63	34.31	141
DW	3	1,2	5	WGRU	0.33	**0.34**	**20.75**	575
				Seq2Point	**0.35**	0.7	40.52	108
				SAED-dot	0.33	0.57	26.45	74
				SAED-add	0.25	0.62	31.01	138
DW	4	1,2	4	WGRU	0.3	0.65	**8.6**	575
				Seq2Point	0.31	0.2	13.1	101
				SAED-dot	0.45	**0.1**	12.71	74
				SAED-add	**0.53**	0.77	**8.6**	138
WM	1	1	1	WGRU	**0.54**	**0.12**	**16.55**	1097
				Seq2Point	0.25	0.15	18.5	150
				SAED-dot	0.51	0.26	18.51	147
				SAED-add	0.45	0.29	28.55	416
WM	2	1	2	WGRU	**0.34**	0.43	**10.45**	1097
				Seq2Point	0.1	0.66	20.57	150
				SAED-dot	0.3	**0.34**	13.1	147
				SAED-add	0.3	0.53	22.01	416
WM	3	1,5	2	WGRU	0.12	0.36	22.74	585
				Seq2Point	0.14	**0.16**	17.2	147
				SAED-dot	0.19	0.36	**14.66**	81
				SAED-add	**0.2**	0.21	15.18	81
WM	4	1,5	1	WGRU	**0.26**	0.66	43.65	585
				Seq2Point	0.22	0.54	42.22	147
				SAED-dot	0.18	**0.39**	50.65	84
				SAED-add	0.18	0.7	**41.93**	81
FZ	1	1	1	WGRU	**0.63**	0.27	33.29	562
				Seq2Point	**0.63**	0.3	33.2	78
				SAED-dot	0.59	0.17	32.78	73
				SAED-add	0.59	**0.13**	**30.56**	145
FZ	2	1	2	WGRU	0.82	**0.13**	28.46	562
				Seq2Point	**0.91**	**0.13**	33.43	78
				SAED-dot	0.82	0.21	**26.86**	73
				SAED-add	0.84	0.23	27.33	145
FZ	3	1,2,4	2	WGRU	**0.52**	0.18	51.18	519
				Seq2Point	**0.52**	**0.03**	**49.52**	74
				SAED-dot	**0.52**	0.29	51.35	69
				SAED-add	**0.52**	0.22	50.52	70
FZ	4	1,2,4	1	WGRU	**0.53**	0.32	**52.57**	519
				Seq2Point	0.42	**0.27**	60.06	72
				SAED-dot	0.49	0.29	50.89	69
				SAED-add	0.5	0.33	51.39	70

Table 4.6.3: Performance Comparison for Dish Washer(DW),Washing M.(WM),Fridge(FZ).

washing machine in category 1, SAED-dot is 7.5 times faster than WGRU trading with a maximum 10% performance in the metrics F1 and MAE. SAED-dot performs similarly to WGRU in category 2, but with 7.5 times faster training time each epoch. In category 3, the SAED-add has the highest F1 score, whereas Seq2Point has the lowest RETE. The SAED models outperform the other models in terms of MAE in this group of experiments. SAED models are trained faster in the fourth category, with lower RETE and MAE values. It is worth noting that when dish washer and washing machine are the target appliances, the SAED models exhibit equivalent or superior performance than the state-of-the-art models, with training time per epoch being up to 7.5 times faster than the WGRU. Furthermore, when disaggregating the washing machine, Seq2Point performs poorly. Results for the Fridge are summarized in Table 4.6.3. State-of-the-art models produce higher F1 scores in categories 1 and 2, whereas SAED-add demonstrates promising results with the smallest RETE and MAE, achieving up to 4 times faster training times than the WGRU. In categories 3 and 4, however, all models perform the same, confirming the SAED method's high generalization potential.

In categories 1 and 2 of kettle, shown in Table 4.6.4, the models have comparable RETE and MAE values, but WGRU achieves the best F1 score in 7.7 slower training time than the SAED-dot. In the third category of experiments, the WGRU is the winner in terms of F1 and RETE, whereas in MAE all the models perform the same. These findings suggest that, as compared to WGRU, SAED models has difficulty disaggregating devices with simple behavior, such as fridge and kettle. Kettle has two states whereas Fridge has a limited number of states with a repeating pattern. SAED gets low F1 scores in categories 1-2 of fridge and kettle, but it performs well in categories 3-4 for fridge. Low F1 scores suggest that the models have trouble identifying on/off states of the target devices. It is also worth noting that SAED outperforms Seq2Point on kettle categories 1 and 2 and has somewhat faster training times. The results of microwave experiments are also included in Table 4.6.4. WGRU is unquestionably the winner in categories 1-2. SAED models outperform both WGRU and Seq2Point in the third category of experiments, but WGRU gets a 17% higher F1 score than SAED in ten times slower training time in category 4. Given that microwave is a multi-state device with variable power consumption and on-state duration, SAED models outperform the state-of-the-art. In general, SAED models perform well when disaggregating multi-state devices rather than simpler devices. Additionally, SAED performs well in experiments of categories 3-4, demonstrating the proposed models' generalization capabilities.

According to Table 4.6.4, SAED models outperform state-of-the-art models in category 1 of television, achieving identical F1 scores while exhibiting reduced MAE errors and faster training periods. Additionally, all models perform similarly in category 2 experiments, with Seq2Point scoring a 5% higher F1 score, whereas SAED-add achieves lower RETE and MAE and training at a quicker rate each epoch. Notably, all models perform better in this category of experiments than category 1. SAED-dot gets a 34% higher F1 score in category 3 than state-of-the-art models. All models perform similarly in terms of RETE, whereas SAED models score up to 60% lower in terms of MAE. SAED models outperform the others in the fourth category of tests.

The results of the experiments regarding the device of a computer are shown in Table 4.6.5. In category 1, SAED-add and Seq2Point perform similarly in terms of recognizing on/off events, with up to 35% better F1 scores than WGRU. In terms of RETE values, WGRU outperforms all other models, whereas MAE scores are practically the same for all models. When SAED variations are compared, SAED-add performs 10% better than SAED-dot, albeit with a longer training time each epoch. In category 2, WGRU outperforms SAED models marginally. Specifically, WGRU achieves a 14% improvement in F1 score, with a maximum improvement of 66%. WGRU achieves lower values for RETE and MAE. SAED-dot model is the clear winner in categories 3 and 4, attaining higher F1 and MAE scores, whereas WGRU has a lower RETE value. In all categories of the experiments, SAED-dot is about twice as quick as SAED-add and up to three times faster than WGRU in terms of training time per epoch. On the contrary, despite the large

Device	Cat.	Train	Test	Model	F1s	RETE	MAE	time/ep
KT	1	1	1	WGRU	**0.65**	**0.09**	**7.35**	563
				Seq2Point	0.28	0.24	17.6	79
				SAED-dot	0.44	0.14	8.57	73
				SAED-add	0.34	0.26	9.46	143
KT	2	1	2	WGRU	**0.9**	0.31	**14.04**	563
				Seq2Point	0.39	0.36	29.8	79
				SAED-dot	0.62	0.3	19.03	73
				SAED-add	0.49	**0.28**	17.35	143
KT	3	1,2,3,4	5	WGRU	**0.41**	**0.05**	**9.92**	1096
				Seq2Point	**0.41**	0.56	10.44	150
				SAED-dot	0.27	0.27	12.24	141
				SAED-add	0.31	0.18	10.95	271
MW	1	1	1	WGRU	**0.32**	**0.09**	6.29	560
				Seq2Point	0.22	0.35	**6.01**	79
				SAED-dot	0.16	0.14	7.51	74
				SAED-add	0.18	0.16	7.61	144
MW	2	1	2	WGRU	**0.44**	0.25	**4.36**	560
				Seq2Point	0.37	0.54	5.29	79
				SAED-dot	0.25	0.19	5.97	74
				SAED-add	0.26	**0.17**	5.98	144
MW	3	1,2	5	WGRU	0.08	0.59	60.53	440
				Seq2Point	0.1	0.555	59.61	41
				SAED-dot	0.21	0.58	56.93	41
				SAED-add	**0.22**	**0.51**	**59.36**	41
MW	4	1,2	1	WGRU	**0.41**	0.19	23.53	440
				Seq2Point	0.36	**0.08**	**22.68**	74
				SAED-dot	0.34	0.2	25.67	41
				SAED-add	0.34	0.15	25.13	41

Table 4.6.4: Performance Comparison for Kettle (KT), Microwave(MW), Television(TV).

number of parameters, Seq2Point achieves nearly identical training times as SAED-dot.

To explore on a deeper level the generalization ability of SAED, in comparison to WGRU and Seq2Point, the calculation of specific metrics takes place. Table 4.6.6 presents the values of AUH, EUH alongside with the corresponding MGL calculations. These metrics are derived using F1 scores and MAE values from experiments classified as category 1. Due to the small size of the experiments, only a subset of the measurements is employed. To compare the models, MGL values are used, where a lower number indicates a better model. On all test devices except computer, SAED models achieve the lowest scores for MGL and Classification Accuracy. Thus, SAED demonstrates a high degree of generalization when recognizing on/off events. Additionally, negative MGL values imply that SAED models perform better on unseen than on seen houses. With regards to MGL and Estimation Accuracy, mixed findings are found, with SAED exhibiting finer values than state-of-the-art models for the devices washing machine and television. As a result, SAED appears to generalize better than WGRU in terms of power estimation levels for these test devices. Seq2Point returns lower values for dishwasher and kettle. The preceding results emphasize that

Device	Cat.	Train	Test	Model	F1s	RETE	MAE	time/ep
TV	1	6	6	WGRU	**0.68**	0.49	40.38	145
				Seq2Point	**0.68**	0.52	42.63	54
				SAED-dot	0.67	0.5	35.68	54
				SAED-add	0.65	**0.41**	31.67	102
TV	2	6	17	WGRU	0.79	0.36	32.06	144
				Seq2Point	**0.8**	0.4	32.94	57
				SAED-dot	0.75	0.34	32.41	53
				SAED-add	0.72	**0.24**	**30.15**	101
TV	3	6,17	14	WGRU	0.31	0.65	36.5	164
				Seq2Point	0.31	0.65	36.3	102
				SAED-dot	**0.47**	**0.6**	**14.37**	62
				SAED-add	0.39	0.67	15.08	114
TV	4	6, 17	1	WGRU	0.14	0.79	42.21	165
				Seq2Point	0.14	0.72	36.4	102
				SAED-dot	**0.56**	0.52	**9.02**	62
				SAED-add	0.49	**0.35**	9.66	112
PC	1	6	6	WGRU	0.34	**0.33**	45.97	148
				Seq2Point	**0.54**	0.46	40.44	53
				SAED-dot	0.43	0.5	44.2	52
				SAED-add	0.51	0.44	**40.1**	101
PC	2	6	17	WGRU	**0.78**	**0.54**	**36.52**	145
				Seq2Point	0.62	0.62	46.72	50
				SAED-dot	0.67	0.65	52	51
				SAED-add	0.62	0.62	48	100
PC	3	6,17	16	WGRU	0.27	0.7	40.2	169
				Seq2Point	0.27	0.62	30.66	105
				SAED-dot	**0.37**	**0.37**	**17.05**	63
				SAED-add	0.34	0.51	20.52	112
PC	4	6,17	5	WGRU	0.54	**0.1**	45.36	167
				Seq2Point	0.54	0.19	55.95	105
				SAED-dot	**0.71**	0.2	**32.41**	62
				SAED-add	0.62	0.2	35,84	112

Table 4.6.5: Performance Comparison for Television(TV), Computer(PC).

SAED demonstrates strong generalization in the context of NILM.

To investigate the suggested method's capacity for knowledge transmission, a transfer learning schema is used. The models are initially trained using washing machine. The network is then fine-tuned on the target device. Finally, inference is done on the target device's unseen data. The data for all the stages of this experiment are the same as the category 1 described in Table 4.6.2. The results on kettle are presented on Table 4.6.7. In comparison to the results of Table 4.6.5, WGRU performs in the same fashion, whereas SAED-add and Seq2Point perform better. It should be highlighted that fine-tuning and testing on more devices revealed unsatisfactory results, indicating that this method should be used with devices that have comparable electric signatures.

When comparing models, an essential and usually overlooked parameter is the inference time. Large-scale applications entail feeding batches of data from numerous residences to disaggregation models. The cost of this application is crucial and is highly dependent on the model's inference time and scalability. Scalability is demonstrated by expanding the disaggregation time period from one day to three months and monitoring the inference time for various test data sizes. As illustrated in Fig. 4.6.2, SAED models achieve comparable inference times to Seq2Point, whereas WGRU is significantly slower.

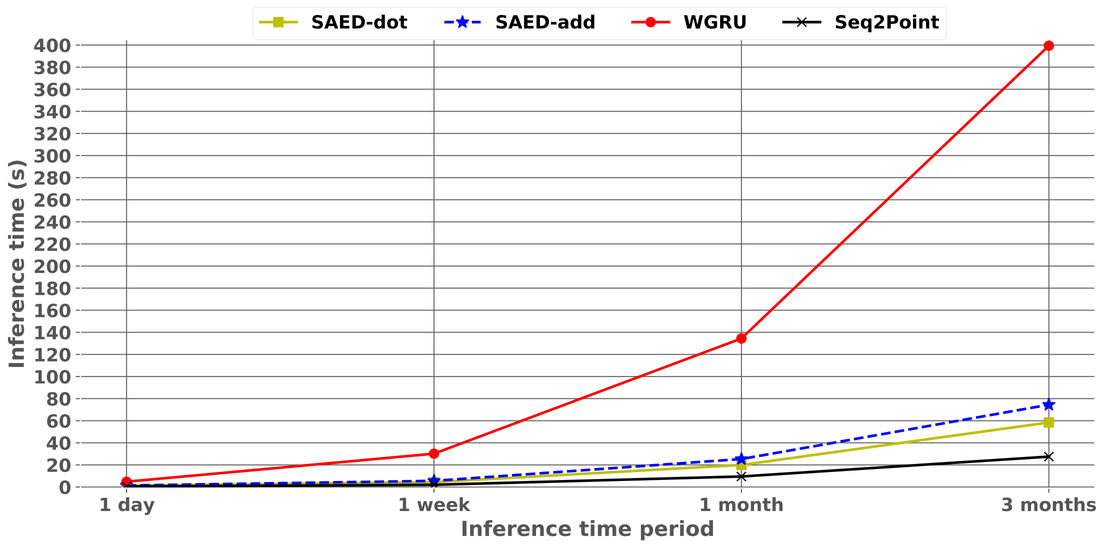


Figure 4.6.2: Inference time versus inference time period for fridge, where 1 day of data is equal to 14400 samples.

As shown in the preceding sections of this work, the attention mechanism enhances the generalization and performance of a lightweight neural network. An ablation experiment is carried out to quantify these enhancements, in which the same network is tested in some conditions without the attention mechanism and is used as a baseline. The models were specifically compared side by side on experiments from categories 1 and 2, as described in Table 4.6.2. The results are shown in Fig. 4.6.3. On both types of tests, SAED models outperformed the baseline model in terms of F1 scores. Furthermore, SAED-dot model earns the greatest F1 score on category 1, whilst SAED-add model achieves the best results on microwave and fridge regarding category 2. In terms of MAE errors, the results are mixed, with the baseline performing similarly to SAED models. The discrepancies in F1 scores between SAED and baseline emphasize the fact that the attention mechanism aids the network in detecting energy changes. As a result, the SAED approach detects more on/off events than the baseline method.

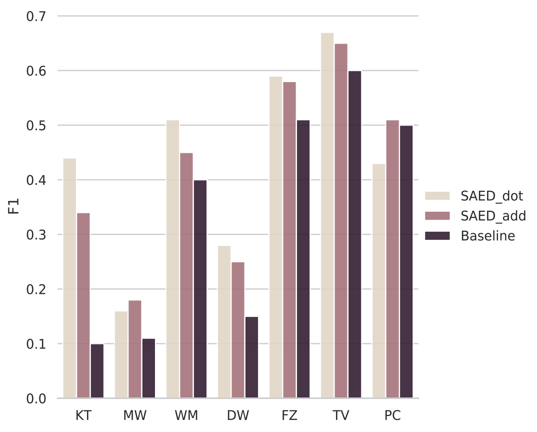
When the proposed SAED models are compared to the lightweight state-of-the-art WGRU, promising results are obtained. In general, SAED models produced equivalent and, in some cases, superior results

			Classification Accuracy			Estimation Accuracy		
Device	S|U	Model	F1s	AUH	MGL[%]	MAEs	EUH[W]	MGL[%]
DW	1|2,5	WGRU	0.33	0.26	19.3	13.22	31.29	136.7
		Seq2Point	0.32	0.32	−4.85	15.44	31.4	**103.35**
		SAED-dot	0.28	0.48	−72.5	13.03	30.42	133.6
		SAED-add	0.25	0.45	**-82**	12.03	31.72	163.6
WM	1|2,5	WGRU	0.54	0.29	46.9	16.55	25.02	51.2
		Seq2Point	0.25	0.12	54	18.55	29.14	57.5
		SAED-dot	0.51	0.27	48.1	18.51	23.56	27.3
		SAED-add	0.45	0.26	**43.4**	28.55	35.1	**22.9**
FZ	1|2,5	WGRU	0.63	0.69	−9.8	33.3	34.08	**2.3**
		Seq2Point	0.63	0.67	−5.55	33.2	37.46	12.85
		SAED-dot	0.59	0.69	−18.4	32.78	32.85	3.25
		SAED-add	0.59	0.7	**-19.65**	30.56	32.68	6.89
KT	1|2,5	WGRU	0.66	0.59	9.9	7.35	24.44	232.5
		Seq2Point	0.24	0.3	−7.15	17.6	27.85	**58.25**
		SAED-dot	0.44	0.45	−2	8.57	23.49	174.1
		SAED-add	0.33	0.37	**-10.4**	9.46	21.05	122.5
MW	1|2,5	WGRU	0.32	0.33	−1.7	6.29	12.79	**103.5**
		Seq2Point	0.22	0.28	−25	6.01	13.34	125.05
		SAED-dot	0.16	0.26	**-68.6**	7.5	18.07	140.9
		SAED-add	0.18	0.28	−53.9	7.61	17.59	131.2
TV	6|1,17	WGRU	0.68	0.52	24.25	40.38	24.27	−39.05
		Seq2Point	0.68	0.49	28.7	42.63	32.67	−23.35
		SAED-dot	0.67	0.55	18.7	35.68	21.75	**-39.7**
		SAED-add	0.65	0.53	**18.45**	31.67	20.67	−34.7
PC	6|16,17	WGRU	0.34	0.54	**-58.8**	45.97	30.74	**-33.15**
		Seq2Point	0.54	0.51	5.55	40.44	31.92	−21.1
		SAED-dot	0.43	0.51	−18.6	44.2	34.75	−21.4
		SAED-add	0.51	0.49	3.9	40.1	31.81	−20.65

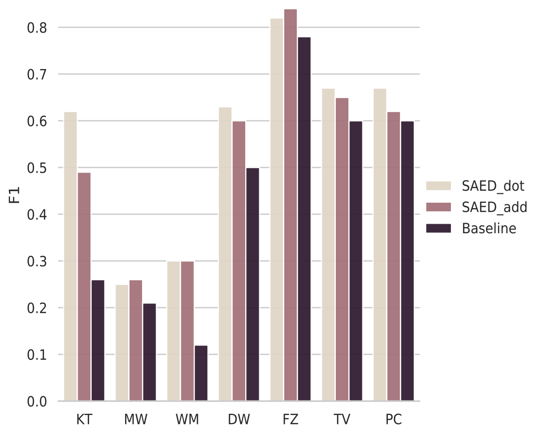
Table 4.6.6: Classification and Estimation Accuracy of the SAED in comparison to the WGRU and the Seq2Point. Seen and Unseen houses are noted as *S* and *U* correspondingly.

Device	Model	F1	RETE	MAE
KT	WGRU	0.66	0.07	9.04
	Seq2Point	0.54	0.1	8.14
	SAED-dot	0.33	0.18	7.25
	SAED-add	0.56	0.19	7.55

Table 4.6.7: Knowledge Transferability Comparison for Kettle(KT), on UK-DALE House 1 data.



F1 scores on same house (Cat 1).



F1 scores on different houses (Cat 2).

Figure 4.6.3: Performance comparison of SAED models to a Baseline model.

than WGRU. In terms of device disaggregation, SAED performs better on complicated devices than on devices with simple time series, albeit this is not the case while disaggregating television. Experiments with a broader spectrum of target devices may yield further information on this topic. Furthermore, as evidenced by the outcomes of the tests in categories 3 and 4, the suggested architecture has high generalization capabilities. Surprisingly, in circumstances where data is scarce, SAED models perform better than WGRU in the majority of cases. Because the suggested architecture is faster in training and inference than WGRU, SAED deployment on embedded systems is more realistic.

After comparing the performance of the SAED technique and Seq2Point, some interesting findings are reached. In many of the experiments, SAED models perform as well as Seq2Point. SAED's capacity to generalize on out-of-distribution data is one of its strengths. Seq2Point gets almost the same training times per epoch as SAED-dot (the faster of the two SAED models), but Seq2Point obtains faster inference times. SAED, on the other hand, is substantially smaller in terms of model size. The answer can be found in the structural differences between convolutional and recurrent neural networks.

In a nutshell, by incorporating the Attention mechanism into lightweight ANN architectures, we are able to create models that are both rapid to train and have a high degree of generalization. As a result, Attention may be a highly effective tool for disaggregating energy using neural network topologies. Attention could be combined with convolutional layers instead of recurrent ones to get even faster training and inference times.

4.7 Neural Fourier Energy Disaggregation

Modern NILM systems are based on deep learning, in which the complete energy consumption of a house is fed into one neural network, and the objective is the energy consumption of a single appliance. Recognizing multiple appliances with a single model has also caught the attention of numerous researchers. Typically, multi-label techniques identify the on/off states of a preset number of appliances (Tabatabaei et al., 2016; Nalmpantis and Vrakas, 2020). The single regression approach is the subject of this research work, with the goal of developing a computationally efficient energy disaggregator.

Apart from the computational difficulty of the disaggregation problem, the experimental setting is influenced by a number of characteristics. These include differences among datasets, the frequency with which energy data is sampled, the time period over which a forecast happens, and the number of active devices. Reproducibility of NILM experiments is difficult due to the intricacy of the environmental setup. To address the issue of comparability, Symeonidis et al. (2019) present a benchmark framework for describing various scenarios for NILM algorithm testing. Batra et al. (2019b) attempt to address reproducibility concerns by implementing nine distinct disaggregation techniques and giving state-of-the-art experimental results. Despite these attempts, a widely agreed standard for comparing NILM systems remains elusive. (Klemenjak et al., 2020).

The contribution of this research is threefold. The first contribution is the design of a unique architecture, entitled Neural Fourier Energy Disaggregator, that integrates the Fourier transform (NFED). It is inspired by FNet (Lee-Thorp et al., 2021), where Fourier transform is used as a faster alternative to attention mechanism. The second contribution is an ablation research that compares two suggested neural network variants. One variant makes use of the Fourier transform, while the other one makes use of the attention mechanism. The final contribution is a comprehensive comparison analysis aimed at determining the optimal model for each appliance using a sophisticated tuning process that takes both experimental and architectural hyper-parameters into consideration. The models that are compared are NFED, window-GRU (WGRU) (Krystalakos et al., 2018), sequence-to-point (S2P) (Zhang et al., 2018a) and self-attentive energy disaggregator (SAED) (Gkalinikis et al., 2020). To ensure a fair comparison, the optimal environmental configuration for each of the four models is determined. Then, using the benchmark framework of Symeonidis et al. (2019) as a guide, it is shown that the suggested model provides outcomes that are near to those obtained by state-of-the-art models while remaining computationally efficient, using fewer learning parameters, and requiring relatively low storage space.

The experiments of this work are based on three public datasets: UK-DALE (**?**), REDD (Kolter and Johnson, 2011) and REFIT (Firth et al., 2017). UK-DALE and REFIT include data from the Uk, while REDD contains data from the USA. REFIT consists of twenty buildings and a broader range of electronics. Five common household appliances are used to test disaggregation models: the dish washer (DW), the refrigerator (FZ), the kettle (KT), the microwave (MW), and the washing machine (WM).

Preprocessing is trivial since neural networks accept unprocessed data as input. It is critical to align the input and goal dates and times. Additionally, the datasets may have some missing values that are substituted with zeros. Preprocessing begins with standardizing the data using the following formula:

$$Z = \frac{x - \mu}{\sigma} \tag{4.10}$$

, where Z is the standard score, x the observations, μ the mean of the sample and σ the standard deviation. The standardization of the target appliance is modified appropriately based on its energy consumption statistics.

The experiments in this work are divided into four stages, beginning with the development of the suggested model and ending with its comparison to current models. The four experimental stages are summarized in table 5.3.2. After establishing the architecture of NFED, the first step is to tweak its hyper-parameters and determine the optimal depth and number of neurons. The experiments employ five cross validations to assess NFED variance on UK-DALE house 1. When two distinct versions of the model are equivalent in terms of performance, the less computationally intensive version is preferred.

The second step is to tweak the environment settings for each model individually. The most critical parameter is the length of the input. The optimal window length for each model is determined by a series of trials using the five-fold cross validation technique on each target appliance. The final configuration of window lengths for each appliance type is shown in table 4.7.2. Figure 4.7.1 demonstrates an instance of these experiment. $F1$ score represents the average of the five cross validation cycles. As illustrated in the image, the lightweight versions SAED and NFED perform better when used with small windows, such as those found in the case of a washing machine. The larger models S2P and WGRU exhibit a decrease in

Experiment	Environment setup	Goal
Hyper parameter tuning of the proposed architecture (NFED).	5 CV on house 1 from UK-DALE.	To select the best hyper parameters of NFED considering the number of the neurons and the depth of the network.
Tuning of input length per appliance for each model.	5 CV on house 1 from UK-DALE.	To find which window length achieves the best performance for each model, given a target appliance.
Ablation study comparing Fourier transform and self-attention mechanism.	Follow the four categories of experiments of Symeonidis et al. (2019).	To compare the effectiveness of Fourier transform within the proposed neural architecture against the mechanism of attention.
Model evaluation using a benchmark framework.	Follow the four categories of experiments of Symeonidis et al. (2019)	To evaluate and compare the performance of the proposed model against the baselines.

Table 4.7.1: Summary of experiments

	Microwave	**Kettle**	**Fridge**	**Washing Machine**	**Dish Washer**
NFED	50	50	350	150	450
S2P	100	300	400	400	500
WGRU	100	150	450	150	350
SAED	100	50	250	50	200

Table 4.7.2: A map of the models under evaluation with the respective best window length for each appliance.

performance as the window size is increased, but then increase again after window length 350. The greatest window size that is attempted is 500 samples, or 50 minutes.

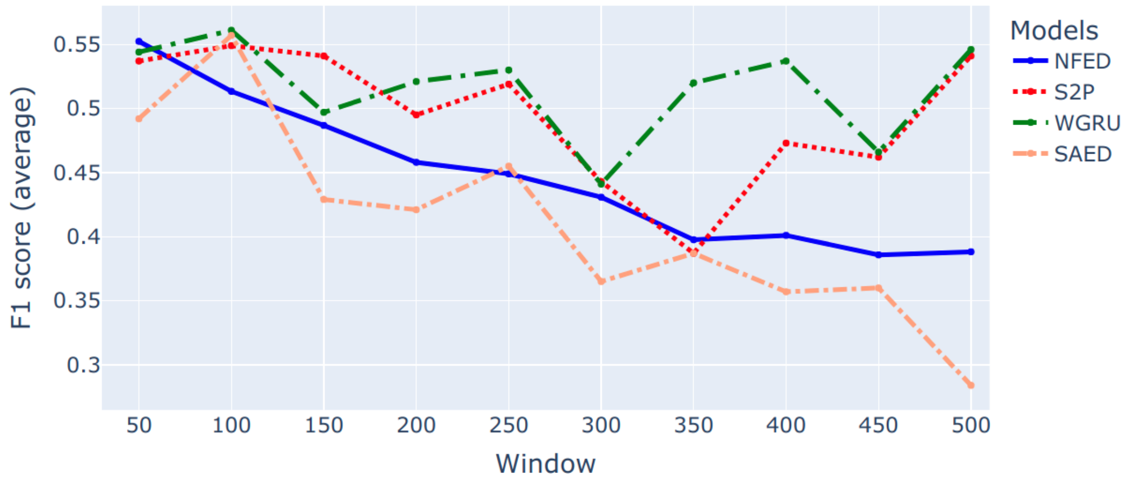


Figure 4.7.1: An example of how the performance of various models is affected by the input length. The target appliance is a washing machine and the evaluation metric is F1 score.

The final two parts of our methodology involve applying the benchmark framework to two variants of the proposed neural network and comparing NFED to three additional models. The ablation study attempts to demonstrate the advantages of applying the Fourier transformation in place of the attention mechanism. The primary advantage of the Fourier transform is that it reduces computing complexity, speeds up inference, and reduces the size of a trained model.

In terms of evaluating and comparing the proposed model to current ones, the approach adheres to the specified benchmark structure by Symeonidis et al. (2019) and includes four basic scenarios. In the first scenario, models are trained and evaluated on the same residence over a period of time. The test data follows the training data chronologically. As a result, little or no shift in distribution is expected. Models that do poorly in these experiments are considered weak, as this is the simplest evaluation scenario. In the second situation, a data distribution shift is expected, as test data originates from different houses that are not seen during training. The varied patterns of energy usage can be linked to the people' habits and the range of equipment. The third and fourth scenarios examine the models' ability to learn across many buildings and to perform testing on the same and distinct datasets. The benchmark is divided into four categories: single building NILM, single building learning and generalization on the same dataset, multi building learning and generalization on the same dataset and generalization on a different dataset. Table 4.7.3 describes the datasets and associated houses that were chosen for each category of the benchmark scheme.

When evaluating the performance of a NILM system, the most frequently used metrics include $F1$ score, Relative Error in Total Energy (RETE) and Mean Absolute Error (MAE). $F1$ score corresponds to the detection of whether a specific appliance is consuming energy. It is computed using the equation 3.13 which is the harmonic mean of Precision and Recall. Precision and Recall are described in equations 3.11 and 3.12 respectively. MAE and RETE measure how much the predicted power consumption diverges from the real one. The former one is measured in Watts, whereas the latter one doesn't correspond to a physical measure. Their equations are described by 3.6 and 4.1 respectively.

The benchmark framework used in this work also involves testing on previously unseen data. The generalization loss (G-loss) metric is used to quantify the models' generalization capabilities Klemenjak et al. (2019). G-loss is defined by equation 4.5 or 4.6, depending on whether the basic metric is $F1$ or MAE. The index *u* stands for unseen and *s* for seen data. Lower G-loss is translated to better generalization. The average generalization performance is computed by the mean generalization loss (MGL) according to

Device	Category1		Category2		Category3		Category4	
	Train	Test	Train	Test	Train	Test	Train	Test
DW	1	1	1	2	1,2	5	1,2	2
FZ	1	1	1	2	1,2,4	5	1,2,4	3
KT	1	1	1	5	1,2,4	5	1,2,4	2
MW	1	1	1	2	1,2	5	1,2	1
WM	1	1	1	4	1,5	2	1,5	3

Table 4.7.3: Buildings used for train and test. In Categories 1-3, UK-DALE was used for both training and testing. In Category 4, UK-DALE was used only for training.For testing DW and KT REFIT was used, whereas REDD was used for testing FZ, MW and WM.

equation 4.7. The average $F1$ score and the average loss are also taken into account using equations 4.8 and 4.9.

Several neural architectures for the problem of NILM have been developed in the literature (Huber et al., 2021). Sadly, only few research articles include source code, many lack essential details, and some are evaluated on private datasets. To address the aforementioned reproducibility difficulties, the baseline models are chosen based on the ease with which previous experimental results can be replicated, their widespread acceptability by other NILM researchers, and the presence of implementations in open source software such as NILMTK (Batra et al., 2019b,a, 2014a). The baseline models are: a convolutional neural network named "sequence-to-point" (S2P) (Zhang et al., 2018a), a recurrent neural network named "online GRU" or "window GRU" (WGRU) (Krystalakos et al., 2018) and a neural network based on the self-attention mechanism named "self-attentive energy dissaggragator" (SAED) Gkalinikis et al. (2020). The first two models were utilized as baselines or as the foundation for developing new solutions. They are also included in the NILMTK toolkit and consist of two strong baselines. SAED is a new architecture, yet it has yielded satisfactory accuracy and is computationally efficient. It is capable of strong generalization and can be used as a basis for lightweight models with minimal learning parameters. Next, a short description of the baseline models is presented.

Sequence-to-point(S2P) is a convolutional neural network proposed by Zhang et al. (2018a). The network's original architecture accepts a sequence of size 599 as input. It is made up of five convolution layers and the non-linear activation function ReLU. A Linear activation function defines the final layer. The layers' details are represented in fig. 4.7.2.

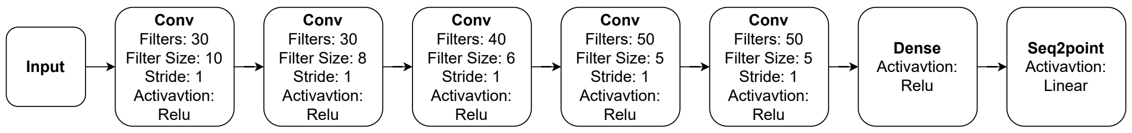


Figure 4.7.2: Architecture of S2P.

Window GRU (WGRU) is introduced by Krystalakos et al. (2018) and its main component is the recurrent layer GRU (Cho et al., 2014b). Before the output, there is a convolutional layer, two Bidirectional GRU layers, and one Dense layer. The dropout approach is used to avoid over-fitting. Srivastava et al. (2014) is used between layers. The input is a look back sliding window. Fig. 4.7.3 illustrates the architecural details.

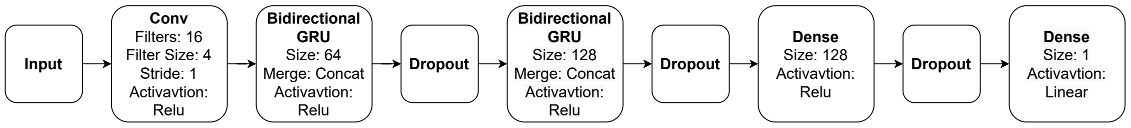


Figure 4.7.3: Architecture of WGRU.

Self-attentive energy disaggregator (SAED) is based on the mechanism of attention and is developed by Gkalinikis et al. (2020). It is a neural network that is computationally efficient. It can be trained up to x7.5 faster than WGRU and make predictions up to x6.5 quicker. A convolutional layer is followed by the attention mechanism in the architecture. The attention mechanism is divided into two types: additive attention and dot attention. Afterwards, there is a bidirectional GRU layer, followed by a dense layer. The entire architecture is depicted in Fig. 4.7.4.

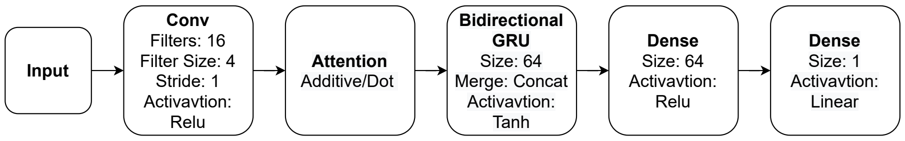
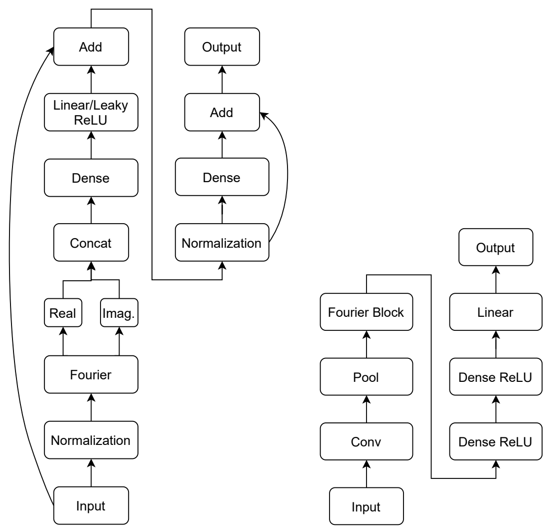


Figure 4.7.4: Architecture of SAED.

The transformer architecture (Vaswani et al., 2017) has demonstrated cutting-edge discoveries in natural language processing (NLP) and computer vision Its success can mostly be due to the attention mechanism (Bahdanau et al., 2015). Models that use the transformer architecture may comprehend the context of the given input and focus on the most important aspects. Because of parallelization, the attention mechanism outperforms recurrent neural networks in terms of computational performance. Researchers have attempted to increase the performance of attention in order to create faster transformer designs (Choromanski et al., 2020; Katharopoulos et al., 2020; Shen et al., 2018; Kitaev et al., 2020). Recently, the Fourier transform has been considered as a replacement for the attention mechanism within the transformer design (Lee-Thorp et al., 2021).The latter design is known as FNet, and its key advantage is that the Fourier transform has no learning parameters.

Neural Fourier Energy Disaggregation (NFED) is a unique neural architecture proposed in this research. To the best of the authors' knowledge, this is the first time a Fourier-based neural network has been proposed for the NILM problem. The network's fundamental component is an architecture known as the Fourier block, which is depicted in figure. 4.7.5a. The block's input is a normalized tensor. After that, the Fourier transform is used. The real and imagined components pass through a dense layer. The activation function can be either linear or leaky relu. It has been observed that leaky relu improves the performance for certain appliances, like dish washer. Following that, the original input is combined with the dense layer's output via a residual connector. There is another layer of normalization, followed by a layer of linear dense one. The dense layer's input is appended as a residual connection to its output, resulting in the block's final output. The end-to-end architecture of NFED is depicted in figure 4.7.5b.It starts with a convolutional layer and then includes a 1D power-average pooling technique. Following that is a Fourier block, the output of which is routed through two non linear dense layers with a relu activation function. Finally, a linear layer provides the network's output.

As previously stated, a second version of NFED is being built based on self-attention. In current neural architectures such as Transformers, the attention mechanism is particularly common (**?**). However, its computing complexity necessitated the search for alternatives. The proposed scheme is used as a case study in this paper to see if the Fourier transform can substitute attention for the NILM problem. As a result, a



(a) Fourier block. (b) NFED architecture.

Figure 4.7.5: The proposed NFED neural network and the Fourier block architecture.

second version of NFED is developed in which the Fourier transform is replaced with self-attention. In the context of an ablation investigation, more details concerning the comparison of the two versions of NFED are discussed later in this study.

There is a specified input length for each pair of appliance and model that optimizes performance. The architectural design, as well as the input length, influence the attributes of the models. A long input means a longer training and inference time for a recurrent neural network. A huge input would effect the number of parameters in a fully connected network, and consequently performance and speed.

Table 4.7.4 contains information about the final models created for five devices as a result of this research. The proposed architecture has the advantage of NFED performing well with a relatively small window size for most appliances. For example, the window for kettle and microwave contains 50 values, which is the smallest window. In general, the large models, WGRU and S2P, perform better with larger windows. This is typically more than double the NFED or SAED window. Except for WGRU, all models are affected by the learning parameters, which rise as the window size grows. Regardless of the window, WGRU keeps the same number of learning parameters. WGRU, on the other hand, suffers greatly in terms of training and inference speed because it processes the input data consecutively. SAED addresses this issue because it just has one recurrent layer and the inference speed is not adversely affected. SAED and S2P are the fastest models in terms of average training speed. NFED is close to the other two and can be faster at times. This appears to be counterintuitive at first because S2P has far more learning parameters. One of the underlying reasons behind this, is that most of its layers are convolutional, which are computed relatively quickly in modern GPUs, and the depth is slightly less than NFED.

The size of the model and the inference time on a CPU are two important properties of neural networks, particularly when deployed on the edge. Because of the restricted resources of edge devices, these two features are critical. The size and inference speed of the models when run on a CPU are listed in the table 4.7.4. SAED, which is less than a 0.5MB in size, is the smallest model. WGRU requires 2.794MB irrespective of the target device that is detected. For instances with a small window, such as a kettle or microwave, NFED requires 1.8MB. In the case of dishwasher, it can consume up to 17.336MB. Finally, S2P is the largest model, with sizes ranging from around 20MB to 102MB. The two smallest variants are appropriate for deployment on resource - constrained devices, but we should also consider their shortcomings. SAED sacrifices a lot of performance, while WGRU can be exceedingly slow due to the recurrent units. If speed is not an issue, WGRU is a viable option because its performance is comparable to that of its competitors. If storage, efficiency, and performance are critical, the NFED model is the best choice. It can be up to x34 less in size than S2P without sacrificing performance, and it has low latency when operated on a CPU.

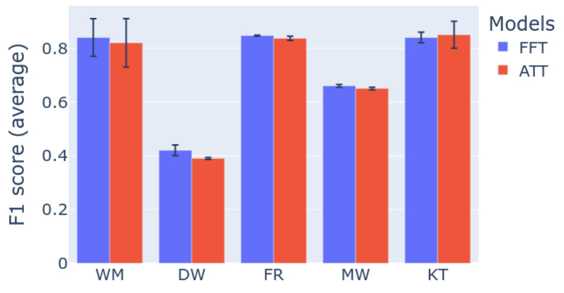
Device	Model	Window	Params	Size(MB)	Train(it/s)	Inference GPU(it/s)	Inference CPU(it/s)
DW	NFED	450	4.3M	17.336	32.14	92.36	51.44
	S2P	500	25.6M	102.597	34.93	120.03	76.10
	WGRU	350	698K	2.794	10.17	32.38	19.90
	SAED	200	**119K**	**0.480**	**36.72**	**163.29**	**103.76**
WM	NFED	150	1.4M	5.444	42.63	119.25	115.13
	S2P	400	20.5M	82.117	24.34	87.96	87.21
	WGRU	150	698K	2.794	14.60	45.87	45.64
	SAED	50	**44.9K**	**0.180**	**66.04**	**270.48**	**269.63**
FZ	NFED	350	3.3M	13.212	23.69	64.39	65.19
	S2P	400	20.5M	82.117	**40.89**	**139.25**	**87.27**
	WGRU	450	698K	2.794	8.02	25.01	15.39
	SAED	250	**164K**	**0.660**	37.79	**141.19**	85.19
KT	NFED	50	449K	1.8	**76.12**	**181.70**	193.77
	S2P	300	15.4M	61.637	31.86	114.12	113.30
	WGRU	150	698K	2.794	14.47	47.31	47.24
	SAED	50	**44.9K**	**0.180**	64.97	291.71	**286.95**
MW	NFED	50	449K	1.8	75.71	174.99	**172.71**
	S2P	100	5.21M	20.677	**78.31**	**206.68**	162.30
	WGRU	100	698K	2.794	21.71	66.58	66.42
	SAED	100	**59.9K**	**0.240**	56.05	179.75	150.87

Table 4.7.4: Properties of the tested models for each appliance. Number of parameters, size of the model per device, training speed (GPU), inference speed (GPU and CPU).

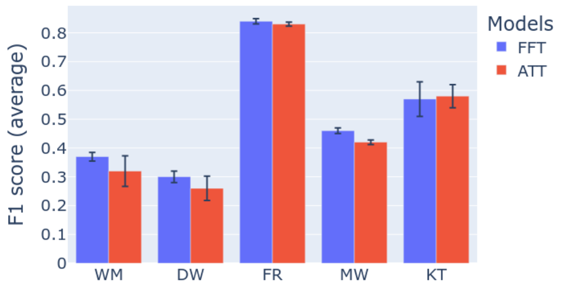
Three categories of experiments are provided in this study. An ablation analysis is carried out to investigate two distinct versions of the designed system and to investigate the advantages of Fourier transformation over self-attention. Following that, there is a thorough comparison of the performance of four neural nets on the topic of energy disaggregation. Finally, an analysis of the experimental observations explains which neural network should be considered for various case studies.

One of the objectives of this study is to emphasize the differences in performance and processing needs between the Fourier transform and the self-attention mechanism when used in the suggested neural architecture. The comparison is carried out in the first 2 groups of the benchmark approach using $F1$ score. The two network variants are also compared in terms of storage space, train and inference speed.

As demonstrated in Table 4.7.5, the attention variant model (ATT) takes up more memory than the proposed model (FFT) and is slower than the Fourier based model. In terms of the performance, Figure 4.7.6 briefly presents the comparison results. Overall, the suggested NFED model outperforms the attention variation and has a lower standard deviation for most devices. Hence, FFT appears to surpass the Self-Attention mechanism in the setting of the innovative deep learning model. Surprisingly, the performance



(a) Category 1: Single building NILM.



(b) Category 2: Single building learning and generalization on the same dataset.

Figure 4.7.6: Benchmark results for the ablation study.

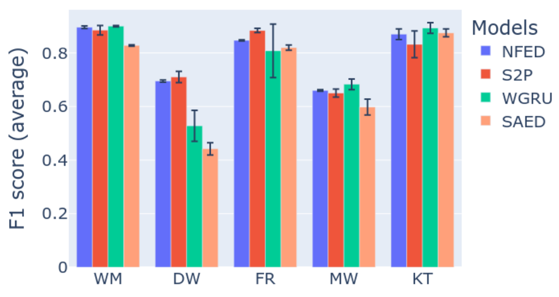
gap is more apparent in Category 2, indicating FFT's good generalization capabilities.

Device	Model	Params	Size(MB)	Train GPU(it/s)	Inference GPU(it/s)
	FFT	4.3M	17.336	32.14	92.36
DW	ATT	4.7M	18.956	27.92	73.43
	FFT	1.4M	5.444	42.63	119.25
WM	ATT	1.4M	5.624	39.63	105.21
	FFT	3.3M	13.212	23.69	64.39
FZ	ATT	3.5M	14.192	37.38	97.14
	FFT	449K	1.8	76.12	181.70
KT	ATT	454K	1.820	70.20	174.08
	FFT	449K	1.8	75.71	174.99
MW	ATT	454K	1.820	71.10	168.50

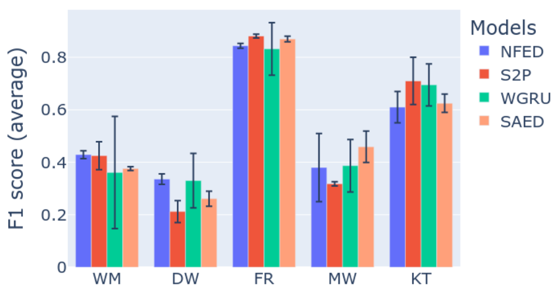
Table 4.7.5: Properties of the ablation study models for each appliance. Number of parameters, size of the model per device, training speed (GPU), inference speed (GPU).

S2P and WGRU, two powerful baseline models, are used to evaluate and compare NFED. Both of these models have a high $F1$ score and a low MAE. The downsides are that S2P contains a large number of parameters, resulting in a rather big trained model. WGRU has few parameters, yet it is slow since it is mostly made up of recurrent units that do sequential operations rather than concurrent ones. A third baseline model, SAED, is a weaker disaggregator but very lightweight and provides good generalization performance thanks to the attention mechanism. The four models are tested and compared for the following appliances using the benchmark methodology mentioned in previous sections: dishwasher, washing machine, fridge, kettle, and microwave. The evaluation metrics are $F1$ score and MAE. Figure 4.7.7 shows the $F1$ score findings. The results for MAE are comparable, and they are shown in figure 4.7.9.

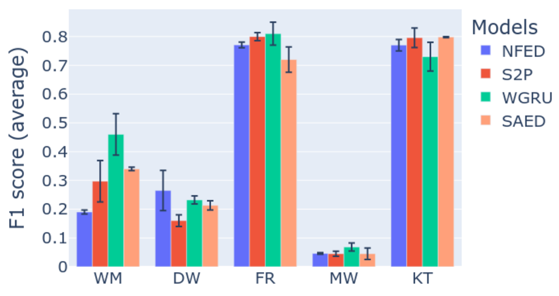
Beginning with the first group of experiments, which evaluates the models on unseen future data from a single building, the proposed model gets the highest or second highest $F1$ score for all devices. From figure 4.7.7a It is clear that NFED not only performs consistently, but also has the smallest standard deviation across many repeats of the same experiment. As shown in figure 4.7.7b, similar results are obtained for



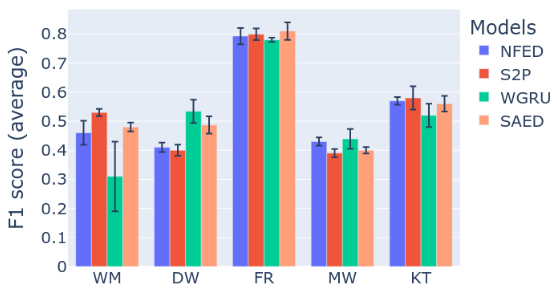
(a) Category 1: Single building NILM.



(b) Category 2: Single building learning and generalization on the same dataset.

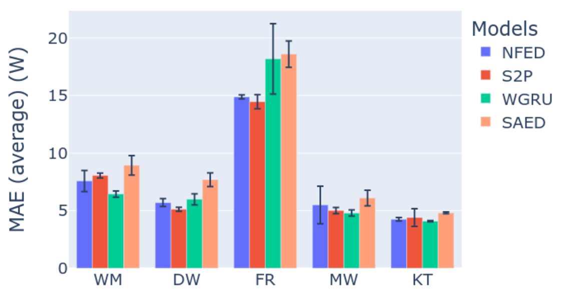


(c) Category 3: Multi building learning and generalization on the same dataset.

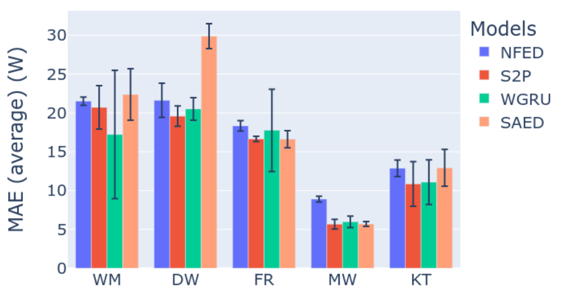


(d) Category 4: Generalization on a different dataset.

Figure 4.7.7: Benchmark results for the models NFED, S2P, WGRU and SAED.

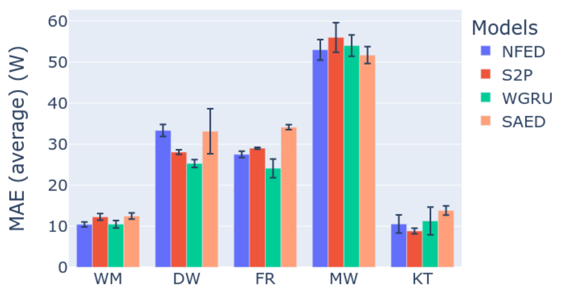


(a) Category 1: MAE results.

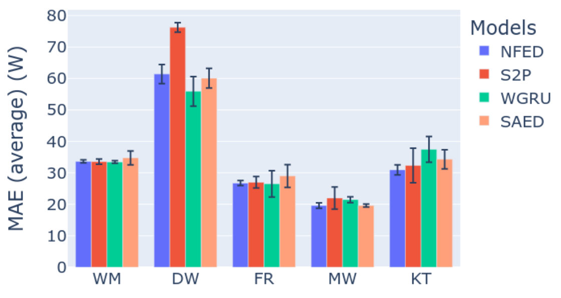


(b) Category 2: MAE results.

Figure 4.7.8: Benchmark Categories 1-2 results for the models NFED, S2P, WGRU and SAED.



(a) Category 3: MAE results.



(b) Category 4: MAE results.

Figure 4.7.9: Benchmark Categories 3-4 results for the models NFED, S2P, WGRU and SAED.

the second category of experiments, where the test data come from a different building. NFED is very competitive in these two categories of experiments, with regards to the appliances dish washer and washing machine. S2P and WGRU are next, with S2P having lower standard deviations but the worst performance in the case of dish washer on category 2. Regarding the refrigerator, all four models perform well, with S2P and NFED taking first and second place with only a slight difference. In terms of microwave, WGRU and NFED are the best models in category 1, but SAED takes the first place in category 2. This can be attributed to SAED's strong generalization competence. Regarding kettle in the first category, all models score more than 80% $F1$. On category 2, performance degrades, with S2P and WGRU performing the best, followed by NFED and SAED.

The benchmark's last two categories are the most difficult. In category 3, a model attempts to learn from many buildings, which is a difficult task because there may be more patterns to learn. The two training buildings may have a different number of appliances with varying energy consumption behaviour. Testing is then carried out on an unseen house. As a result, the model is prone to learning the similarities between the two training houses, and testing is based on these learned representations. Similarly, category 4 is more difficult because test data come from a different electricity grid. Despite the fact that category 4 is generally more difficult, the final result is heavily dependent on the actual complexity of the testing house, such as how many appliances it has. Overall, for both multi-building training categories, the proposed model outperforms or is on par with the other models. SAED exhibits strong generalization due to its low performance reduction from single building cases. Table 4.7.6 contains additional information on the models' generalization performance. Although the two strong baselines, S2P and WGRU, are competitive, neither is steadily a the best model. Because of the complexities of these tests, determining the best model for a specific appliance is difficult. From figures 4.7.7c and 4.7.7d a reasonable conclusion that may be drawn is that NFED performs at or above the baselines. The two models with the best overall performance are the NFED and the S2P.

Because of the problem's complexity, comparing NILM models is difficult. In the real world, data are out-of-distribution. This is a fundamental unsolved issue in machine learning. The benchmark framework used in this study simulates the aforementioned problem, and the results show that no model can perform similarly in out-of-distribution data. Furthermore, in the real world, it is critical to consider the model's properties such as its size and how fast it can run on various computing resources. A reasonable technique to compare different NILM models is to take into account all of the benchmark's experimental findings and the models' attributes.

Figure 4.7.11 depicts a spider diagram for the case of dish washer. It provides the $F1$ score for each of the benchmark's four categories, as well as information about the models' size, inference speed on a GPU, and inference performance on a CPU. All quantities have their optimal values in the disk's outer space. The closer you are to the center, the worse the outcome. As can be seen, NFED performs admirably in all four categories, making it an excellent contender for real-world deployment. When the other properties are considered, NFED is the third fastest model. S2P would be a suitable candidate if speef is crucial but not at the expense of accuracy. On the other hand, S2P is the largest by far, which would make it prohibitively expensive if storage space is limited. Each of the five appliances examined in this research is analyzed in the same manner and the respective spider diagrams are depicted in figures 4.7.10 - 4.7.14. To summarize, each model has a number of advantages and disadvantages and may or may not be a good fit depending on the significance of the factors and the overall system needs.

			Classification Accuracy			Estimation Accuracy		
Device	S|U	Model	F_1s	AUH	MGL[%]	MAEs	EUH[W]	MGL[%]
DW	1|2	NFED	0.69	0.34	51.7	5.7	21.62	279.2
		S2P	**0.71**	0.21	70.1	**5.12**	19.58	282.4
		WGRU	0.52	0.33	**37.5**	5.99	20.51	**242.4**
		SAED	0.44	0.26	41.0	7.69	29.88	288.7
WM	1|2	NFED	0.89	0.43	**52**	7.57	21.52	184.2
		S2P	0.88	0.42	**52**	8.05	20.71	157.3
		WGRU	**0.9**	0.36	59.9	**6.44**	17.22	167.4
		SAED	0.83	0.37	54.6	8.94	22.37	**150.3**
FZ	1|5	NFED	0.84	0.84	0.4	14.88	18.33	23.2
		S2P	**0.88**	0.88	0.3	**14.46**	16.64	15.1
		WGRU	0.81	0.83	−3.0	18.18	17.76	−2.3
		SAED	0.82	0.87	−**6.1**	18.59	16.62	−**10.7**
KT	1|2	NFED	0.87	0.61	29.9	4.25	12.87	202.8
		S2P	0.83	0.71	**14.7**	4.4	10.86	**146.7**
		WGRU	**0.89**	0.69	22.2	**4.1**	11.08	170.8
		SAED	0.87	0.62	28.6	4.81	12.93	168.8
MW	1|2	NFED	0.66	0.38	42.4	5.49	8.9	62.0
		S2P	0.65	0.32	51.1	5.01	5.67	13.2
		WGRU	**0.68**	0.39	43.3	**4.79**	5.98	24.7
		SAED	0.59	0.46	**23.2**	6.1	5.69	−**6.5**

Table 4.7.6: Comparison of Classification and Estimation Accuracy between the models. Seen and Unseen houses are noted as S and U correspondingly. In the demonstrated case, AUH equals F_1 unseen and EUH equals MAE unseen.

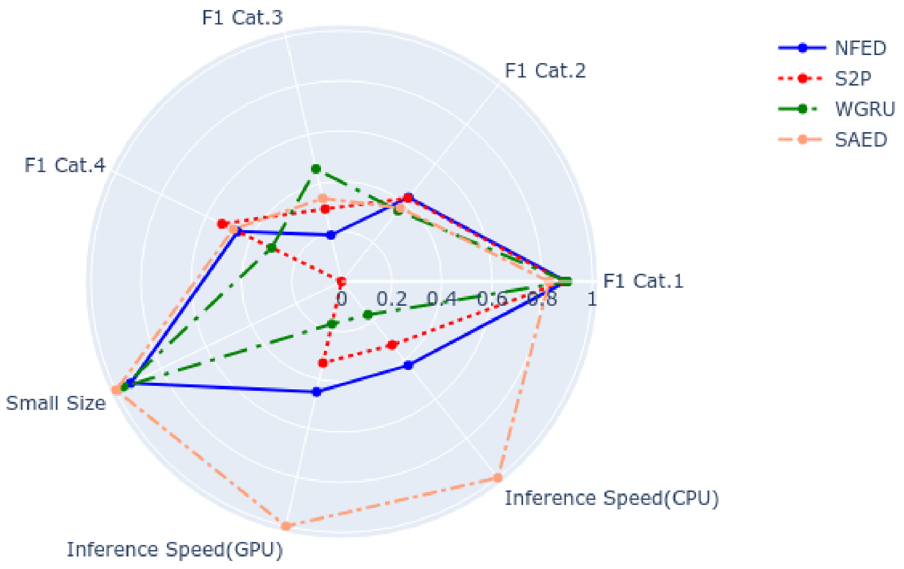


Figure 4.7.10: Diagram that summarizes the capabilities of the models to disaggregate a washing machine.

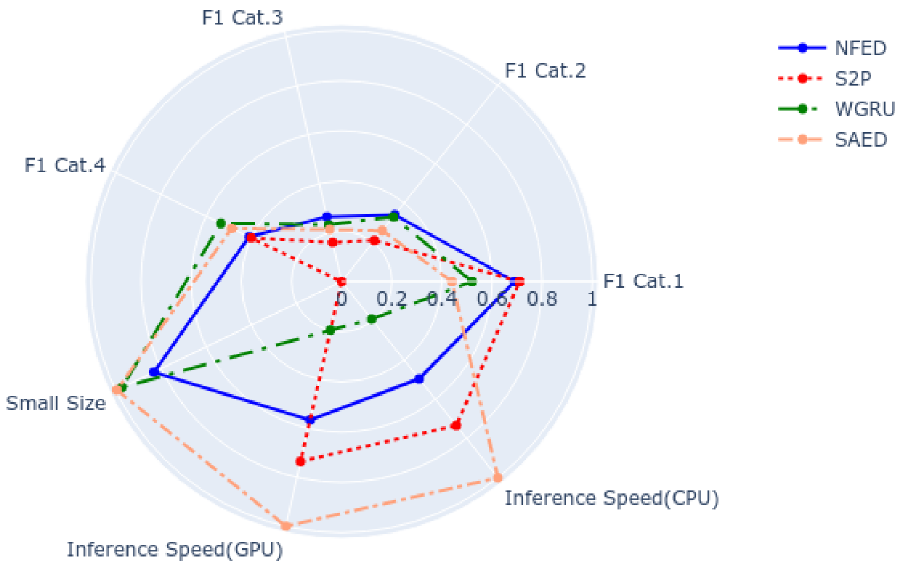


Figure 4.7.11: Diagram that summarizes the capabilities of the models to disaggregate a dish washer.

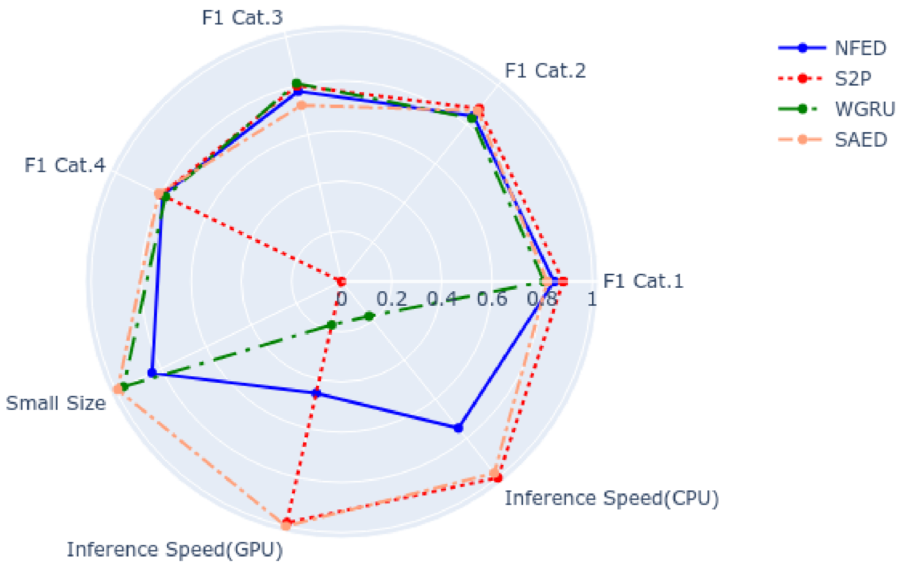


Figure 4.7.12: Diagram that summarizes the capabilities of the models to disaggregate a fridge.

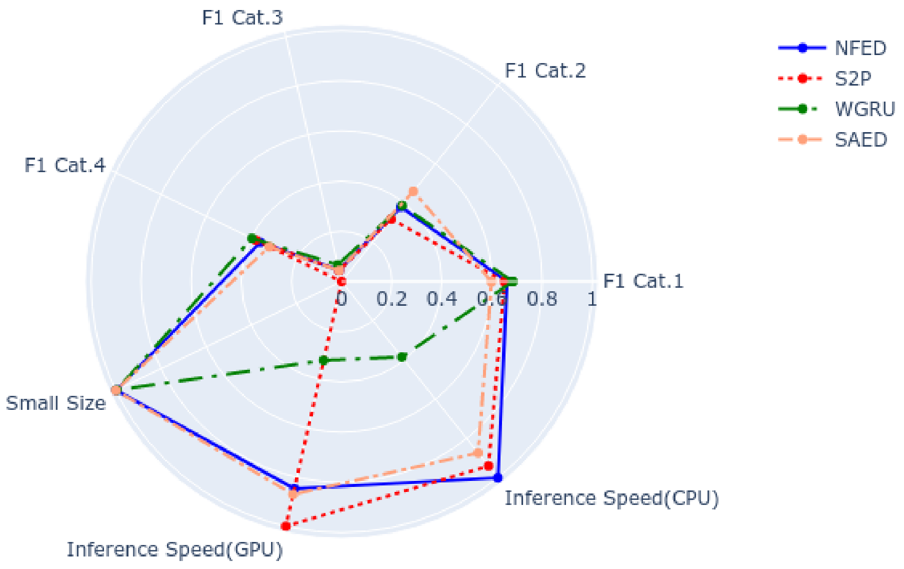


Figure 4.7.13: Diagram that summarizes the capabilities of the models to disaggregate a microwave.

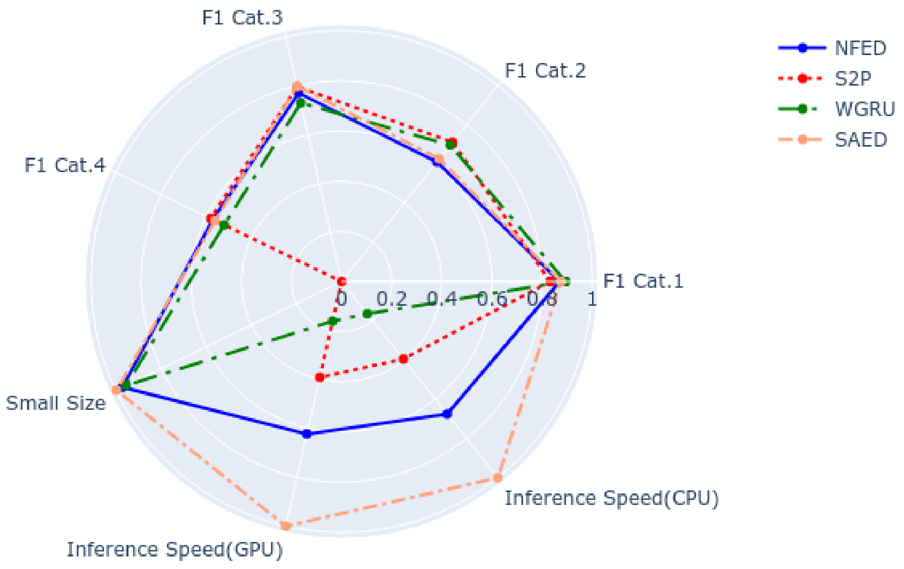


Figure 4.7.14: Diagram that summarizes the capabilities of the models to disaggregate a kettle.

It can be difficult to design a non-intrusive load monitoring system. The system requirements may differ based on the target environment in which the model will be implemented. Because of the abundance of resources, running energy disaggregation models on the cloud can be more flexible. A cloud solution, on the other hand, can be highly expensive as the system expands up. The alternative is to run such models on an embedded device, which has restricted resources. This paper introduces NFED, an unique neural network that is suitable for both solutions. NFED uses a little amount of space, has a quick inference time, and delivers comparable or higher performance outcomes. The use of the Fourier transform, which can be computed quickly and has no learning parameters, is the key to NFED's efficiency.

Fourier transformation should be employed in more architectures in the future, especially if the models are aimed at edge devices. Wavelets, in addition to Fourier, are suggested for investigation within NFED or another neural topology. The advantage of wavelet transformation is that it contains more information about time, whilst Fourier just provides information in the frequency domain. Researchers should not only run experiments on specific datasets, but also use a benchmark framework to evaluate novel models. NILM systems should be compared using individual case studies and taking into account all criteria, including model performance and attributes.

5 Time Series Embedding Representations

"It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change."
— Charles Darwin

Signal2vec Nalmpantis and Vrakas (2019b) is a novel algorithm, that maps any time-series into a vector space. It is a generalization of word2vec Mikolov et al. (2013) and extends its applicability to sequences in continuous space. The two key concepts of Signal2vec include a quantization process and the skip-gram model Mikolov et al. (2013). The latter is responsible for constructing the embedding feature space. The learning process of the embedding space is unsupervised, computationally efficient and scalable. The main advantage in comparison to other time series representation algorithms is that the embeddings are built only once and can be reused by other machine learning systems that are applied in unseen datasets. This section describes two versions of signal2vec, a supervised one and an unsupervised one. The former one is requires domain experts in order to provide the right annotations. The latter one is domain agnostic and is more generic.

5.1 Related Work

Time series is a sequence of data in time order, with values in continuous space. The order can also be irrelevant to time, but the important characteristic is that there might be features which depend on the order. This type of data has always attracted the interest of scientists in a vast range of areas such as speech recognition, finance, physics, energy, health,electroencephalography, biology etc. Some of the tasks performed in problems involving time series are: motif discovery, forecasting, source separation, subsequence matching, anomaly detection, segmentation etc. With the exponential increase of devices using sensors, the affordable access to data storage and processing power and the evolution of Internet of Things (IoT), time series play a central role.

Aghabozorgi et al. (2015) presented a review on time series, with a focus on clustering. The taxonomy of clustering time series data includes the following categories: whole time series clustering, subsequence clustering and time point clustering. These categories have in common three different approaches: shape-based, features-based and model-based. Bagnall et al. (2017) reviewed and evaluated recent algorithmic advances of time series from the perspective of classification. The suggested taxonomy includes whole series, intervals, shapelets, dictionary based, combinations and model based.

In time series problems, regardless the approach, the performance of the solution is heavily affected by the representation of the data. The categories of representations can be classified into data adaptive, non-data adaptive, model-based and data dictated. The first one includes techniques such as Adaptive Piecewise Constant Approximation (Keogh et al., 2001b), Singular Value Decomposition (Korn et al., 1997), Symbolic Natural Language (Portet et al., 2009), Symbolic Aggregate ApproXimation (Lin et al., 2007). Approaches, which belong to the second representation, are: Discrete Wavelet Transform (Chan and Fu, 1999), spectral DFT (Faloutsos et al., 1994), Piecewise Aggregate Approximation (Keogh and Pazzani, 2000) and Indexable Piecewise Linear Approximation (Chen et al., 2007a). Model based representations are based on statistics such as Markov Models and Hidden Markov Model (Minnen et al., 2007) and Auto-Regressive Moving Average (Corduas and Piccolo, 2008). Finally, the most popular data dictated approach is Clipped (Ratanamahatana et al., 2005).

With the emergence of deep learning and its unprecedented results in computer vision and other areas, research efforts, using deep neural networks in time series problems, have been increased. Yan and Yu (2015) proposed an architecture with denoising autoencoders, used in anomaly detection for gas turbine combustors. A multi-sensor anomaly detection system, which was proposed by Malhotra et al. (2016), consists of an encoder-decoder based on LSTM units. Chung et al. (2016) described a sequence-to-sequence (seq2seq) model, called Audio Word2Vec, which learns fixed length embeddings from varying size audio data. Malhotra et al. (2017) proposed a deep encoder, named TimeNet. TimeNet is an unsupervised solution and uses Gated Recurrent Units (GRUs). It is a general classification model, because it learns embeddings of time series from different domains. Gugulothu et al. (2017) proposed a seq2seq model, based on recurrent neural nets, for the problem of estimating the remaining useful life (RUL) of a system from sensor data.

Recent work shows that deep neural networks have the potential to revolutionize time series analysis. Representing time series data in a vector space is key to efficient and reusable models. Signal2vec is a general model which builds a hyperspace of embeddings. An extensive analysis is presented, examining the nature of the embeddings and how they can be used in real world problems.

5.2 Signal2Vec

Recent work shows that deep neural networks have the potential to revolutionize time series analysis. Representing time series data in a vector space is key to efficient and reusable models. Signal2vec is a general model which builds a hyperspace of embeddings. An extensive analysis is presented, examining the nature of the embeddings and how they can be used in real world problems.

Signal2vec framework consists of two main steps: tokenization and skip-gram model. The former one is a discretization process, transforming a continuous time series into tokens. The later one transforms the sequence of tokens into embeddings. In the case of a supervised tokenization, a third step is required, to create a vocabulary of tokens. Vocabulary extraction is not needed if tokenization is unsupervised. Figure 5.2.1 illustrates the steps of the framework.

5.2.1 Supervised and Domain Specific Approach

A straightforward way to tokenize a time series is to use meaningful tokens, which describe the data in natural language. It was firstly described by Nalmpantis et al. (2018) as a framework for energy data, called Energy2vec. It is a supervised approach and requires labeled data and a set of rules to automate it. Specifically for household energy time series, data are divided into windows and each window is labeled, consisting a textual description of the energy status at that specific time.

In this implementation, the size of each window is one minute. The tokens are the energy states of the appliances in the house. Due to the complexity of defining the energy status of an appliance, only three states are used. The first state is called START and describes the first time an appliance is switched on, after a period of being off. The second state is called STOP, indicating the first time the appliance is switched off, after a period of being on. Finally, the state ON, describes an appliance during operation.

In order to create the vocabulary of tokens, the following naming convention is used. The name of each appliance is concatenated with the three energy states. To set an example, the energy states of a “kettle” will

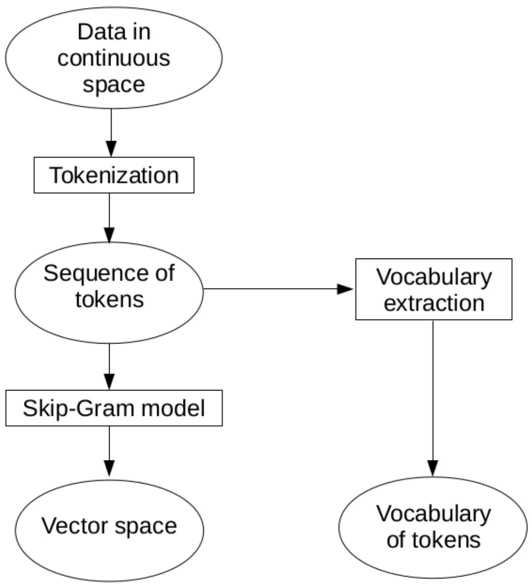


Figure 5.2.1: High level diagram of Signal2Vec framework.

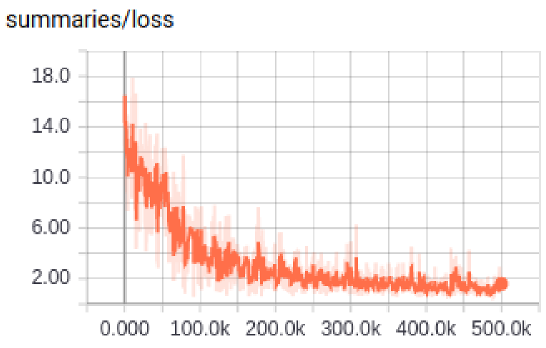


Figure 5.2.2: Loss function during training of multistate energy vectors.

be mapped to the following tokens: “kettle” when it operates, “start_kettle” when it starts and “stop_kettle” when it stops. The total number of tokens will be three times the number of devices plus two tokens named “noise” and “zero state”. As noise is defined any unknown energy consumption and zero state means that there is no energy consumption.

The outcome of tokenization is a sequence of tokens, grouped in windows of one minute length. Each window describes what is happening in the house during a specific minute. A labeled window could be the following:

[start_kettle, fridge, stop_radio, oven]

From the above example it is clear that during this specific minute, someone started the kettle, the fridge was working, someone was cooking using the oven and the radio was turned off.

Signal2vec is based on word2vec, which uses either skip-gram model or continuous bag-of-words (CBOW). Skip-gram predicts the words around the target word and CBOW predicts the target word given its neighbours. Both methods can be applied having minor differences on the results. For consistency, skip-gram is selected for all the experiments.

Following tokenization, a time series is mapped to a sequence of tokens. This sequence is now called a corpus. If the tokens are not abstract states but reflect real-world conditions of the time series, then the corpus is a human description of the total signal. The collection of the tokens consists a vocabulary. In order to apply the skip-gram model, a context is also defined as the window to the left and to the right of the target token. The object of the algorithm is to predict the context given a specific token. The architecture is a shallow neural network with one hidden layer and it is trained with pairs of tokens. One token is the target and the other one belongs to the context. Below there is an example for the case of supervised tokenization. It shows some of the training pairs of the sequence [fridge, start_oven, radio, start_kettle, stop_television], using a context of 2 tokens:

(start_oven, [fridge, radio]), (radio, [start_oven, start_kettle]), (start_kettle, [radio, stop_television])

The network trains the weights of the hidden layer from the frequencies each pairing shows up. A more formal definition of the objective function, covering both the supervised and the unsupervised versions of tokenization, is defined next.

Let tkn_1, tkn_2,...tkn_T be a sequence of T training tokens. Then the objective function tries to maximize the average log probability according to the formula:

$$1/T \sum_{t=1}^{T} \sum_{-c \leq_i \leq_c, i \neq 0} \log p(tkn_{t+i} | tkn_t) \tag{5.1}$$

, where c is the training context.

The order of the tokens in a context doesn't affect the result of the algorithm, which depends mainly on the frequency of the tokens. In word2vec model this is mentioned by Mikolov et al. (2013) as a limitation of the model, because it cannot capture rules that dictate the order of words in a sentence. In time series usually there aren't any syntactic rules and there is no difference if a token is before or after its neighbors.

The network is trained computing Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012) loss function. The optimizer is the Adagrad with learning rate 0.001. The size of each embedding is 300 and the window is 6 tokens. A summary of the loss function for the case of unsupervised tokenization is shown in Figure 5.2.2. The data come from House 1 during the year 2014.

The source of the data was the UK-DALE dataset (Kelly and Knottenbelt, 2015b), which includes both the aggregated and the individual power consumption of the appliances in a house. House 1 is selected, because it has the most devices of all the houses. The preferred programming language is Python. The tool named NILMTK (**?**) is used for accessing the database and preprocessing the energy data. In order to distribute computation to many CPU cores and a GPU, the model is developed in Tensorflow. Tensorflow comes along with a suite of visualization tools, called Tensorboard. It is used to plot diagrams of the model, visualize the embeddings and evaluate the model. Tensorboard's visualization tool uses PCA and TSNE in order to plot the embedding space in three or two dimensions. The evaluation of the embedding space is mainly done with tensorboard's similarity tool, which supports both cosine and euclidean distance.

Word2vec captures the meaning of words and their relation to each other. During supervised tokenization meaningful tokens are used and it is expected that the model captures the appliance usage pattern. In order to get a physical image of the energy behavior, diagrams depicting the frequency of each energy state, are generated. The supervised tokenization process comprises three energy states per appliance, thus there are three diagrams for each appliance. The three different states are the operation mode, starting and stopping. They correspond to three learned vectors named "appliance", "start_appliance" and "stop_appliance". Consequently, each diagram is mapped to one vector and similar diagrams implies that their respective vectors are close in a euclidean space.

The energy embeddings are trained using data from House 1 in the period 2013-2016. For a smaller or different period, a different geometrical vector space would be built. The reason is that there might be different appliances in the house or some appliances might be used during specific seasons.

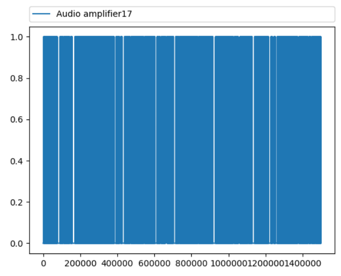


Figure 5.2.3: Frequency of operation mode of audio amplifier

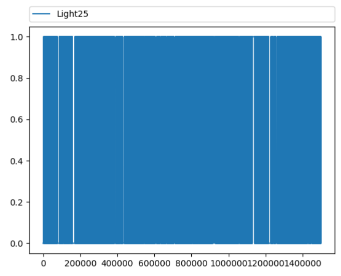


Figure 5.2.4: Frequency of operation mode of light

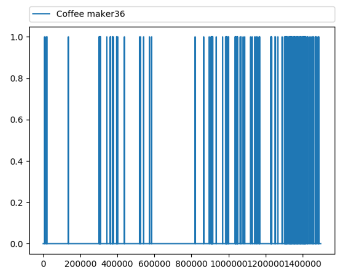


Figure 5.2.5: Frequency of operation mode of coffee maker

The trained embeddings are loaded on Tensorboard and PCA is used for dimensionality reduction and visualization in three dimensions. Next, the similarity tool is used to find vectors that are close to each other. Both Euclidean and cosine distance metrics are used, although there isn't any significant difference in the results. For each similarity search, the respective plots of the input vector and its five closest ones are compared. The results show that embeddings with small distance in the geometrical space, have similar frequency diagrams.

The frequency diagrams of "Audio amplifier" and "Light25" are shown in Figure 5.2.3 and Figure 5.2.4, respectively. From these two images it can be observed that the residents are equally using both devices in a daily basis. In contrast, Figure 5.2.5 represents a coffee maker which is used occasionally with some long periods being inactive. The same conclusion can be derived from the embedding space. Table 5.2.1 includes the similarity search of audio amplifier. The operating state of "Audio amplifier" is given as input and the five nearest points in the original space are returned. The closest embedding corresponds to the working sate of "Light25".

Table 5.2.1 summarizes the results of four different cases of neighbor embeddings in the energy vector space. The four input vectors for each search are: Television, Audio amplifier, Coffee maker and zero state. The table shows the five closest vectors along with the respective frequencies. The frequencies are presented annually to highlight any behavioral changes along time. According to the results, the expected outcomes are validated. Given that the embeddings are trained in a specific time period, energy states with similar frequency, correspond to neighbor embeddings. Furthermore, there is no evidence that the geometric energy space can capture seasonal energy behavior.

It is notable, that there are three distinct groups of appliance states in the vector space and they are correlated to the frequencies. As it can also be observed from Table 5.2.1, a group of appliances is very rarely or never used and is similar to the zero state. Another group of appliances is used almost daily or is always on, e.g. television, laptop computer, fridge freezer etc. Finally, there is a group of appliances which

Table 5.2.1: Similarity search results of energy states and their respective frequencies

Distance	**Device States**	**2013**	**2014**	**2015**	**2016**	**Total**
0	Television7	36260	52824	35963	25015	150062
1.265	Mobile phone charger38	103710	120430	289924	0	514064
1.286	Light48	12181	16100	8659	10037	46977
1.314	Laptop computer4	58081	72577	51753	31246	213657
1.334	Light31	98461	121358	57072	0	276891
1.344	Computer monitor14	43729	47400	24802	21072	137003
0	zerostate	0	0	0	0	0
1.284	Baby monitor46	0	0	0	0	0
1.296	start_Mobile phone charger34	0	0	0	0	0
1.299	stop_Tablet computer charger27	10	9	30	6	55
1.307	stop_Wireless phone charger32	19	81	0	3	103
1.309	start_Clothes iron41	21	7	4	4	36
0	Audio amplifier17	42389	84191	49104	37641	213325
1.297	Light25	65238	167924	140093	75298	448553
1.312	Solar thermal pumping station3	82008	103636	89678	37366	312688
1.313	start_solar thermal pumping station3	3002	3082	2410	808	9302
1.318	Broadband router18	0	0	0	137150	137150
1.328	start_Fridge freezer12	7445	9535	6534	3234	26748
0	Coffee maker36	17	25	975	968	3044
1.323	stop_Computer monitor14	457	746	719	453	2375
1.323	start_Ethernet switch21	3188	3049	1410	849	8496
1.326	start_Light48	101	106	53	63	323
1.339	Light48	12181	16100	8659	10037	46977
1.340	Ethernet switch21	17892	16460	6083	16008	56443

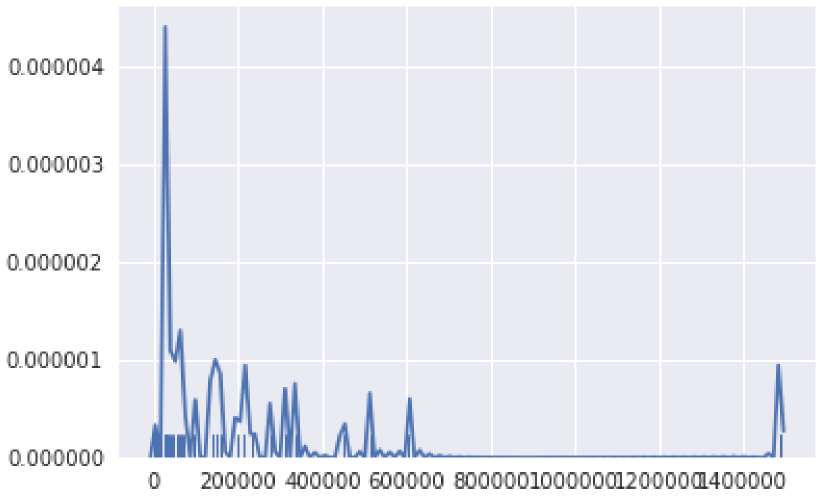


Figure 5.2.6: Frequency distribution of tokens, derived from supervised tokenization.

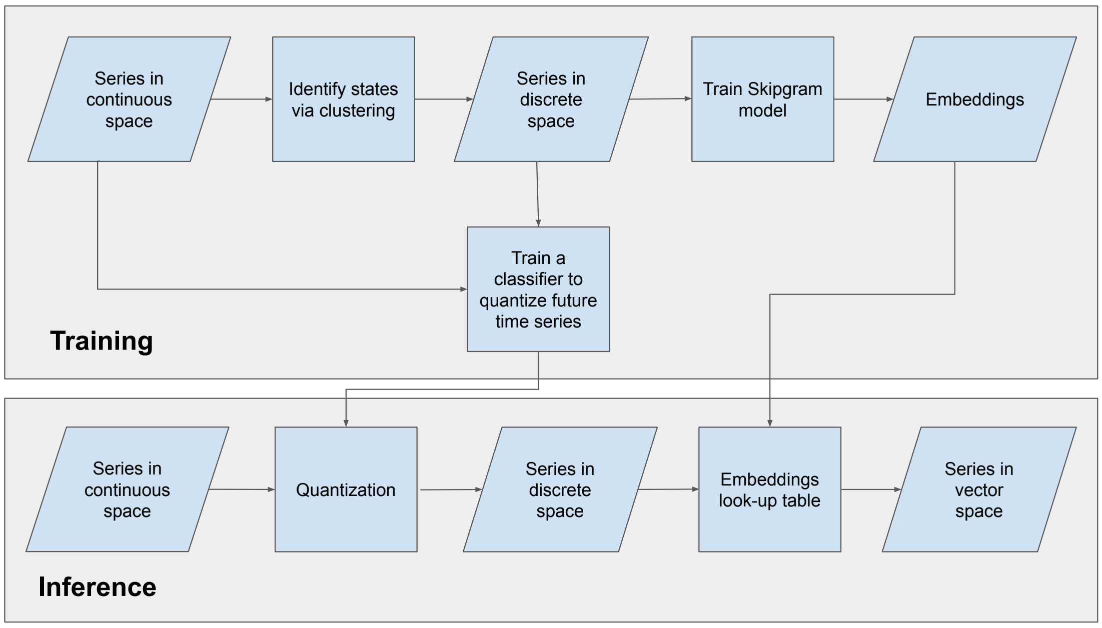


Figure 5.2.7: Illustration of Signal2Vec algorithm.

is used occasionally, e.g. coffee maker. The third group of medium frequencies is obscure, because some devices might have medium to high frequency and some medium to low. Such an example is "Light48" which is close to both "Television" and "Coffee maker". The frequency distribution of all the tokens is illustrated in Figure 5.2.6.

5.2.2 Unsupervised Domain Agnostic Approach

An illustration of the entire workflow of Signal2Vec is presented in Figure 5.2.7. The given time series is quantized via clustering and each cluster defines a token. The number of clusters is estimated using silhouette score, which tells which objects fit well in their cluster Rousseeuw (1987). The search space of the number of clusters is also constrained by a minimum and a maximum desirable number of clusters. In the domain of energy buildings the hypothetical bounds of the space can be estimated by calculating the number of possible energy states using the complexity of power draws Egarter et al. (2015c). Once the clusters have been found, a function is trained to map any time series to a discrete sequence of tokens. This function can be a classifier, like k-nearest neighbors, which is trained only once and then can be used to map any new time series to the finite space of tokens.

In the terminology of word2vec, a token is a word, the finite space of tokens is called vocabulary and a sequence of tokens is called corpus. In order to apply the skip-gram model we need to define a context.

Definition 5.2.1. *Context:* Given a sequence of tokens and a sliding window of length W, with W odd, let TKN_{target} be the middle token of the window. Then, the context consists of the rest of the tokens inside the window.

The objective of the skip-gram model is to predict the context given a specific token. The architecture is a shallow neural network with one hidden layer and is trained with pairs of tokens. One token is the target and the other one belongs to the context. The network trains the weights of the hidden layer from the frequencies each pairing shows up. The formal definition of the objective function is given below.

Let $TKN_1, TKN_2, ...TKN_n$ be a sequence of N training tokens. Then the objective function tries to

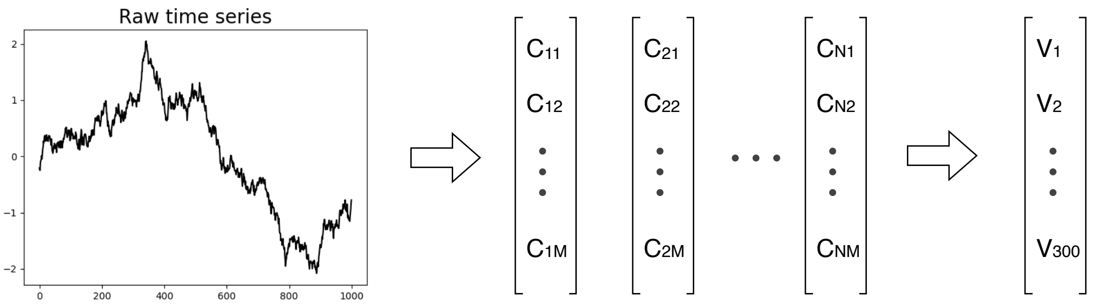


Figure 5.2.8: Mapping a time series to one vector.

maximize the average log probability according to the formula:

$$1/N \sum_{t=1}^{N} \sum_{-c \leq i \leq c, i \neq 0} \log p(TKN_{t+i}|TKN_t) \tag{5.2}$$

, where c is the training context.

The loss function of the neural network is Noise Contrastive Estimation (NCE) Gutmann and Hyvärinen (2012). Adagrad Duchi et al. (2011) is the selected gradient-based optimizer, with learning rate 0.001. The size of each embedding is 300 and the context is 6 tokens.

Definition 5.2.2. *Signal2Vec:* Let a time series S of length N, a mapping function $f : R \rightarrow T^n$ and a look up function $g : T^n \rightarrow V^{n*M}$, with T a finite set of n tokens and V an $M - dimensional$ vector space with n vectors. We define Signal2Vec as the mapping of a time series into a vector space with the following formula:

$$Signal2Vec = g \circ f : R \rightarrow V^{n*M} \tag{5.3}$$

Following the definition of Signal2Vec, the constructed vector space has as many vectors as the size of the set of tokens, with each vector having 300 dimensions. Thus, a time series is mapped to a sequence of tokens, which is mapped into a sequence of 300-dimensional vectors. Such a representation is not very helpful as it worsens the problem of dimensionality explosion. The concept of the average vector representation solves this problem.

Definition 5.2.3. *Average vector representation:* Let a time series S of length N mapped into a sequence of N vectors with $M - dimensions$ each. Let these vectors be $\hat{C}_i, i = 1, ..., N$. The average vector representation $S2V_{avg}$ of the time series S is defined by:

$$S2V_{avg} = 1/N \sum_{i=1}^{N} \hat{C}_i \tag{5.4}$$

Algorithm 1 SIGNAL2VEC TRAINING Trains a classifier for quantization and builds a vector space of the given time series.

Input: A time-series ts_data and the desired minimum and maximum states min_states and max_states respectively.

Output: A trained classifier which can be used to quantize relevant time-series and a dictionary to map the quantized values to vectors.

Data: Signal2vec training data.

```
1  $labels \leftarrow dict()$
2  $scores \leftarrow dict()$
3  $selected\_states \leftarrow 0$
4  $max\_score \leftarrow 0$
5  for $states \leftarrow min\_states$ to $max\_states$ do
6      $clustering\_model \leftarrow KMeans(n\_clusters \leftarrow states)$
7      $clustering\_model.fit(ts\_data)$
8      $labels[states] \leftarrow clustering\_model.labels$
9      $scores[states] \leftarrow silhouette\_score(ts\_data, labels)$
10     if $scores[states] > max\_score$ then
11         $max\_score \leftarrow scores[states]$
12         $selected\_states \leftarrow states$
13     end
14 end
15 $tokens \leftarrow labels[selected\_states]$
16 $classifier \leftarrow KNeighborsClassifier(n\_neighbors = selected\_states)$
17 $classifier.fit(ts\_data, tokens)$
18 $vocabulary \leftarrow set(tokens)$
19 $tokens\_sequence \leftarrow classifier.predict(ts\_data)$
20 $embeddings\_dict \leftarrow SkipGram.train(tokens\_sequence, vocabulary)$
21 return $classifier, embeddings\_dict$
```

Algorithm 2 SIGNAL2VEC INFERENCE Transforms a given time series into a vector.

Input: A time-series ts_data, a $classifier$ to quantize the given time series and a dictionary $embeddings\_dict$ to map the discrete sequence to a sequence of vectors.

Output: A vector representation $S2V_{avg}$ of the given time series.

Data: A time-series or a segment of any size.

```
22 $tokens\_sequence \leftarrow classifier.predict(ts\_data)$
23 $embeddings\_sequence \leftarrow list()$
24 foreach $token$ in $tokens\_sequence$ do
25     $embeddings\_sequence.append(embeddings\_dict[token])$
26 end
27 $S2V_{avg} \leftarrow average(embeddings\_sequence)$
28 return $S2V_{avg}$
```

Consequently a time series of length N can be represented by just one $M - dimensional$ vector, like in Figure 5.2.8. The benefit of this mapping is that the final representation is independent from the initial length N. In case a time series is very long, it can be broken into equal segments and the final vector representation will be the concatenation of the average vectors of each segment.

The complete training and inference algorithms of Signal2Vec are described by Algorithm 1 and Algorithm 2 respectively. Training Signal2Vec is fast because the neural net is very shallow and it doesn't slow inference because it happens only once. Therefore, inference time mainly depends on the inference time of the classifier as the mapping of tokens to embeddings is $O(1)$. Thus, the inference time complexity of k-NN is $O(N * S)$, where N is the length of the time series under transformation and S is the number of samples the classifier was trained. Considering that S doesn't change, we conclude that time complexity of Signal2Vec inference process is linear. Selecting a faster classifier is subject of future research and is advised for future implementations.

5.3 Applications and Experimental Results

Signal2Vec is applied to two machine learning tasks: a multi-class classification and a multi-label classification. This proposal presents two real-world experiments: appliances' classification and energy disaggregation. The first investigates the proposed framework's ability to efficiently classify signals from various appliances. The second is a straightforward method for solving a blind source separation problem, such as energy disaggregation, with Signal2vec.

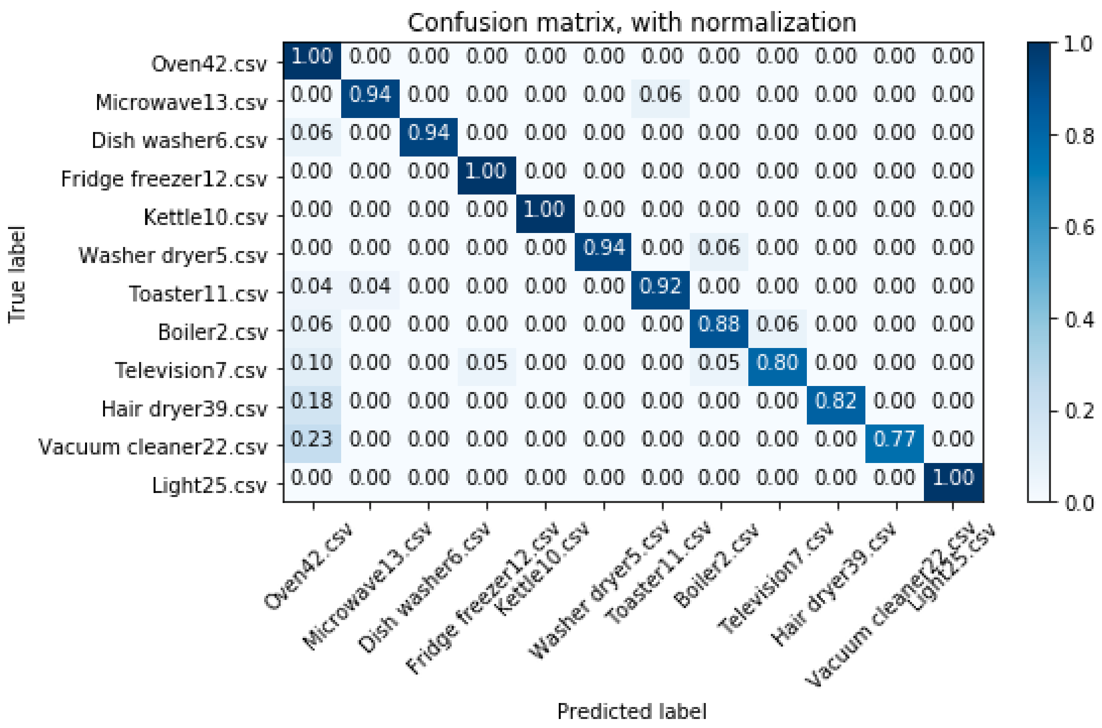
5.3.1 Multiclass Classification

The first experiment is a multiclass classification problem, identifying 12 different appliances: oven, microwave, dish washer, fridge freezer, kettle, washer dryer, toaster, boiler, television, hair dryer, vacuum cleaner and light. The classifier is a random forest with 200 estimators. The dataset of different labeled signals is created as follows. Firstly, the submetered data of 12 appliances are converted to sequences of 300 dimensional vectors, using Signal2vec. Next, the data are broken into smaller pieces with fixed length, corresponding to a specific time period. For example during 9 months with data sampled every 6s, a time period of 1 day and 12 appliances would give approximately 4188 labeled sequences of vectors. Then the average vector of each 1 day length time series is calculated. The average vector is the input to the classifier and the label is one of the 12 appliances. The metric that is used is mainly macro f1-score. The robustness of the results was validated using a k-fold cross-validation, after the initial data had been randomly shuffled. The parameter k is chosen as k=3, because for larger values the test data sample was not statistically representative of the broader dataset.

The results vary depending on the size of each time series. The smaller the time period is, the worse the results are. This can be explained because in periods smaller than a day some devices are not used at all. Indeed, for devices that are used daily, such as fridge freezer, the classifier could recognize the appliance successfully even with a time period of 4 hours with individual f1-score 0.95. The respective macro f1-score for the 12 appliances was 0.39. Consequently, the experiments for 12 appliances are meaningful for time period greater than a day. Another approach would be to have an extra label for time periods during which no device is on, known as zero state, but this is left for future work.

The framework is robust when the appliances are increased. For example using 7 day length time series for the cases of 6, 12 and 17 appliances the macro f1-score is 0.97 (+/- 0.019), 0.94 (+/- 0.015) and 0.79 (+/- 0.015) respectively. Figure 5.3.1 shows a confusion matrix summarizing the classification results of 12 appliances. It is worth mentioning that most of the misclassifications concern the oven. This is not a surprise because an oven's energy consumption heavily depends on the way it is used.

Table 5.3.1 compares Signal2vec against raw data and two other representations, SAX Lin et al. (2007) and PAA Keogh and Pazzani (2000). The results involve time series with sizes 1, 5 and 7 days, using 3 cross



True label	Oven42.csv	Microwave13.csv	Dish washer6.csv	Fridge freezer12.csv	Kettle10.csv	Washer dryer5.csv	Toaster11.csv	Boiler2.csv	Television7.csv	Hair dryer39.csv	Vacuum cleaner22.csv	Light25.csv
Oven42.csv	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Microwave13.csv	0.00	0.94	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00
Dish washer6.csv	0.06	0.00	0.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Fridge freezer12.csv	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Kettle10.csv	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Washer dryer5.csv	0.00	0.00	0.00	0.00	0.00	0.94	0.00	0.06	0.00	0.00	0.00	0.00
Toaster11.csv	0.04	0.04	0.00	0.00	0.00	0.00	0.92	0.00	0.00	0.00	0.00	0.00
Boiler2.csv	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.88	0.06	0.00	0.00	0.00
Television7.csv	0.10	0.00	0.00	0.05	0.00	0.00	0.00	0.05	0.80	0.00	0.00	0.00
Hair dryer39.csv	0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.82	0.00	0.00
Vacuum cleaner22.csv	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.77	0.00
Light25.csv	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 5.3.1: Confusion matrix of 12 classes using Signal2vec.

Table 5.3.1: Macro f1 score for 12 appliances.

TS size	Signal2vec	Raw	SAX	PAA
1 day	0.730	0.730	0.682	**0.814**
3 days	**0.878**	0.732	0.748	0.869
5 days	**0.930**	0.681	0.767	0.897
7 days	**0.942**	0.693	0.766	0.920

validation. Other classifiers have also been tested. Regardless the representation, random forest showed the best results. SAX and PAA have been tuned to get the best possible f scores. Overall Signal2vec is superior, not only giving the best results, but also because the dimension of the final representative vector is independent of the length of the time series. Signal2vec is surpassed by PAA, only for short time series such as 1 day size. A possible explanation is that temporal patterns are more important during short periods, because a device is used rarely. When calculating the average vector any temporal patterns are lost, whereas in PAA they are maintained. For future experiments more sophisticated methods can be evaluated e.g. weighted average vector. Finally, it is notable that for larger time series the results for raw data are getting worse. This is explained by the dimensionality explosion, which signal2vec faces by compressing all the information into a single vector of constant dimensions.

5.3.2 Multilabel Classification

The goal of the experiments is threefold. Firstly, to show that dimensionality reduction of energy data leads to computationally light solutions for the task of multi-label NILM. Secondly, to evaluate the effectiveness of eight different time series representations on this particular problem. Finally, the best configuration of multi-NILM framework is compared against the work of Tabatabaei et al. (2016). Due to the large number

Experiment	Environment setup	Goal
Model selection	5 CV on house 1 from UK-DALE during 1st half of 2014	To select the best parameters and the best classifier for each representation.
Evaluation	Train on house 1 from UK-DALE during 2013 and 1st half of 2014, test on house 1 of UK-DALE during 2nd half of 2014	To compare the different configurations of multi-NILM framework.
Comparison with existing systems	Train/test on UK-DALE similar to evaluation. 5 CV on houses 1 and 3 from REDD to replicate previous work	To compare the best multi-NILM configuration with previous solutions.

Table 5.3.2: Summary of experiments

of parameters of a NILM system e.g. sampling rate, different datasets, different time of the training and testing data and others, a direct comparison by using the results of other systems with different settings would not be fair. Therefore, the models proposed by Tabatabaei et al. are implemented from scratch, their experiments are replicated as close as possible and new ones are carried out for further investigation.

A synopsis of the three different types of experiments is shown in Table 5.3.2. It outlines the setup of the environment of each experiment and the goal that is pursued. More details are provided in the description of each experiment.

The energy data that are used in the experiments come from two popular datasets UK-DALE Kelly and Knottenbelt (2015b) and REDD Kolter and Johnson (2011). The latter one contains energy data from houses in the US and the former one from UK houses. For parsing the data we use NILMTK **?**. All experiments, including details of the implementation and results are available at https://github.com/ChristoferNal/multi-nilm..

The examined time series, as mentioned in previous sections are: PAA, SAX, 1d-SAX, DFT, SFA, BOSS, WEASEL and Signal2Vec. The only one that is suitable for transfer learning is Signal2Vec and for this reason we construct the embedding space only once. The vectors are produced from the aggregated energy signal of House 1 from UK-DALE dataset during the first 5 months of 2014 and with sampling rate 6s. As it is known from Word2Vec the created embeddings generalize much better when trained with as much data as possible. However, due to limited energy data and to demonstrate the transfer learning property, the training set of the embeddings excludes the rest of the UK-DALE data and all the REDD dataset.

The machine learning models that are selected for the evaluation are limited to lightweight models supporting multi-label classification. These models include decision trees, extra trees, random forests and feed-forward neural networks. The parameters of the models were selected using grid search. Among these classifiers the best ones were mostly variations of the neural network. In this series of experiments we avoid using any advanced or complex algorithms like ensembles to keep the computational complexity as low as possible. This will make the models cost efficient in a cloud solution or efficient when running on a device with limited computation capabilities.

As far as the evaluation is concerned the metrics that are used in multi label classification tasks are micro and macro f1 scores. Macro-averaging firstly measures the performance of each class and then takes the mean, whereas micro-averaging calculates the overall performance **?**. The main difference is that macro-f1

Table 5.3.3: Model selection per time-frame for Signal2Vec, BOSS, DFT and PAA

Period	Signal2Vec	BOSS	DFT	PAA
10mins	NN(1000)	NN(2000,100,100)	ExtraTrees(500)	ExtraTrees(500)
30mins	NN(1000,100)	NN(2000,100,100)	ExtraTrees(500)	ExtraTree
1h	NN(1000,100)	NN(1000,100)	ExtraTrees(1000)	NN(2000,100,100)
2h	NN(1000,100)	NN(2000,100)	ExtraTrees(1000)	NN(100,100,100)
4h	NN(1000,100)	NN(1000,100)	ExtraTrees(200)	ExtraTrees(200)
8h	NN(1000)	NN(2000,100)	ExtraTrees(2000)	ExtraTrees(200)
24h	NN(1000,100)	NN(2000,100)	ExtraTrees(2000)	NN(1000,100))

emphasizes low performance of infrequent classes. Both measures are taken into consideration, however macro-average is used as the main metric to compare different models.

The methodology of model selection comprises pairing of each of the time series representations with all the candidate classifiers, taking into account the various parameters of both representation algorithms and classifiers. Overall a few thousand different combinations are taken into consideration to avoid underrating any of the eight algorithms under evaluation. The model selection experiments are based on House 1 of UK-DALE dataset during the first half of 2014 and use a 5 cross-validation. The best machine learning models for each representation are presented in Tables 5.3.3 and 5.3.4. In these tables the numbers in the parenthesis represent the number of trees for the tree based models and number of neurons for the neural networks. A neural network can also have many layers, in which case the parenthesis includes the number of neurons per layer.

Power disaggregation depends a lot on the sampling rate or the length of the time frame that is used for inference. The sampling rate is kept stable at 6s. For the various time frames, different models and parameters are selected. The length of the time frames varies from 10 minutes to one day. Regarding Signal2Vec, the initial embeddings, that have been created at the sampling rate of 6 seconds, are not retrained. A retrain would be desirable though, as it is expected to capture different frequency features and increase the performance. The only parameter that is configurable for Signal2Vec is the number of average vectors per time-frame. It has been found that longer time-frames require more vectors. This is not a surprise as very long signals lead to summing many vectors which in turn results in obscured information. The rest of the algorithms are parameterized using grid search. For more details of the parameters please refer to the related github repository. The appliances that are used during model selection are: oven, dish washer, fridge, washer dryer, boiler, vacuum cleaner, microwave, kettle, toaster, television, hair dryer and light.

After model selection, the best machine learning models paired with the respective parameterized time series approximators are trained and tested. Training occurs in house 1 of UK-DALE dataset during March of 2013 to May of 2014. Testing takes place at the same house for the 12 appliances during the period June of 2014 to December of 2014. This is a very long period of testing, in contrast to the majority of the experiments in the literature, where NILM systems are tested for a period of a week or a couple of months. The results are presented in Figure 5.3.2. The two most efficient time series representations are BOSS and Signal2Vec, with the second one demonstrating overall the best performance. The following two models are DFT and PAA, showing quite robust results, especially when the time window is long.

It is worth mentioning that Signal2Vec is the only representation which supports transfer learning. It was trained with a time-frame equal to 6 seconds and the same vectors are used for windows with duration from 10 mins to one day. All other representations are adapted and parameterized for each different time

Table 5.3.4: Model selection per time-frame for SFA, 1d-SAX, SAX and WEASEL

Period	SFA	1d-SAX	SAX	WEASEL
10mins	NN(2000,100,100)	RandomForest(100)	NN(2000)	NN(1000)
30mins	NN(2000,100,100)	RandomForest(100)	NN(2000)	NN(1000)
1h	NN(1000,2000,100)	ExtraTrees(100)	NN(100,)	NN(1000)
2h	NN(2000,100,100)	ExtraTrees(100)	NN(100,)	NN(100)
4h	ExtraTrees(500)	NN(1000,)	NN(1000,)	NN(100)
8h	NN(100,50,100,50)	NN(100,100)	NN(2000)	NN(2000,100)
24h	ExtraTrees(1000)	NN(1000,)	NN(100,100)	NN(1000,2000)

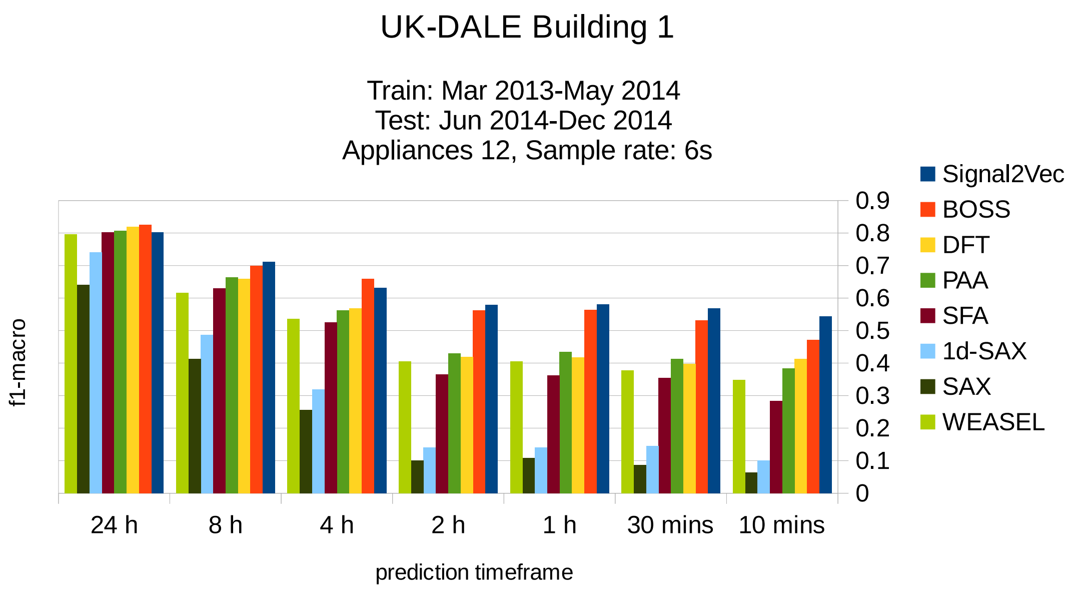


Figure 5.3.2: Appliances: oven, microwave, dish washer, fridge, kettle, washer dryer, toaster, boiler, television, hair dryer, vacuum cleaner, light.

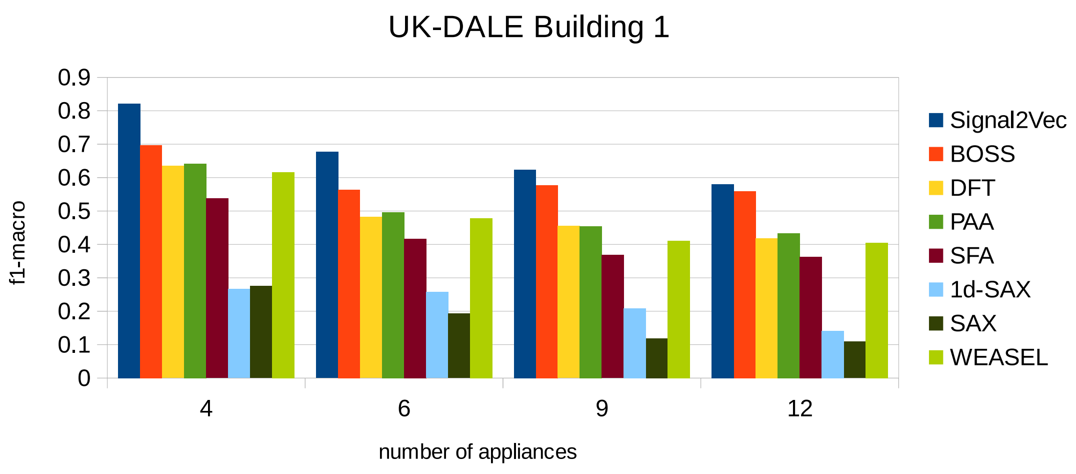


Figure 5.3.3: Macro f1 score per number of appliances, for experiments with 1 hour time frame.

frame. The only parameter of Signal2Vec is the number of average vectors per time frame which is 1 for short windows and 5 for the ones longer than 2 hours. Another notable observation is that Signal2Vec not only outperforms BOSS, but also uses a smaller neural network with almost half neurons.

Another important parameter in energy disaggregation is the number of devices that are recognized. This set of experiments include 4 to 12 different appliances. Figure 5.3.3 presents some representative results with time period one hour. The 12 appliances are the ones that are previously mentioned. The set of 9 appliances contains: microwave, dish washer, fridge, kettle, washer dryer, toaster, television, hair dryer and vacuum cleaner. The set of 6 is: oven, dish washer, fridge, microwave, kettle, and toaster. Finally the set of 4 devices includes the ones that are mostly used in NILM research papers: dish washer, fridge, microwave and kettle. Signal2vec again achieves the best f1 macro score, with BOSS following. PAA, DFT and WEASEL are very close, SFA follows showing average performance, while SAX and 1d-SAX give the worse results.

Additionally, Table 5.3.5 presents both f1 macro and micro, testing the disaggregation capabilities of two different sets of appliances of house 1 in UK-DALE dataset. One set includes high energy consumption devices such as: oven, dish washer, fridge, washer dryer, boiler and vacuum cleaner. The other set includes low energy consumption devices such as: microwave, kettle, toaster, television, hair dryer and light. As it is seen from the table, f1 macro and f1 micro most of the times are in good agreement. For the most costly appliances this time BOSS gives slightly better results than Signal2Vec, whereas for the low cost ones Signal2Vec outperforms BOSS. A possible explanation of this result could be that high energy devices are easier to be recognized as they have a strong impact on the total power signal. On the other hand low consumption devices can easily be overlapped and thus difficult to trace them in the total signal. Given that Boss is very robust to noise, we can assume that it sees the latter group of devices as noise. From the perspective of Signal2Vec, the results could be explained assuming that the embedding space has successfully learned the frequency that the two groups of devices are used, with the low consumption ones being used in a daily basis.

To conclude, three different types of experiments have been presented. Multi-NILM framework proved to be more efficient when using Signal2Vec with a shallow neural network. It exceeded all other configurations of the framework not only in performance but also in computational efficiency. Only in a few cases BOSS showed better performance. In order to enhance the significance of our results, a statistical test is also carried out. The hypothesis of the test is that the two versions of multi-NILM framework, based on

Table 5.3.5: Results disaggregating high and low cost appliances.

TS	High Cost		Low Cost	
Repr.	f1-macro	f1-micro	f1-macro	f1-micro
Signal2Vec	0.55 ± 0.004	0.77 ± 0.008	**0.61 ± 0.008**	**0.73 ± 0.012**
BOSS	**0.56 ± 0.003**	**0.79 ± 0.001**	0.57 ± 0.002	0.67 ± 0.004
PAA	0.42 ± 0.015	0.77 ± 0.010	0.44 ± 0.010	0.55 ± 0.009
DFT	0.41 ± 0.003	0.78 ± 0.001	0.44 ± 0.001	0.60 ± 0.001
WEASEL	0.42 ± 0.000	0.76 ± 0.003	0.41 ± 0.001	0.52 ± 0.004
SFA	0.35 ± 0.007	0.73 ± 0.005	0.38 ± 0.011	0.52 ± 0.014
1d-SAX	0.19 ± 0.000	0.74 ± 0.003	0.07 ± 0.004	0.22 ± 0.010
SAX	0.20 ± 0.031	0.69 ± 0.078	0.03 ± 0.053	0.12 ± 0.214

Table 5.3.6: Comparing multi-NILM against Tabatabaei's et al proposed models.

Dataset	House	RAkEL Time	MLkNN Time	RAkEL Wavelet	MLkNN Wavelet	NN Signal2Vec	NN BOSS
REDD	1	0.476	0.490	0.450	0.504	**0.518**	0.369
REDD	3	0.372	0.414	0.312	0.434	**0.446**	0.170
UK-DALE	1	0.288	0.423	0.337	0.430	**0.492**	0.439

Signal2Vec and BOSS accordingly, are equivalent, regardless the time window of the experiments. The hypothesis is tested with the Nadeau and Bengio correction to the paired Student-t test **?**, to avoid violating the key assumption of the Student's t-test that the observations in each sample are independent. In order to reject or not the null hypothesis, $alpha$ value is assumed to be 0.05. After calculating the p value equal to 0.0401, the null hypothesis can be rejected and the experimental results are statistically significant.

According to the literature, the models proposed by Tabatabaei et al. are often used as a strong baseline to evaluate state-of-the-art solutions. Unfortunately these experiments don't always take into account all the parameters of a NILM system setup and sometimes a comparison with the results of another research paper is not fair. In order to achieve an objective evaluation the experiments that are described by Tabatabaei et al. are replicated as close as possible by developing the models from scratch. Additional experiments are also conducted to strengthen the results.

Tabatabaei et al. proposal has already been described in previous section and includes two very popular ensemble methods, named MLkNN and RAkEL. Both of them are evaluated extracting features either in time domain or in frequency domain. In time domain the method that is used is time delay embeddings Kantz and Schreiber (2004). It has two parameters the time delay and the dimensions of the embeddings. The first parameter is determined using the time-delayed mutual information method. The second one is configured via the method of false nearest neighbors. In frequency domain the signal is transformed using the Haar wavelet.

Our setup is found to be very similar to the original one. Following the same methodology, the best time delay is determined to be 30 seconds and the best dimension equal to 6. These parameters apply for both REDD and UK-DALE. Apart from the techniques that are mentioned in the original paper, we also verified that these parameters give the best results by running 5 fold cross-validation experiments for different values of the parameters. With respect to wavelet coefficients, they are selected so that they retain at least 95% of

Table 5.3.7: Per appliance comparison on House 1 REDD

Appliance	RAkEL Time	MLkNN Time	RAkEL Wavelet	MLkNN Wavelet	NN Signal2Vec	NN BOSS
oven	0.0	0.03	**0.12**	0.08	0.11	0.0
refrigerator	0.27	**0.43**	0.19	**0.43**	**0.43**	0.41
microwave	0.22	0.33	0.27	0.33	**0.45**	0.21
washer dryer	0.09	0.29	0.11	0.28	**0.34**	0.04
bath GFI	0.27	0.5	0.31	0.48	**0.55**	0.24
sockets	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
light	0.39	0.43	0.4	0.41	**0.46**	0.22

the signal energy. The sampling rate is 6s for all datasets and houses and the window is set to 5 minutes. Houses 1 and 3 are selected from REDD. The appliances that are disaggregated in House 1 are: oven, refrigerator, light, microwave, bath GFI, outlet and washer. For House 3 the appliances are: electronics, furnace, washer dryer, microwave, bath GFI and kitchen outlet. House 1 from UK-DALE includes the largest variety of devices: oven, dish washer, fridge, washer dryer,boiler, vacuum cleaner, microwave, kettle, toaster, television, hair dryer and light.

The results of the experiments are summarized in Table 5.3.6 using macro f1-score as a metric. The best multi-NILM model, based on Signal2Vec and a shallow neural network, outperformed all other models in all three houses. It is very important to emphasize that the vector space that is used by Signal2Vec, for the experiments on REDD dataset, is the same one trained on UK-DALE dataset. This is evidence of the generalization capabilities of the constructed embeddings. The properties that have been learnt from the energy consumption in a house in UK, are also applicable to houses from the USA. The second best model is MLkNN using wavelets, but only for houses from REDD. On UK-DALE the second best model is the one based on BOSS. On the other hand BOSS doesn't show very consistent behaviour as it shows the worst results for house 3 from REDD. In general, RAkEL shows lower f1-score when compared to MLkNN. A more detailed sample of the experimental results is presented in Table 5.3.7. The results come from REDD house 1 and represent the f1-score of each appliance. Signal2Vec achieves the best scores for all the appliances, apart from the oven. RAkEL in frequency domain performs the best score for the oven.

More confidence to our results is added using again the Nadeau and Bengio correction to the paired Student-t test. The hypothesis now is that there is no statistically significant difference on the performance of MLkNN using wavelets and multi-NILM based on Signal2Vec, when the time window is 5 minutes. Given that $alpha$ values is equal to 0.05, p values is calculated equal to 0.023. Therefore, the null hypothesis can be rejected and the experimental results are statistically significant.

Another aspect of this comparative analysis is to show how multi-NILM framework leads to computationally more efficient models. Figure 5.3.4 illustrates how each of the models scales in terms of the dimensionality of the data. The horizontal axis shows the window that has been used to train and test the models and ranges from 1 day to 10 mins. The vertical axis refers to the inference time of each classifier in seconds. The input to the models is energy data from REDD corresponding to a period of seven days. Therefore smaller time window means more predictions within the seven days. Observing the diagram, MLkNN shows an exponential scalability on both time and frequency domains. RAkEL responds better with inference time being increased almost linearly for a period up to 30 minutes, after which it starts increasing exponentially. The two versions of multi-NILM framework perform the fastest inference, which is expected because they depend on very light classifiers and the input space is reduced by the respective

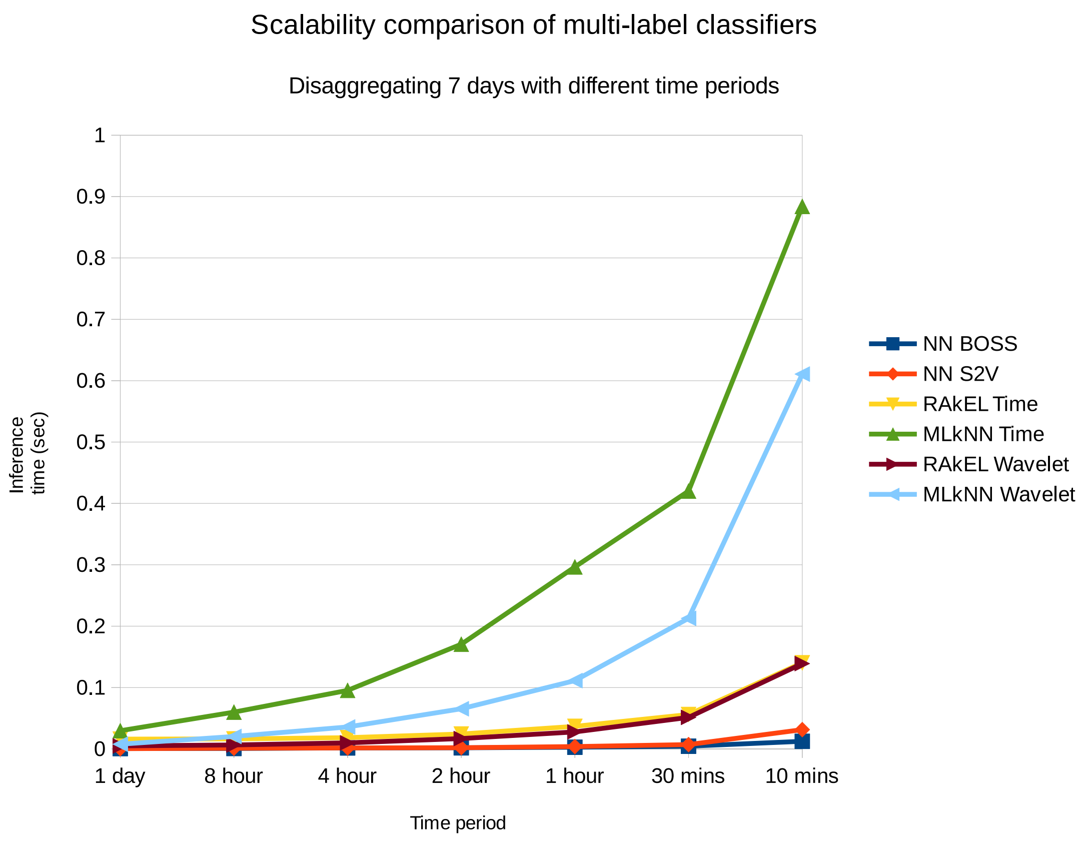


Figure 5.3.4: Scalability comparison of multi-label NILM algorithms.

time series approximator.

In a nutshell this study revisits the problem of power disaggregation through the lens of multi-label classification task. A novel framework is proposed in order to tackle the problem of dimensionality explosion, reduce the cost of a NILM system and enable a cost efficient solution. Eight popular time series representations are used for the first time in a NILM system. Signal2Vec demonstrates the most promising results and is the only model to support knowledge transfer. When compared to a strong baseline from the literature, Signal2Vec achieves the best macro-f1 score in all datasets and in a more efficient way.

Future research will focus on improving the model of Signal2Vec and increase the performance of multi-NILM framework. A key idea to improve Signal2Vec is to improve its quantization process by using an alternative method of clustering, such as Gaussian Mixture Models. The entire quantization process could also be replaced by an algorithm which transforms a time series into words like PAA and then apply the skip-gram model on top of this transformation. Regarding Multi-NILM framework, new experiments are advised to be designed using other innovative dimensionality reduction methods. More advanced multi-label classifiers, like deep neural networks, should also be considered, but always with the goal to find a balance between efficiency and accuracy.

The exponential increase of connected devices make it mandatory to look for low cost solutions that can run on low end devices. Researchers should focus on systems which apart from achieving better results, should also be practically viable.

6 Information Theoretic Principles in Deep Learning

"We are what we repeatedly do. Excellence, then, is not an act but a habit."
— Aristotle

Modern deep learning architectures frequently incorporate pooling operations, which contribute to data invariance and robustness to perplexity. Pooling is widely employed in image recognition models, but it is also utilized in speech recognition, natural language processing, signal processing, and a variety of other fields. The purpose of pooling is to reduce the size and complexity of a joint feature representation while retaining as much information as possible. The most frequently used pooling techniques are maximum pooling, average pooling, stochastic pooling, and their combinations.

The concept of feature pooling dates back to a seminal article on complex cells in the visual cortex by Hubel and Wiesel (1962). It is used in many hand-crafted feature engineering methods such as SIFT Lowe (2004) and HOG Dalal and Triggs (2005). Especially max Jarrett et al. (2009) and average pooling LeCun et al. (1990, 1998) are frequently employed in convolutional neural nets. Stochastic pooling has also shown state-of-the-art results for regularization of deep convolutional neural networks Zeiler and Fergus (2013). Graham (2014) present an alternative version of stochastic pooling that is called fractional max-pooling. The main idea behind this technique is that pooling regions don't have a predefined size but it can be chosen in a stochastic manner.

Springenberg et al. (2014) propose an All-CNN model, which consists entirely of convolutional layers, in an effort to simplify neural network topologies. Even though pooling operators are not used as independent building blocks in this architecture, they are incorporated in the convolutional layers. Recent work supports the use of pooling approaches by exhibiting state-of-the-art performance on a variety of image analysis benchmarks. The suggested architecture includes a tree based pooling function that combines max and average pooling (Lee et al., 2016). A prior study by Scherer et al. (2010) also demonstrated the performance benefit of combining more than one pooling function.

Pooling is an integral part of several neural network architectures and there is a plethora of examples predicating its benefits. However, most of the studies are empirical and there is a lack of a complete theoretical framework. The goal of this research is to shed light on the dynamics of pooling operation, using information theoretic concepts.

6.1 Theoretical Analysis of Feature Pooling

To the best of the authors' knowledge, a summary of the only previous theoretical analysis of pooling is presented, describing the statistical properties of two basic pooling operations for a two-class categorization problem. Thereafter, there is a theoretical analysis of pooling in deep neural networks from the perspective of information theory.

6.1.1 Statistical Properties of Pooling Layers

A detailed theoretical analysis of feature pooling in visual recognition is presented by Boureau et al. Boureau et al. (2010). The approach that is followed is simplified considering a two-class classification problem and assuming the features are i.i.d. Bernoulli random variables. According to the authors, the assumption of independence is invalid, since neighbouring image features are highly correlated. Despite its shortcomings, the outcomes of the study are still relevant and they are verified empirically. The two pooling operations under examination are max and average. The analysis evaluates how the statistical properties of the two methods affect the capability of a model to separate two different classes. The underlying reasons that justify this performance are obscured and are contributed to several factors, such as the sparsity of the features and the sample cardinality.

The outcomes about channel capacity of pooling operation can be used to understand the mechanism of max, average and stochastic pooling. Using proposition 6.1.1 and the statistical properties of the variables X and A, it will be explained how these operations work and in which cases one method is favorable to another.

The statistical properties of the input X are the same for each case of pooling and they are described along these lines. Let $\overline{x}$ denote the mean of the joint feature representation X_N with size N and σ_X its standard deviation. The respective formulas are:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{6.1}$$

$$\sigma_X = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2 \tag{6.2}$$

6.1.2 Understanding Pooling via the Lens of Information Theory

This section establishes a connection between pooling operator and information theory. With this in mind, the function of pooling is revisited. Although there is no formal definition, it is widely accepted that the objective is to downsample a given feature map, while retaining relevant information. Downsampling the data is easy and makes the overall architecture computationally lighter. However, determining which features are relevant is hard. It not only depends on the task, but also on the characteristics of the data.

To set an example, imagine a dataset of images, each of them illustrating either a horse or a car. The task is to separate the two classes. By defining the characteristics of a horse and a car, the task looks trivial but it will not generalize on a diversified dataset. It would require different knowledge to separate images depicting horses from a group of images depicting horses, donkeys and zebras.

Let's define the objective of pooling from the perspective of information theory.

Definition 6.1.1. Let a deep neural network with one pooling operation after hidden layer i. Denote the output features of hidden layer i as a random variable X and the output of the pooling operation as a random variable A. Then pooling channel is the information channel defined by the pooling operator.

A pooling channel can be represented by the Markov chain in Fig. 6.1.1. The mutual information between the two random variables X and A is:

$$I(X;A) = H(A) - H(A|X) \tag{6.3}$$

, where H(A) is the entropy and H(A|X) the conditional entropy. Equation (6.3) holds for both discrete and continuous random variables, because the mutual information between two random variables is the limit of

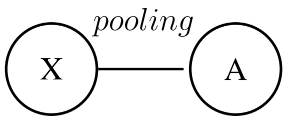


Figure 6.1.1: Markov chain of pooling operation.

the mutual information between their quantized versions Cover and Thomas (2012). Then the capacity of the channel is defined as follows Cover and Thomas (2012):

$$C = \max_{p(x)} I(X;A) \tag{6.4}$$

The ideal channel would allow all the information to be transferred, without loss. In the case of pooling, the goal is to maintain as much relevant information as possible. As it was discussed previously measuring the relevance of information would require prior knowledge of the task, otherwise our best estimation would be just a guess. Consequently the best channel is the one that makes no assumptions and this is interpreted as the channel with maximum capacity. From (6.3) it is obvious that I(X;A) is max when H(A) is maximized and H(A|X) is minimized. The following proposition outlines the behaviour of mutual information for the case of a pooling channel and will help to define its maximum capacity.

Proposition 6.1.1. *Let the random variables X and A be the input and output of a pooling channel and f the function that describes the pooling operator. The function f can be deterministic or stochastic and the mutual information $I(X;A)$ is given by:*

(i) For f being deterministic:

$$I(X;A) = H(A) \tag{6.5}$$

(ii) For f being stochastic:

$$I(X;A) = H(A) - H(\overrightarrow{r}) \tag{6.6}$$

, where $\overrightarrow{r}$ is a row of the transition matrix p(A|X), with rows representing A and columns representing X.

Proof. The first statement of proposition 6.1.1 is proved by estimating H(A|X) from (6.3), when pooling is a deterministic function. Then the conditional entropy in details is developed as follows:

$$H(A|X) = \sum_x p(x)H(A|X = x) \tag{6.7}$$

$$H(A|X) = \sum_x p(x)H(1) = 0 \tag{6.8}$$

Thereupon, for the case of a deterministic function, the mutual information depends only on the entropy of the output features and is described by (6.5).

For the second statement, where pooling is a stochastic function, let's consider the transition matrix p(A|X), with rows representing A and columns representing X. Let $\overrightarrow{r}$ be a row of the transition matrix, then (6.3) becomes (6.6). □

A less formal proof of (6.5) can be derived by using a Venn diagram. Fig. 6.1.2 depicts in details the relationship between entropies and mutual information for two random variables. Assuming pooling is a deterministic function, it can be concluded that the uncertainty of knowing the value of A given the value of X is zero. Indeed, a deterministic function describes a rule for determining the value A with probability

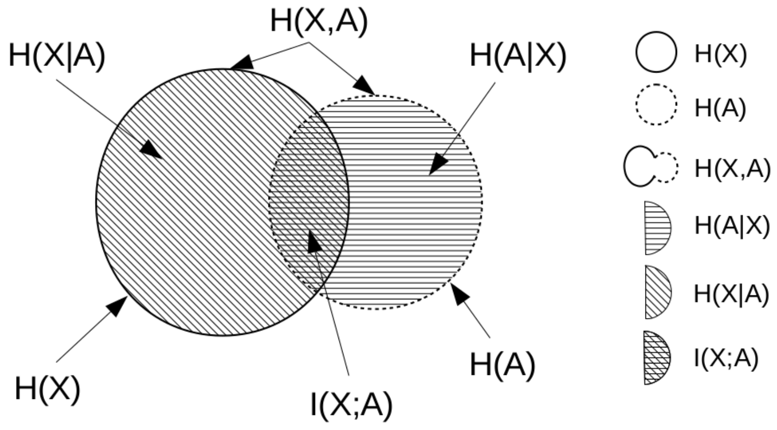


Figure 6.1.2: Relationship between entropies and mutual information for two random variables.

equal to one. With H(A|X) equal to zero, the mutual information of the two variables is equal to H(A). Then the Venn diagram should illustrate the right circle thoroughly inside the left one.

The outcomes about channel capacity of pooling operation can be used to understand the mechanism of max, average and stochastic pooling. Using the equations derived in the previous section and the statistical properties of the variables X and A, it will be explained how these operations work.

The statistical properties of the input X are the same for each case of pooling and they are described along these lines. Let $\overline{x}$ denote the mean of the joint feature representation X_N with size N and σ_X its standard deviation. The respective formulas are:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{6.9}$$

$$\sigma_X = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2 \tag{6.10}$$

Max pooling is a function that given a set of values, it chooses the largest one. Pooling is applied using kernels, so let X_r be a joint feature representation of a kernel with r elements, then the function is:

$$f_{max}(X_r) = max(x_1, x_2, ..., x_r) \tag{6.11}$$

It is a deterministic choice function and the mutual information between the input and the output, according to (6.5), is equal to H(A). Examining the statistical properties of max pooling will now clarify how it affects H(A).

Remark 6.1.1. *The capacity of max pooling channel is not affected by the max operator and is increased when the cardinality of features is high.*

Proof. By definition, it is obvious that the mean will be increased, thus $\overline{x}_A \geq \overline{x}$. In order to evaluate the standard deviation, more elaboration is needed. By using kernels , the standard deviation in (B.20) is developed as follows:

$$\sigma_X = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{r}\sum_{j=1,x_{ij}\neq a_i}^{r}(x_{ij} - \overline{x})^2 + \frac{1}{M*r}\sum_{i=1}^{M}(a_i - \overline{x})^2 \tag{6.12}$$

where M is the number of kernels, r the number of elements of each kernel and a_i the maximum value of each kernel. It is assumed that there is no overlap during pooling and so $N = r * M$. The proof with overlap is similar. Next, the standard deviation of max pooling is given by:

$$\sigma_A = \frac{1}{M}\sum_{i=1}^{M}(a_i - \overline{x}_A)^2 \tag{6.13}$$

Comparing (6.12) and (6.13), it is concluded that the change of standard deviation depends on the distribution of the values of the input features. The input also determines the cardinality of A, which also affect the entropy. From the definition of entropy, higher cardinality means higher entropy. □

Thus, max operation doesn't control the entropy of the output and there is no guarantee that it will maximize the capacity of pooling. The capacity of the channel is proved that it depends on the input feature distribution and the cardinality of the output A. This is in good agreement with the finding of Boureau et al. Boureau et al. (2010), that max pooling is well adopted to rare activated features.

Average pooling is analyzed in the same fashion, because calculating the average is a deterministic function.

$$f_{avg}(X_r) = \frac{1}{r}\sum_{j=1}^{r} x_j \tag{6.14}$$

Therefore, the mutual information is described by (6.5).

Remark 6.1.2. *The output of average pooling channel tends to be uniform.*

Proof. The mean of the original X_N and the mean of the output of pooling A_M are equal and will be symbolized as $\overline{x}$. Equation (B.20) is developed using kernels as in:

$$\sigma_X = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{r}\sum_{j=1}^{r}(x_{ij} - \overline{x})^2 \tag{6.15}$$

Using (6.14), let $a_i = f_{avg}(X_i)$. Then the formula of standard deviation for the features A_M is:

$$\sigma_A = \frac{1}{M}\sum_{i=1}^{M}(a_i - \overline{x})^2 = \frac{1}{M}\sum_{i=1}^{M}(\frac{1}{r}\sum_{j=1}^{r}x_{ij} - \overline{x})^2 = \frac{1}{M}\sum_{i=1}^{M}[\frac{1}{r}\sum_{j=1}^{r}(x_{ij} - \overline{x})]^2 \tag{6.16}$$

From (6.15) and (6.16) it is concluded that $\sigma_A \leq \sigma$. Consequently, the distribution of the output A will have smaller standard deviation and the values of its elements will tend to the expected value $\overline{x}$. These two properties of average pooling, lead to a more uniform distribution and make it robust to feature invariance. □

Following proposition 6.1.2, the distribution still depends on the input data. The maximum unique elements of the final distribution is M and it happens when each kernel gives a unique value. The mechanism of average pooling doesn't affect the number of unique output values and therefore it doesn't guarantee maximization of the entropy.

In practice, most of the times the output of average pooling would rarely be repeated. A good example is the special case of images illustrating motifs such as a chess board. With kernels of size 2x2 and assuming a chessboard pattern with values zero and one, the average value would always be the same.

The case of stochastic pooling is different, since it is a non deterministic choice function. Hence, using the second part of proposition 6.1.1, we derive the following proposition.

Proposition 6.1.2. *The capacity of stochastic pooling channel is weakly parameterised by the stochastic function.*

Proof. Without loss of generality, let the stochastic function be the equal probability of selecting one of the features. Therefore, if N is the cardinality of features, the probability of selection of each feature is $a = 1/N$. Then, replacing the entropy of the row vector in (6.6) by a function depending on the probability a we have:

$$I(X; A) = H(A) - f(a) \tag{6.17}$$

□

The mutual information of stochastic pooling depends on both H(A) and H(A|X). The first term is still free of any parameter of the channel and depends on the data. The difference against max and average pooling is that stochastic pooling has some control over the mutual information via the parameter a.

The outcome of the previous analysis is that current pooling solutions don't have the properties to handle the entropy of the output H(A) and their performance is highly affected by the distribution of the data. This problem is tackled by revisiting the problem of max entropy sampling and designing a new pooling operation based on entropy.

6.1.3 Entropy Pooling

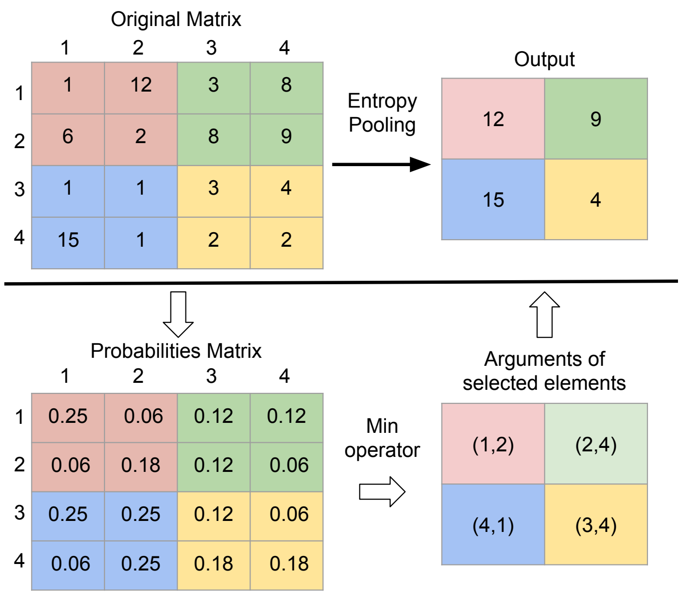


Figure 6.1.3: The process of entropy pooling. Min operator is selecting the most rare features.

As it was shown previously, conventional pooling functions don't have the properties to manage the entropy of A in (6.3). A more advanced approach is required, to accomplish complete control of the entropy H(A) and minimize the conditional entropy H(A|X). The latter requirement can be met by defining a deterministic function. For the former one, we need to understand the problem in depth and reformulate it.

Maximizing the mutual information in (6.5) can be reduced to the problem of maximum entropy sampling (Shewry and Wynn, 1987). It is defined as a design problem of selecting a subset T from a set S of N random variables, with regard to retain as much information as possible. In the context of pooling, the problem is to choose the most informative subset, subject to spatial constraints.

Following the variable definitions of the Markov chain in Fig. 6.1.1, let f_{entr} be the required function. According to the principle of maximum entropy. the probability distribution that best describes A, is the uniform distribution. Assuming only that the size of A is predefined and equal to M, the required function has lower bound equal to zero and upper bound the entropy of the equivalent uniform distribution.

$$0 \leq f_{entr}(X) \leq log|M| \tag{6.18}$$

In line with (6.18), the output of f_{entr} should approximate a uniform distribution. Existing solutions for the problem of maximum entropy sample can be adopted (El-Shaarawi and Piegorsch, 2001). However, this would be computationally inefficient for large neural networks because it is an NP-Hard problem (Ko et al., 1995). For the purpose of this study, a novel algorithm is proposed, named entropy pooling. The proposed version of entropy pooling is non-optimal, but computationally efficient and extendable.

The algorithm of entropy pooling computes the probabilities p for each feature map with size N. Next, the map of probabilities is divided into regions, according to the specified kernel size and strides, in the same way as in classic pooling operations. For each region the element with the smallest probability is selected. The process is illustrated in Fig. 6.1.3. A formal definition of entropy pooling is given below.

Definition 6.1.2. Entropy pooling is a pooling channel which operates according to the deterministic function f, described by the following formula:

$$f_{entr}(X_r) = X_r[g(P_r)], \tag{6.19}$$

$$g(P_r) =_{1 \leq i \leq r} p_i \tag{6.20}$$

, where X_r is the input feature map, P_r the constructed map of probabilities of the features and r the size of a region. The capacity of entropy pooling is given by:

$$I(X; A) = H(f_{entr}(X_r)) \tag{6.21}$$

So far, the desired function is deterministic and selects features with high sparsity, handling the amount of information that will be propagated via the neural network. It remains to explain why choosing sparse features increases the entropy. The intuition is that rare features cannot be selected several times and as a result the output will have a flatten distribution with high cardinality.

A more rigorous explanation is given considering the bounds in (6.18), the property that entropy is concave and the fact that the final feature map A has no element with $p < 1/M$. Observing the graph of a random entropy function such as Fig. 6.1.4, the peak is at $p = 1/M$ and every solution to the right has lower entropy. As a conclusion, selecting features that are activated rarely, increases the output entropy of pooling.

The theoretical outcomes are validated by a series of experiments. Entropy pooling is used in various architectures, showing how entropy affects the performance of a neural network. A direct comparison against max and average pooling shows that the new pooling operation is robust and gives better or comparable results.

The aim of the experiments is twofold. The first goal is to validate the theoretical outcomes and the second one to demonstrate that entropy pooling is robust and can perform on par with other pooling operations. For the purpose of the experiments, two versions of entropy pooling are used. One which gives an output with high entropy and one with low entropy. The former one works as it was described in the previous section. The latter one selects the most frequent features instead of the sparse ones. An optimal solution is not examined because it is out of the scope of this study and it would be computationally intractable. Thus, the experiments are not expected to demonstrate better than state-of-the-art results.

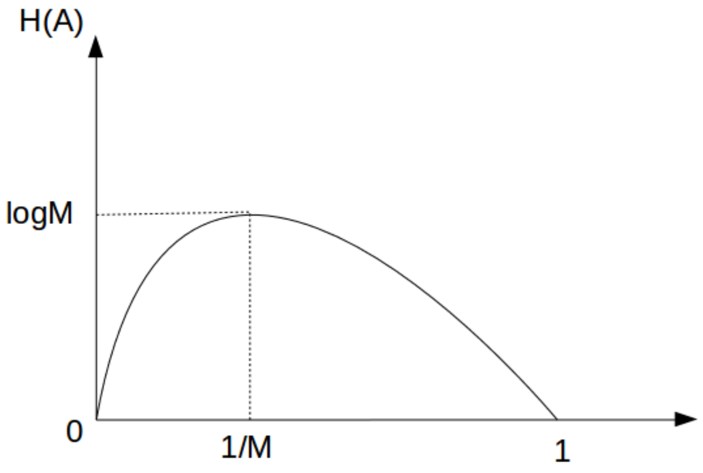


Figure 6.1.4: Diagram of an entropy function.

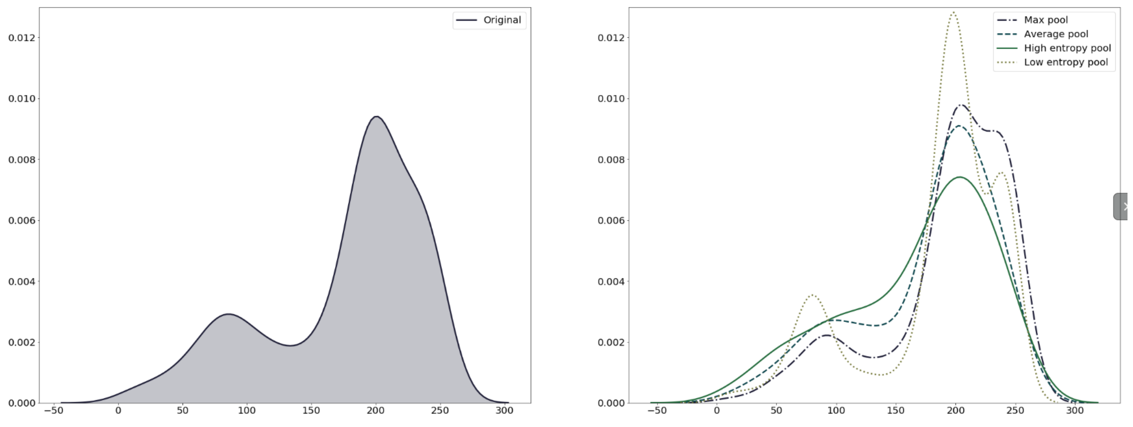


Figure 6.1.5: Max pool output distribution of a bright image.

The experiments are developed in python using Tensorflow or Pytorch. The models are trained and evaluated using the following datasets: Cifar10, Cifar100 (Krizhevsky and Hinton, 2009), MNIST and FMNIST (Xiao et al., 2017). Details about parameters of the models, that don't affect the outcomes, are omitted for the sake of space. The code of the experiments, along with all the details can be found at www.github.com/available-for-camera-ready.

6.1.3.1 Validation of Theoretical Outcomes

The effect of pooling operation on images. Considering pooling as an information channel, pure pooling is applied on images. Max, average and entropy pooling are examined using random images from the popular dataset Cifar10. The goal is to observe how the distribution of an image is changed by the three operators. The pipeline is very simple, a single channel is introduced to each pooling and then the output is plotted. Then Shannon entropy, mean and standard deviation are calculated. Tables 6.1.1 and 6.1.2 present two representative examples with the respective results of a dark image and a bright one.

The experiments verify the theoretical outcomes. The most robust operators are entropy and average pooling. The output of entropy pooling tends to be uniform and its Shannon entropy is the first or second highest. The output of average pooling, also tends to be uniform while standard deviation is always lower than the original and the mean equal to the mean of the original image. Shannon entropy of average pooling is very high for the majority of the images of this dataset. This is explained by the high cardinality of the output due to the unique results of the function of average. In case of images that have a pattern such as a chess board the cardinality would be relatively low.

Table 6.1.1: Effect of Pooling on a Bright Image

Pooling Type	**Properties**			
	Cardinality	***Mean***	***SD***	***Entropy***
Original Img	221	176.06	60.35	7.3
Max	115	192.14	54.14	6.56
Average	199	176.06	57.01	7.51
High Entropy	154	168.25	60.9	7.06
Low Entropy	86	182.5	57.37	5.85

Table 6.1.2: Effect of Pooling on a Dark Image

Pooling Type	**Properties**			
	Cardinality	***Mean***	***SD***	***Entropy***
Original Img	240	81.19	73.18	7.38
Max	164	105.69	75.43	7.15
Average	221	81.19	68.51	7.71
High Entropy	165	94.38	72.67	7.2
Low Entropy	79	69.66	73.43	5.75

Table 6.1.3: Comparison of Low and High Entropy

Dataset	***Low Entropy***	***High Entropy***
Cifar10	53.48 +-0.52	**59.07 +-0.25**
Cifar100	22.76 +-0.32	**26.80 +-0.95**
MNIST	71.73 +-12.98	**72.93 +-16.64**
FMNIST	74.54 +-34.19	**77.88 +-10.41**

According to average function, average pooling is not just selecting features, it generates new ones and the output space is not any more integers in [0, 255]. In order to make more direct the comparison with the other two pooling operators, which act as selectors, an alternative version of average pooling is also taken into account. The result of average is rounded to the closest integer. Then, the output features belong to the same group with the original ones, e.g. integers in the range [0, 255]. This new version of average pooling behaves in the same way. The output features still have high entropy, but at this time it is lower than the Shannon entropy coming from entropy pooling. The cardinality is also lower as expected.

The output of max pooling has higher mean value than the original image. The standard deviation doesn't show a consistent behaviour, which confirms the theoretical analysis. Regarding the entropy it is empirically verified that it depends on the data. More specifically, it is observed that high entropy and max pooling have similar behaviour for dark images. The equivalent entropies have almost the same value. On the other hand bright images are transformed into a lower entropy feature representation via max pooling. These observations are shown in tables 6.1.1 and 6.1.2. The difference between a bright and a dark image, is also depicted in Fig. 6.1.5 and **??**, which show the distribution of the original images and the one of the output of max pooling. The case of the dark image is closer to the uniform distribution. In practice this happens because max pooling pays attention to the higher values and tends to ignore lower ones.

Table 6.1.4: Image Classification Results

Neural Network	**Image Dataset**	**Pooling Operation**		
		Max	***Average***	***High Entropy***
LeNet	MNIST	98.93 +-0.01	98.64 +-0.01	98.51 +-0.05
	FMNIST	99.32 +-0.07	99.18 +-0.04	99.16 +-0.05
ResNet20	Cifar10	**49.25 +-39.29**	91.87 +-0.11	90.57 +-0.04
	Cifar100	64.81 +-0.44	67.77 +-0.16	62.44 +-0.48
ResNet20 2p	cifar10	85.05 +-0.06	87.15 +-0.24	88.57 +-0.22
	cifar100	56.31 +-0.43	59.41 +-0.81	62.75 +-0.08

Increasing information flow in a shallow neural network Concerning the large number of parameters of a deep neural network and the huge complexity, the simplest way to investigate how information flows in a neural network is to build a very shallow one. With this in mind, the simple neural network consists of a convolutional layer, followed by a pooling operation and a fully connected layer. The datasets that are used are: Cifar10, Cifar100, MNIST and FMNIST. Each of the datasets corresponds to a ten class classification task, apart from Cifar100 which has one hundred labels. The model is trained and tested each time with one of the following pooling operations: high entropy and low entropy. The goal is to verify that pooling features with higher entropy benefit classification accuracy regardless of the dataset or the task.

The intuition that lies in this statement is that pooling is a bottleneck and should maximize the amount of information that passes to deeper layers. By looking at table 6.1.3 and comparing low and high entropy operations, it is confirmed that high entropy pooling always achieves better performance.

Comparative Analysis of Pooling Operations There are many open questions on how a neural network is trained and it is very difficult to isolate and measure the impact of a single layer. In order to validate the robustness of entropy pooling two popular architectures are used.

The first one is LeNet (LeCun et al., 1998) and consists of two sets of convolutional and pooling layers, followed by two fully-connected layers and finally a softmax classifier. LeNet is evaluated on MNIST and FMNIST. Table 6.1.4 shows that the best accuracy is achieved by max pooling, whereas average and high entropy have equivalent accuracy.

The second architecture is ResNet20 (He et al., 2016). Two variations of ResNet20 are utilized, one with one pooling operation before the last fully connected layer and one with one extra pooling operation after the first convolutional layer. In this paper, the former one is called ResNet20 1P and the latter one ResNet20 2P. ResNet is trained and validated using Cifar10 and Cifar100.

The accuracy results of ResNet20 1P are presented in table 6.1.4. Among the various pooling operations, average fits the best. This is not a surprise as the original model is proposed with average pooling. When it comes to max pooling, the model doesn't always converge efficiently. Specifically for Cifar10, the final accuracy can fluctuate a lot, which can be attributed to the deficiency that max pooling is sensitive to feature variability. High entropy pooling demonstrates descent results for both datasets.

It is worth mentioning that the place of pooling inside this neural network might be the reason that shows the weakness of max pooling. Being at the end of the model means that all features are important and well defined by previous layers. In accordance to our theoretical conclusions max pooling misses important information that might have small absolute values, whereas average and high entropy pooling preserve the most important features.

Regarding ResNet20 2P, the pooling operation before the last fully connected layer is the average one, across all experiments. Only the first pooling is replaced with max, average and high entropy. The maximum accuracy is achieved with high entropy pooling and the rest of the results are as expected.

6.1.4 Conclusion and Future Work

This study strengthens the understanding of deep learning, by scrutinizing pooling operation from the information theory perspective. Using fundamental information theoretic principles it is evident how pooling operators enhance the performance of neural networks. The presented mathematical analysis shows the strengths and the vulnerabilities of existing pooling functions, emphasizing the need of a new property of these functions that controls the information flow.

Thereupon, pooling is revisited as a special case of the problem of max entropy sampling, suggesting a novel robust solution, named entropy pooling. The theoretical outcomes are validated thoroughly via practical experiments and the proposed pooling strategy is empirically compared to existing approaches.

These findings add to a growing body of literature on developing a complete theory of deep learning. Further work is suggested on investigating the behaviour of pooling during training of a neural network, paying attention to the order of pooling inside the network. Researchers are highly encouraged to use entropy pooling, as it can be swapped into to any existing neural network architecture. Finally, the design and development of an optimal and computationally tractable pooling operation will be a challenge for years.

6.2 Variational Information Bottleneck Pooling

VIB-Pooling is an original pooling layer for deep neural networks based on fundamental principles. VIB-Pooling is a variational procedure that does not have any learning parameters. According to the Information Bottleneck, it chooses features based on the principle of Maximum Entropy and adds a regularization term to the neural network's objective function. The methodology employs a variational approach and automatically adjusts to select relevant features through the use of a component known as Online Adaptation.

6.2.1 Related Work on Deep Variational Information Bottleneck

Deep neural networks have demonstrated remarkable performance in a variety of tasks, but no formal theory exists to describe the underlying mechanics of their learning ability. Applying information theory in the domain of deep learning has recently gained a lot of interest, and there is a lot of potential to answer many open research questions.

The Information Bottleneck (IB) principle is a promising generalization of the theory of rate distortion (Tishby et al., 2000). It is a framework that may be used in a number of information processing systems. It was firstly introduced to statistical learning theory by Shamir et al. (2010) and later to deep learning by Tishby and Zaslavsky (2015). The latter work establishes a link between the optimal architecture of a deep neural network and the compression of the input layer relative to the output layer. This is illustrated in a type of a graph called the Information Plane, which is the plane representing the mutual information pf each layer with regards to the input and the output. Information Plane has been recently used to reveal more details about the dynamics of training a neural network (Shwartz-Ziv and Tishby, 2017). Saxe et al. (2019) presents a comprehensive investigation of the entropic dynamics of neural networks, taking the Information Bottleneck into consideration and argues that the concept of a diffusion and compression phase during training is not generic. The authors demonstrate that this assumption is true only for double-sided saturating nonlinearities such as tanh, not for ReLU or linear functions. A comprehensive mathematical description of the precise conditions under which the diffusion and compression phases occur, during neural network training, is still missing.

Many researchers have been inspired by the Information Bottleneck framework to introduce alternative formulations of the IB objective function that help overcome challenges such as estimating mutual information and showcase that it improves the robustness of deep neural networks to adversarial attacks and benefits their generalization capabilities (Alemi et al., 2017; Strouse and Schwab, 2017; Fischer, 2020; Gondek and Hofmann, 2003; Achille and Soatto, 2018b). The Deterministic Information Bottleneck (DIB) proposed by Strouse and Schwab (2017), replaces mutual information with entropy and is computationally more efficient than the previous version. Alemi et al. (2017) suggests the Deep Variational Information Bottleneck (Deep VIB) as a stochastic variation of information bottleneck. The authors show that this technique improves neural network training efficiency, generalization, and robustness to adversarial attack. The Deep VIB objective function is a variant of the Information Bottleneck objective function. A multilayer perceptron is used to compute the mean and covariance matrix of hidden layer features. The reparameterization trick (Kingma et al., 2015b) is used to resample features, efficiently compute gradients, and compute a regularization term of the objective function.

Other researchers have reached similar theoretical conclusions from a different viewpoints. Achille and Soatto (2018b) proposes a generalization of the dropout technique called Information Dropout. By infusing noise into a latent variable and adding a regularization component to the objective function, Information Dropout learns optimal representations. The objective function is generated using the Information Bottleneck principle and shows a high degree of similarity to the objective function of other approaches, such as

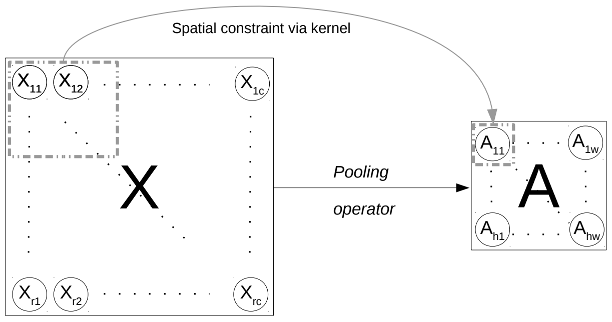


Figure 6.2.1: Multivariate pooling channel.

Variational Dropout (Kingma et al., 2015b). From a Bayesian perspective, we want to compute the posterior distribution $p(\mathbf{w}|D)$ of the model's weights $\mathbf{w}$ given a training dataset D and a prior distribution $p(\mathbf{w})$. The true posterior can then be approximated by minimizing the negative variational lower bound of the marginal log-likelihood of the data.

6.2.2 Design Requirements of Pooling Channel

According to the definition of pooling channel 6.1.2 maximizing its capacity is the first requirement that a pooling operation should meet. The second requirement is that the output information is disentangled. The latter requirement is met trivially by using the theorem that defines the independence bound on entropy (Cover and Thomas, 2012). The theorem basically states that when some random variables are independent, their joint entropy is maximized. Therefore, if the output variables of the pooling channel $a_1, a_2, \dots, a_m$ are drawn according to $p(a_1, a_2, \dots a_m)$ then $H(a_1, a_2, \dots, a_m) \leq \sum_{i=1}^{m} H(a_i)$ with equality if and only if a_i are independent. This is illustrated more clearly in the following definition of a multivariate Gaussian pooling channel, which will aid in formulating a pooling operation that satisfies the specifications and requirements later.

Definition 6.2.1. Multivariate Gaussian Pooling Channel: Let a deep neural network with a pooling operation after hidden layer i. Assume that the output features of the hidden layer i are independent and identically distributed (i.i.d.) Gaussian random variables and consist a multivariate Gaussian random variable X_N. Similarly for the output of the pooling operation with the multivariate Gaussian A_s. Then the information channel defined by the pooling operator is called a multivariate Gaussian pooling channel and can be represented by the diagram in Fig. 6.2.1.

6.2.3 Designing VIB-Pooling

VIB-Pooling consists of the following components: a sub-sampling function, the Online Adaptation matrix and a customized objective function. The sub-sampling function maximizes the capacity of the pooling channel and outputs disentangled information. The Online Adaptation matrix is a score matrix and its elements are weights that adapt to new features. Lastly, the objective function includes a regularization term in order to meet the Information Bottleneck criteria. The components that are described above are presented in details along with the entire architecture of VIB-Pooling.

Revisiting the maximum entropy sampling problem. Maximizing the entropy of a pooling channel is equivalent to the maximum entropy sampling problem (MESP). The goal of MESP is to choose a most informative subset of s random variables from a set of n random variables, subject to side constraints (Lee, 2006). It is known to be NP-Hard, e.g. for $n = 30, s = 15$ there are 155117520 possible solutions (Ko et al., 1995). There are numerous exact and approximate solutions in the literature, the majority of which use a branch and bound methodology. It is also customary to assume that the random variables are Gaussian, because the joint Gaussian distribution has the highest entropy of all probability distributions with the same mean and covariance. (Anstreicher, 2020).

Despite the large number of MESP solutions in the literature, even fast approximate solutions would be very slow to be estimated for each epoch of training a large neural network. The multivariate Gaussian pooling channel is used for this purpose, with the assumption of i.i.d. Gaussian random variables. Given a batch of feature vectors for each iteration during training, the mean and variance of these random variables can be computed empirically. The deterministic selection function represented by proposition 6.2.1 is the function that fits our needs.

Proposition 6.2.1. *Let a multivariate Gaussian pooling channel with n i.i.d. Gaussian random variables, $N_1(\mu_1, \sigma_1^2), N_2(\mu_2, \sigma_2^2), \dots, N_n(\mu_n, \sigma_n^2)$. The deterministic function that selects the Gaussian variables with the largest variances is described by the following formula:*

$$f(X_k) = X_k[g(\Sigma_k)], \tag{6.22}$$

$$g(\Sigma_k) =_{1 \le i \le k} \sigma_i^2 \tag{6.23}$$

, with k being the kernel size that pooling operates on, Σ_k the respective diagonal covariance matrix and σ_i^2 its k elements. The pooling function has the following properties:

(i) Maximizes the capacity of the pooling channel.

(ii) Returns disentangled information.

Proof. Let a multivariate Gaussian pooling channel with input the random variable X_n following a multivariate Gaussian distribution $N_X(\boldsymbol{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$. Denote $x_i, i \in [0, n]$ the i.i.d. random variables of X_n and $N_i(x|\mu_i, \sigma_i^2)$ their Gaussian distributions. Similarly, the output of the channel, given a kernel size k, will be A_s with $N_A(\boldsymbol{a}|\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A)$ and s i.i.d. Gaussian random variables $a_j, j \in [0, s]$. Given the deterministic function f that selects the random variables with the largest variances, the capacity of the channel is described by (6.5) and we have (Cover and Thomas, 2012):

$$C = \sup_{p(x)} I(X; A) = \sup_{p(x)} H(A) = \sup_{p(x)} \left(\frac{1}{2} \ln |\boldsymbol{\Sigma}_A| + \frac{s}{2}(1 + \ln 2\pi) \right) \tag{6.24}$$

From (6.24) it is obvious that the capacity is maximized when the variance of A is maximized. Therefore, Σ_A, which is a diagonal matrix, is maximized when the random variables x_i with the largest variance are selected.

The second part of the proposition is straightforward from the theorem that defines the independence bound on entropy (Cover and Thomas, 2012). □

Online Adaptation. A pooling operation in the context of a neural network, as previously explained, can be reduced to the MESP problem. Then, given a batch of features, each with n elements, each batch is treated as a sample of a multivariate random feature vector. Assuming that we have a multivariate Gaussian pooling channel, let its elements be sampled from $N_1(\mu_1, \sigma_1^2), N_2(\mu_2, \sigma_2^2), \dots, N_n(\mu_n, \sigma_n^2)$. The random variables with the biggest variances should be chosen using the pooling channel's selection function. When examining the complicated and dynamic environment of a deep neural network, there are a few factors to consider. The network's parameters are constantly changing throughout training, so elements that are significant in one iteration might not be important in the next. the network's parameters are constantly changing throughout training, so elements that are significant in one iteration might not be important in the next. Furthermore, the data batches that pass through the network are often a small sample that may or may not be representative. In the case of a drifting distribution, the same issue arises during inference. The Online Adaptation matrix is used to address the aforementioned issues. Online Adaptation is a score matrix that gives more weight to features chosen by the pooling operator and holds lower values for features chosen previously. The weights of characteristics that were selected only a few times will be removed after a few iterations. Algorithm **??** describes the method.

Algorithm 3 ONLINE ADAPTATION Computes a score matrix

```
Input: A scaling factor $\alpha$ and a matrix, named $selected\_features$, with the value 1 for each element that
       has been previously selected and 0 for the rest.
Output: A score matrix, named $scores$ with $n$ weights.
29 $scores \leftarrow scores + \alpha \cdot selected\_features$
30 foreach $kernel$ in $scores$ do
31     $scores[kernel] \leftarrow softmax(scores[kernel])$
32 end
33 return $scores$
```

Information Bottleneck criteria Given a neural network with parameters θ, the data inputs X and a task Y, let Z be a latent representation with distribution $p_\theta(z|x)$ and $\hat{Y}$ the predictions of the neural network. According to the principle of Information Bottleneck the model can be described as follows.

$$Y \leftarrow X \rightarrow Z \rightarrow \hat{Y} \tag{6.25}$$

Therefore, the Information Bottleneck objective is to minimize the following Lagrangian (Tishby et al., 2000):

$$\mathcal{L} = I(X;Z) - \beta I(Y;Z) \tag{6.26}$$

which is interpreted as a twofold objective function. The first term suggests a minimal representation and the second one a sufficient representation. In general it is hard to estimate the mutual information between two random variables. An empirical approximation of the original Information Bottleneck objective is (Achille and Soatto, 2018b; Alemi et al., 2017):

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} {}_{z\sim p_\theta(z|x_i)}[-\log p_\theta(y_i|z)] + \beta KL(p_\theta(z|x_i)||p_\theta(z)) \tag{6.27}$$

In line with Proposition 1 of Achille and Soatto (2018b) the disentanglement of the hidden factors can be achieved by minimizing the equivalent IB Lagragian with independent factors, which is in good agreement with the assumption of i.i.d. random variables of the multivariate Gaussian pooling channel. Therefore the objective function becomes:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} {}_{z\sim p(z|x)}[-\log p(y_i|z)] + \beta KL(p(z|x_i)||\prod_{i=1}^{|z|} q_i(z_i)) \tag{6.28}$$

The first term of the loss function is the average cross entropy loss function, which is commonly used in deep learning (Achille and Soatto, 2018b). This term is also proved to be the upper bound of the Deep-VIB objective function under Dropout regularization (Kirsch et al., 2020). The second term is the Kullback-Leibler divergence of the latent distribution $p_\theta(z|x_i)$ with respect to the prior independent factors $q_i(z_i)$. The Kullback-Leibler divergence can now be computed analytically as per proposition 6.2.2.

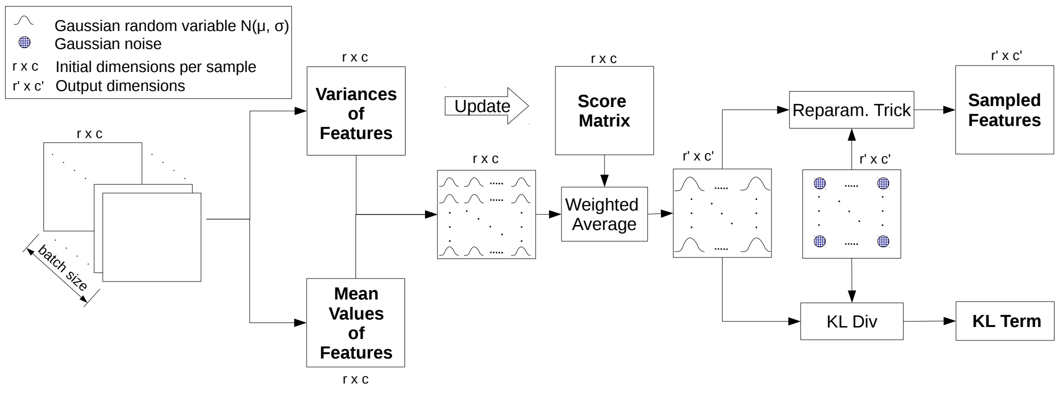


Figure 6.2.2: VIB-Pooling.

Proposition 6.2.2. ***VIB-Pooling cost*** *Let $\boldsymbol{z} \sim p(\boldsymbol{z}) = N(\boldsymbol{z}|\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$ be a multivariate Gaussian random variable with i.i.d. components. Using the reparameterization trick (**?**) we have $p(z|x)ds = p(\epsilon)d\epsilon$ where $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{\epsilon})$ is a deterministic function of $\boldsymbol{x}$ and the Gaussian random variable $\boldsymbol{\epsilon} \sim N(\boldsymbol{\epsilon}|\boldsymbol{\mu}_\epsilon, \boldsymbol{\Sigma}_\epsilon)$. Then, we have:*

$$KL(p(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))) = log(\frac{|\boldsymbol{\Sigma}_z|}{|\boldsymbol{\Sigma}_\epsilon|}) + tr(\boldsymbol{\Sigma}_z^{-1}\boldsymbol{\Sigma}_\epsilon) + (\boldsymbol{\mu}_z - \boldsymbol{\mu}_\epsilon)^T\boldsymbol{\Sigma}_z^{-1}(\boldsymbol{\mu}_z - \boldsymbol{\mu}_\epsilon) - n \tag{6.29}$$

Proof. The proof is straightforward considering the KL divergence of two multivariate normal distributions (Cover and Thomas, 2012). □

The end-to-end architecture of VIB-Pooling is illustrated in Figure 6.2.2. Firstly, the input of the module consists of features with dimensions $batch\_size \times r \times c$. Then, two matrices are computed and each of them has dimensions $r \times c$. One of the matrices represents the variances of the features over the number of samples which is equal to $batch\_size$. The other one includes the mean values of the features. Utilizing the values of these two matrices as means and variances we have a random matrix with shape $r \times c$, the elements of which represent normal distributions of the features. Using the variances matrix and the selection function from proposition 6.2.1, the score matrix is updated according to the online adaptation algorithm **??**. Next, the score matrix is used to compute the weighted average pool giving a random matrix of Gaussian variables with size $r' \times c'$. The dimensions r' and c' depend on the kernel size of pooling. Then using the reparameterization trick the output of the layer is a $r' \times c'$ matrix with elements the resampled features. KL divergence is computed as per proposition 6.2.2 and is used as a regularization term of the objective function.

6.2.4 Comparative Analysis

It is possible to include VIB-Pooling into any neural network and has been tried on a variety of designs and datasets. Consistently, it shows enhanced performance, especially when the model is presented with data that exhibit distribution shifts. This section explains the outcomes of the experiments, highlighting both the model's advantages and disadvantages.

The experiments concern image classification tasks utilizing the following public datasets: KMNIST (Clanuwat et al., 2018), SVHN (Netzer et al., 2011) and Cifar10 (Krizhevsky and Hinton, 2009). KMNIST includes grey-scale images depicting Japanese characters. SVHN contains digits and numbers in natural

Table 6.2.1: Test results on Cifar10 with ResNet and MobileNetV2 being trained in 10 epochs.

Model	ResNet			MobileNetV2		
Test Data	Avg	Max	VIB	Avg	Max	VIB
No Transform	78.8±0.8	75.1±0.6	**79.7±0.5**	78.9±0.4	77.7±1.0	**80.0±0.6**
Gaussian Blur	56.5±2.5	**59.1±2.4**	57.0±2.2	61.8±3.0	61.2±2.9	**62.3±2.9**
Horizontal Flip	78.5±2.4	74.9±2.4	**79.5±1.5**	78.7±3.4	77.6±3.4	**79.7±2.6**
Vertical Flip	55.9±0.8	53.5±0.7	**56.5±0.4**	54.4±0.5	53.8±1.0	**55.0±0.7**
Rotation 30	54.3±0.7	54.1±0.6	**55.3±0.6**	51.8±0.8	**54.3±1.1**	53.2±1.1
Affine Shear	50.0±2.1	48.8±1.2	**51.3±1.6**	48.2±1.7	47.9±1.4	**49.0±0.9**

scene images. Cifar10 includes natural images. There are ten classes of pictures in each dataset. The datasets are divided into train, validation, and test data for each experiment. Data augmentation is not necessary as a model's performance is tested separately when it is presented with data from a different distribution. Six different assessment settings are used to evaluate the models. In the first case, the test data is used as-is. Gaussian blur, horizontal and vertical flip, 30 °rotation and affine shear are the other five transformations.

Pooling operations are assessed and compared using four well-known neural architectures spanning LeNet (Lecun et al., 1998), ResNet (He et al., 2016), MobileNetV2 (Sandler et al., 2018) and DeepLayerAggregation (DLA) (Yu et al., 2018). The first two architectures are implemented from scratch and the original code of the last two architectures can be found at https://github.com/kuangliu/pytorch-cifar under the MIT License. One of the three pooling operations is used to replace the default pooling layers in each network. The parameters of VIB-Pooling are $\beta = 0.01$, $\alpha = 0.2$ and the noise distribution is $N(0, 0.01)$. These parameters are fixed throughout all models. To improve the performance of the models, these parameters could be further optimized. Pytorch Lightning (Falcon, 2019) is used to build the experiments and the models. The optimization algorithm SGD with learning rate 0.001 and momentum 0.9 is used for training. All of the experiments have a batch size of four. The majority of the tests train the models for 10 epochs because we discovered that more training has no effect on the final results and a tiny performance improvement is beyond the scope of this comparative analysis. Furthermore, the 10 epoch experiments are repeated numerous times, and the data tables provide the average accuracy as well as the standard deviation. The code can be found at: "link provided for camera ready." The computer resources employed in the studies include a Titan Xp GPU.

The experiments with Cifar10 and the neural networks ResNet and MobileNetV2 are summarized in table 6.2.1. The models are trained during ten epochs, and the experiments are repeated ten times. The results reveal that VIB-Pooling outperforms traditional approaches, with higher performance boosts for both models ResNet and MobileNetV2. VIB-Pooling delivers features that are more invariant to modifications in the majority of out-of-distribution testing conditions. The experimental findings of the same models and setup, but utilizing the SVHN dataset, are shown in table 6.2.2. Again, VIB-Pooling outperforms the others in practically all test circumstances.

Table 6.2.3 lists the experimental results regarding LeNet and DLA. The latter one is trained in 100 epochs and VIB outperforms the others. The validation accuracy of DLA for the three types of pooling is illustrated in Figure **??**, where VIB is ahead except for a few epochs around 45 and 50. LeNet is trained in 10 epochs and is the only case in which the suggested technique comes in second when no transformation testing is performed. In terms of out-of-distribution data, LeNet VIB and max pooling are practically equivalent, with the average one being the worse.

Table 6.2.2: Test results on SVHN with ResNet and MobileNetV2 being trained in 10 epochs.

Model	ResNet			MobileNetV2		
Test Data	Avg	Max	VIB	Avg	Max	VIB
No Transform	94.5±0.7	93.7±0.5	**94.9±0.4**	94.0±0.3	91.1±0.3	**94.2±0.2**
Gaussian Blur	94.4±0.6	93.5±0.5	**94.7±0.3**	93.8±0.2	91.0±0.3	**93.9±0.2**
Horizontal Flip	65.9±0.7	66.3±0.5	**67.3±0.3**	**66.1±0.3**	63.0±0.2	65.4±0.1
Vertical Flip	65.6±0.9	65.6±0.3	**66.5±0.7**	**65.9±0.2**	64.6±0.2	65.5±0.1
Rotation 30	83.3±1.5	79.4±1.0	**83.4±1.3**	82.0±0.2	79.3±0.4	**82.8±0.1**
Affine Shear	**62.8±0.7**	57.1±0.7	61.5±0.9	60.9±2.1	59.4±2.9	**62.6±2.6**

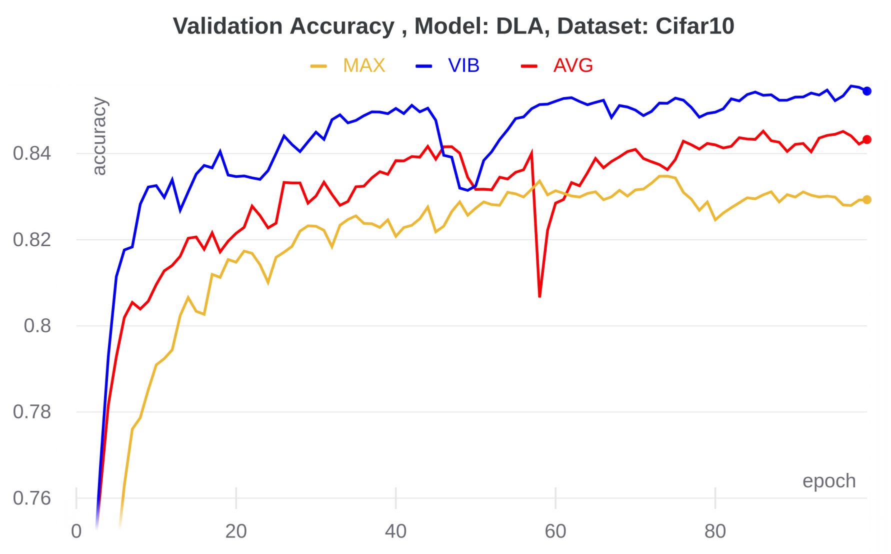


Figure 6.2.3: Training DLA with various pooling operations. VIB-Pooling outperforms average and max pooling.

Table 6.2.3: Test results of LeNet with KMNIST and DLA with SVHN. LeNet is trained in 10 epochs and DLA in 100.

Model	LeNet			DLA		
Dataset	KMNIST			Cifar10		
Test Data	Avg	Max	VIB	Avg	Max	VIB
No Transform	95.2±0.1	**96.0±0.1**	95.6±0.3	84.3±0.5	83.0±0.9	**84.5±0.5**
Gaussian Blur	94.8±0.2	95.0±0.1	**95.1±0.2**	71.6±3.0	**71.8±2.5**	70.0±2.6
Horizontal Flip	**58.8±0.3**	**58.8±0.3**	**58.8±0.3**	**84.3±2.9**	82.9±3.1	**84.3±2.1**
Vertical Flip	62.3±0.6	**63.3±0.9**	62.9±0.8	57.3±0.7	55.9±1.1	**57.4±0.5**
Rotation 30	72.4±0.4	**73.8±0.6**	73.2±0.6	57.9±0.9	**59.1±1.0**	58.2±1.0
Affine Shear	59.3±0.7	60.0±0.8	**60.1±0.6**	54.8±1.9	53.5±1.3	**55.7±1.6**

To conclude, when it comes to ResNet and no transformation testing, the suggested pooling layer outperforms max pooling by up to 5.7 % and average pooling by up to 1.1 %. The performance gain for MobileNetV2 is up to 2.8 % and 1.3 % when compared to the max and average layers, respectively. In terms of DLA, it is 1.7 % and 0.2 %, respectively. LeNet is the only model in which VIB-Pooling comes in second after max pooling. One possible explanation is that the pooling layer in ResNet, MobileNetV2, and DLA is closer to the last layers, making the regularization term more relevant to the objective function. LeNet, on the other hand, contains two pooling layers that are closer to the first levels. In general, the experimental results are quite positive, as the proposed pooling operation outperforms the standard approaches and enhances the robustness of the model. These findings are consistent with the theoretical findings of this study and justify the ideas that were employed to develop the model.

6.2.5 Conclusion and Future Work

The VIB-Pooling algorithm is based on fundamental ideas such as maximal entropy and information bottleneck. It includes variational elements but no learning parameters. It just has a few hyperparameters that should be tweaked for optimal performance. The experimental findings show that the proposed pooling layer is quite competitive versus traditional ones. It outperforms max and average pooling most of the times and is more robust to disturbances and out-of-distribution data.

The methodology used in this work establishes a theoretical methodology on how to explain the dynamics of neural architectures or develop new ones. Building an entire neural architecture from scratch, based on information theoretic principles would be an intriguing research direction. The suggested VIB-Pooling layer has space for improvement in future study. It is recommended to look at fast solutions to the maximum entropy sampling problem that take into account the dependencies of variables. Furthermore, more complex Online Adaptation algorithms should increase the performance. Finally , how to merge different regularization terms that come from different layers using the IB objective function remains unsolved.

6.3 Real World Applications

6.3.1 Speech Classification

Internet-of-Things emerged from the amalgamation of the physical and digital world via the Internet. Billions of devices that are used in our daily life, are connected to the internet. Our environment is surrounded by mobile phones, smart appliances, sensors, Radio Frequency Identification (RFID) tags and other pervasive computing machines, which communicate with each other and most importantly with humans. From the humans perspective the most natural way to communicate is by speaking. Speech recognition has been one of the most difficult tasks in artificial intelligence and machine-to-human user interfaces have been restricted so far to other options such as touch screens. Yet, two technological advancements paved the way for more friendly user interfaces based on sound.

The first technological advancement is the rise of multimedia devices like smart phones. Especially the development of digital assistants and their incorporation not only in mobile phones but also in smart home or smart car kits, has established the need for audio based interactions with humans. The second advancement is the Deep Learning revolution in many applications of artificial intelligence.

Deep neural networks have shown a tremendous success in many domains including, but not limited to, computer vision, natural language processing, speech recognition, energy informatics, health informatics etc. Such models are already applied to real world applications such as medical imaging Viswanathan et al.

(2021), autonomous vehicles Fayyad et al. (2020), activity recognition Lentzas and Vrakas (2019), energy disaggregation Nalmpantis and Vrakas (2020) and others. Speech recognition is not an exception and there is an increasing interest in audio based applications that can run on embedded or mobile devices Solovyev et al. (2020).

Some examples of sound recognition tasks are automatic speech recognition (ASR), speech-to-text (STT), speech emotion classification, voice commands recognition, urban audio recognition and others. For several years, researchers were trying to manually extract features from sound that are relevant to the task. Thus, the traditional pipeline of such systems includes a preprocessing step, feature extraction and a learning model Tsipas et al. (2015); Zhang et al. (2018b). The first two steps mainly include unsupervised signal processing techniques, extracting information in the frequency domain, exploiting frame-based structural information and others Bountourakis et al. (2019). Recently, deep neural networks have demonstrated unprecedented performance in several audio recognition tasks, outperforming traditional approaches Vrysis et al. (2020b,a); Zhang et al. (2018b); Han et al. (2020).

The purpose of this study is to create effective deep learning systems that are resilient to nuances in audio classification tasks. Speaker recognition and voice command identification are the tasks for evaluating the suggested models. The first concerns the recognition of five different speakers using the well-known dataset "Speaker Recognition Dataset Prominent Leaders Speeches." The latter seeks to detect ten voice commands from Google's popular dataset "Google Speech Commands." Two unique neural architectures are created using these datasets. These models outperform identical state-of-the-art models in terms of performance, but they are computationally lighter due to fewer learning parameters. Furthermore, the suggested models employ entropy-based feature pooling, which increases their robustness to noise.

6.3.1.1 Related Work

Modern deep learning systems have shown unprecedented performance in many domains outperforming humans. Some non exhaustive areas, where deep learning thrived, are image classification (Simonyan and Zisserman, 2014; He et al., 2016), speech recognition (Chorowski et al., 2015; Bahdanau et al., 2016) and natural language understanding (Devlin et al., 2019; Qu et al., 2020). The performance of neural networks is further boosted by modern hyperparameter optimization methods such as automl (He et al., 2021), quantum genetic algorithms (Lentzas et al., 2019) and swarm intelligence (Bacanin et al., 2021).

This technological advancement is already transforming the industry spanning many sectors such as energy (Nalmpantis and Vrakas, 2020), food (Makridis et al., 2020), automotive (Falcini and Lami, 2017; Fayyad et al., 2020), medicine (Viswanathan et al., 2021) and many others. In this light, deep learning has raised the baseline in many sound recognition tasks such as automatic speech recognition (ASR), speech-to-text (STT), speech emotion recognition, voice commands recognition, environmental sound recognition (ESR) and others. The traditional pipeline of such systems includes a preprocessing step, feature extraction and a learning model (Zhang et al., 2018b; Bountourakis et al., 2019; Vrysis et al., 2020c). Feature extraction is not always computationally efficient (Tippaya et al., 2017) and modern approaches in feature preprocessing suggest fast methods for computing local features from image and video frames (Abdulhussain et al., 2019). Deep neural networks can provide end-to-end solutions without the need for handcrafted feature engineering and outperforming traditional approaches (Vrysis et al., 2020b,a; Zhang et al., 2018b; Han et al., 2020).

So far, the majority of machine learning research has been performance oriented, ignoring other aspects like efficiency. As a result, the best performing models consist of millions or billions of parameters requiring a cloud of powerful GPUs to run. Reaching such computational limits and in accordance to the demands of the industry, researchers realized that building efficient neural networks with limited size is essential. A strong example is modern language models like Bert (Devlin et al., 2019), which is now replaced by smaller versions such as MobileBERT (Sun et al., 2020). Similar efforts in IoT applications managed to achieve state-of-the-art performance by reducing the size of the layers and replacing them with more efficient ones such as attention mechanism (Gkalinikis et al., 2020; Dong et al., 2020). Other methodologies suggest quantization aware fine tuning (Li et al., 2020a) for NLP tasks or dimensionality reduction for time series (Nalmpantis and Vrakas, 2019b). Compression methods have also been utilized in speech recognition (Li et al., 2019; McGraw et al., 2016) and there is an increasing interest in audio based applications that can run on embedded or mobile devices (Solovyev et al., 2020).

Coucke et al. (2019) present a neural network with dilated convolution layers which combined gated activations and residual connections. Their contribution is twofold. The proposed neural network fits in embedded devices and the dataset that they created, named "Hey Snips" is public with utterances recorded by over 2.2K speakers. Kusupati et al. (2018) propose a novel recurrent neural network (RNN) architecture named FastGRNN, which includes low-rank, sparse and quantized matrices. The developed neural network is up to 35x smaller than other state-of-the-art RNNs. The goal of this study is to develop neural networks that can easily be deployed on IoT devices. Zeng and Xiao (2019) propose a model called DenseNet-BiLSTM for the task of keyword spotting (KWS). DenseNet-BiLSTM is evaluated utilizing the Google Speech Commands dataset (Warden, 2018). Their main contribution is the combination of a new version of DenseNet named DenseNet-Speech and BiLSTM. DenseNet learns local features and at the same time maintains sequential patterns. BiLSTM learns time depended features. Solovyev et al. (2020) use different representations of sound such as Wave frames, Spectrograms, Mel-Spectograms and MFCCs and compare different convolutional neural networks. The outcome is that the best performing networks are the ones inspired by VGG (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016). The models are evaluated on the Google Speech Commands dataset, achieving accuracy over 90%.

Apart from high accuracy, small size and efficiency, deploying a sound recognition machine learning model on edge devices and in acoustically noisy environments requires it to be robust. Zhang et al. (2018b) present an overview of models which are resilient to noise and compare previous approaches with deep learning ones. Deep learning outperforms older methods, however most of these models do not meet the rest of the requirements for deployment on computationally constraint machines and in noisy real environments. **?** focus on both efficiency and robustness proposing a shallow convolutional neural network with only three layers: a convolutional, a max pooling and a softmax one. The convolutional layer consists of many filters that, according to the authors, play the role of a cochlear filter. The proposed system incorporates a preprocessing step, converting an audio signal into spectrogram image features. The system is evaluated using the Real Word Computing Partnership Sound Scene Database in Real Acoustic Environments (Nakamura et al., 1999) and outperforms other larger convolutional neural networks. One important outcome from this study is that the number of filters improve the robustness of the model. Another noteworthy outcome is that the robustness is stronger when training occurs with both clean and noisy data. Adding noise and augmenting data has been a good practice to enhance a model's robustness (Chen et al., 2020), however it is not always feasible to replicate the real-world data distribution. Therefore, it is equally important for the model itself to encapsulate mechanisms that will make it resilient to unpredictable noise. Pervaiz et al. (2020) study the performance of machine learning models when they are trained with noisy data. The authors evaluate a Gaussian Mixture Model (GMM) and some variations of convolutional neural networks using the Google Speech Commands Dataset. The main conclusion is that augmenting noise in training data improves the performance of the models. Wang (2020) suggests a hierarchical audio content classification approach that leverages the robustness of the system. The solution consists of three components: voice activity detection based on entropy, speech/music discrimination using support vector machine and post-processing which is rule based. The target classes are noise, speech and music.

The method is efficient and robust but more experiments are advised to be conducted on more complex classification tasks e.g. with more labels.

The literature review shows that there is plenty room for improvement in sound recognition tasks. This research focuses on audio classification tasks with regards to robustness and efficiency of deep neural networks without degrading performance or increasing the parameters of the model. The models utilize pooling operations for noise invariance and experimental results show the importance of the maximum entropy (maxent) principle when designing neural networks.

6.3.1.2 Systems Architecture

The deep learning models that are used in the experiments are trained with audio data in the frequency domain. Since one model consists of 2D convolutional layers and the other one of 1D layers, there are two respective preprocessing steps. For the 2D case, we compute the Spectogram of the input, using the algorithm of Short Time Fourier Transform (STFT). The advantage of STFT is that it maintains temporal information because the Fourier transformation applies to segments of the given data and not to the entire time series. Apart from the audio data, STFT also has two parameters the frame length and the stride. The result of the transformation is a complex matrix. Next, the energy spectogram is computed using the magnitude of the complex elements and calculating their logarithm. Finally, the angle of the complex elements is also concatenated in the feature set. Regarding the 1D case, the Fourier transformation is applied on the given batch of data. The input to the model is the absolute value of the first half of the frequencies.

This research is inspired by two popular neural network architectures that demonstrated state-of-the-art results in computer vision named AlexNet (Krizhevsky et al., 2012) and ResNet (He et al., 2016). Variations of AlexNet, ResNet and others like InceptionV3, Xception and VGG have been employed in speech commands recognition by prior research (Solovyev et al., 2020). Despite the high performance of these models, there is still room for improvement in terms of both efficiency and robustness.

Efficiency is achieved by reducing the parameters without large performance sacrifices. A key component for better efficiency is the introduction of batch normalization layers in between of the main ones. Robustness is leveraged via the pooling layers. The type of these layers has been found through a systematic experimentation and evaluation of combinations of max, average and entropy functions.

The first model is a convolutional neural network with six 2D convolutional layers and activation function ReLU. After each of the first four convolutional layers there is a batch normalization layer and a pooling one. The architecture is shown in 6.3.1. Pooling layers are not specified yet. As it will be explained later on, different configurations are found to work better in the four different environmental setups of the experiments. This architecture is evaluated on the speech commands dataset.

The second model is a residual neural network. The residual block consists of a convolutional layer and a parameterized number of convolutional layers in parallel to the first one. The output of the residual convolutions is concatenated, pass through a ReLU activation function and a pooling layer. The entire architecture includes five residual blocks, followed by a pooling operation and three dense layers. The details of the architecture are shown in 6.3.2. The two pooling layers are combinations of entropy, max and average ones and the best configuration is found to be different depending on the robustness scenario. The residual architecture is evaluated on the speaker recognition dataset.

6.3.1.3 Materials, Methodology and Experimental Results

Two datasets are chosen for the evaluation of the developed models. The first one is the "Speaker Recognition Dataset Prominent Leaders Speeches" which can be downloaded from kaggle using the following link `https://www.kaggle.com/kongaevans/speaker-recognition-dataset`. It in-

Table 6.3.1: Architecture of 2D convolutional neural network.

2DConvNN	Output shape
Input	(batch size, 122, 257, 2)
BatchNormalization	(batch size, 122, 257, 2)
Conv2D 32 filters	(batch size, 122, 257, 32)
BatchNormalization	(batch size, 122, 257, 32)
Pooling Layer	(batch size, 61, 128, 32)
Conv2D 32 filters	(batch size, 59, 126, 32)
BatchNormalization	(batch size, 59, 126, 32)
Pooling Layer	(batch size, 29, 63, 32)
Conv2D 128 filters	(batch size, 27, 61, 128)
BatchNormalization	(batch size, 27, 61, 128)
Pooling Layer	(batch size, 13, 30, 128)
Conv2D 256 filters	(batch size, 11, 28, 256)
BatchNormalization	(batch size, 11, 28, 256)
Pooling Layer	(batch size, 5, 14, 256)
Conv2D 128 filters	(batch size, 5, 14, 128)
Conv2D 64 filters	(batch size, 5, 14, 64)
Flatten	(batch size, 4480)
Dense	(batch size, 12)
BatchNormalization	(batch size, 12)

Table 6.3.2: Architecture of the residual neural network.

Residual Block m= # of convolutions	Residual Network f= # of filters
Conv1D	Residual Block (m=16, f=2)
m x Conv1D	Residual Block (m=32, f=2)
Conv1D	Residual Block (m=64, f=3)
ReLU	Residual Block (m=128, f=3)
Pooling Layer	Residual Block (m=128, f=3)
	Pooling Layer
	Flatten
	3 x Dense

Table 6.3.3: Environment setup based on the presence of noise. The four scenarios are also represented by the abbreviation of three letters. E.g. FFT stands for train and evaluation without noise and test with noise.

Scenario	Training	Evaluation	Testing
Naive approach (FFF)	False	False	False
Known noise (TTT)	True	True	True
Noise augmentation (TTF)	True	True	False
Out of distribution (FFT)	False	False	True

cludes speeches of the popular leaders: Benjamin Netanyahu, Jens Stoltenberg, Julia Gillard, Margaret Thatcher and Nelson Mandela. The length of each audio is one second with sampling rate 16kHz and PCM encoded. The dataset also contains a folder with audio files representing background noise like laughing, clapping etc. The goal is to recognize the speaker taking into consideration background noise. The second dataset is the Speech Commands dataset (Warden, 2018), which has been a standard one for the task of speech commands classification targeting devices with limited computational resources. It includes 60K audio files with length around 1 second. The audio files are PCM encoded with sampling rate 16kHz. There are 32 different labels, from which only 10 are the target ones. The rest of the labels are considered as silence or unknown. The target labels are left, right, up, down, yes, no, go, stop, on, off. Figure 6.3.1 depicts a pie chart with the proportion of each target command in the training set.

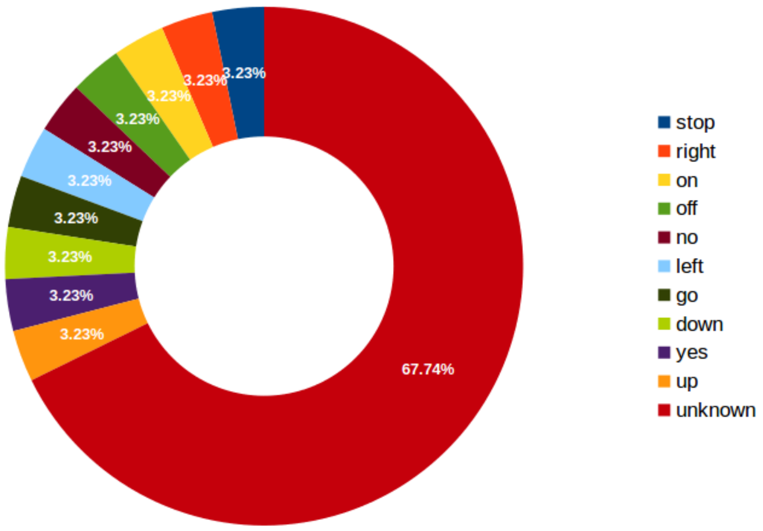


Figure 6.3.1: Proportions of target commands in Speech Commands training dataset.

The current research focuses on training and evaluating audio classification models in terms of performance, efficiency and robustness. In order to make a thorough comparative analysis a new evaluation framework is proposed with respect to the environmental setup. Four scenarios are suggested with different training, evaluation and testing environments considering whether noise should be included or not. The first one is the naive approach where there is no noise. Train, evaluation and test data are clean and are assumed to come from the same distribution. The second scenario regards noise that can be predicted, which means that we can add noise during training and evaluation. For example it would be expected to hear people laughing in an office. The test environment is supposed to have similar noise. The third case is when data

Table 6.3.4: Experimental results showing the accuracy of the residual neural network with different pooling operations. The experiments cover the four different scenarios: "naive approach (FFF)", "known noise (TTT)", "noise augmentation (TTF)" and "out of distribution (FFT)".

Pooling config.	FFF	TTT	FFT	TTF
AVG - AVG	0.988	0.967	0.683	0.961
ENTR - AVG	0.984	0.949	0.644	0.961
MAX - AVG	0.993	0.955	0.640	0.971
AVG - ENTR	**0.996**	0.966	**0.685**	0.970
ENTR - ENTR	0.983	0.956	0.659	0.975
MAX - ENTR	0.987	0.943	0.655	**0.977**
AVG - MAX	0.992	**0.968**	0.681	0.976
ENTR - MAX	0.989	0.962	0.656	0.955
MAX - MAX	0.992	0.966	0.654	0.976

are augmented with noise and then testing includes clean data. This is a rare scenario in the real world, but we include it for completeness. The last and maybe closest to real conditions scenario is when we train our data and the model has to make robust predictions even when there is unpredictable noise. In this case noise is anything that is out of distribution or not included in the target classes. Table 6.3.3 summarizes the four different scenarios with the following names respectively: "naive approach (FFF)", "known noise (TTT)", "noise augmentation (TTF)" and "out of distribution (FFT)". Because of space limitations most of the tables in this document refer to the four scenarios with their abbreviations. The abbreviations refer to whether there is noise in the training, evaluation or testing environment with F for false and T for true. Thus, FFT means that noise is included only in the test data.

The neural network based on the residual architecture is evaluated on the speaker recognition task using the dataset "Speaker Recognition Dataset Prominent Leaders Speeches". The model is configured through a systematic experimental study evaluating all the possible combinations of max, average and entropy pooling layers. The process is repeated for all the four scenarios that were described previously. In order to make the comparison of different variations of the model fair, each one is trained, evaluated and tested several times. Next, the mean and standard deviation of accuracy results are calculated and compared to find the most robust model.

Table 6.3.4 shows the results of all the different combinations of the three pooling operations for the residual neural network. In this table the name of the pooling configuration has two parts. The first part refers to the pooling of the residual block and the second one to the last pooling of the entire network. For example if the residual block has the max pooling layer and the last pooling of the network is the entropy one the pooling configuration is referred as MAX-ENTR. The most robust versions of the architecture are AVG-ENTR, AVG-MAX, AVG-ENTR and MAX-ENTR for the scenarios "naive approach", "known noise", "out of distribution" and "noise augmentation" respectively.

In order to shed light on the impact of the different pooling types on the robustness of the model, the neural networks with the configurations ENTR-ENTR, MAX-MAX and AVG-AVG are separately evaluated in terms of the cross entropy error. Figure 6.3.2 presents the experimental results in the four different environmental setups. The scenario "out of distribution" is scaled down 10 times in order to fit with the bars of the other three scenarios. The conclusion is that max pooling helps the neural network to achieve high performance under known conditions, especially in the "naive approach". Average pooling shows similar results, however it is easily biased by noise if it is present in the training data. In the case where noise is present only in the test data, average pooling is robust. Finally, entropy pooling has the lowest error in the scenario "noise augmentation" and performs on par with average pooling in the scenario "out of distribution". Entropy pooling is robust when the distribution of testing data is different from the distribution of training data. An example of the validation error during training of the three versions of the model is shown at Fig. 6.3.3. According to the diagram the three versions of the model all converge with the one that uses entropy pooling achieving the lowest error. The configuration with average pooling follows and the worse performance is from max pooling.

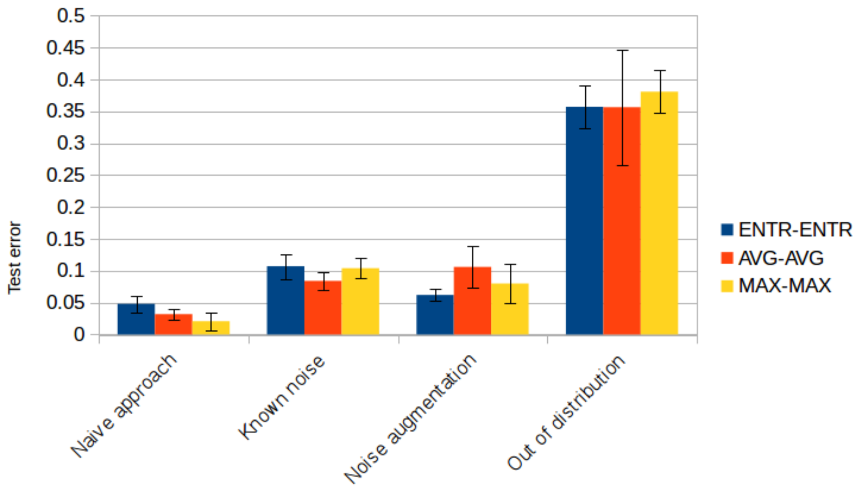


Figure 6.3.2: Evaluation of the residual neural network using one type of pooling each time. The out of distribution testing results are scaled down 10 times to fit the diagram with the rest of the results.

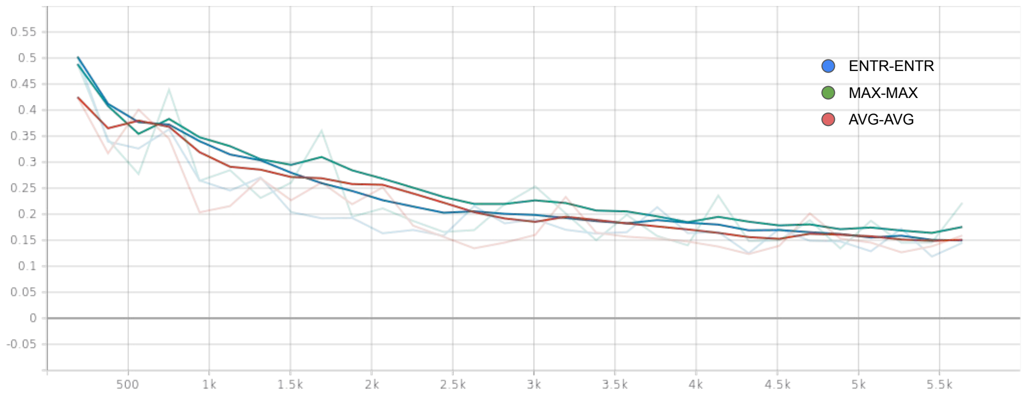


Figure 6.3.3: The diagram depicts the cross entropy error of the residual neural network during training. The three versions of the network correspond to the pooling types max, average and entropy. All three versions converge for the speaker recognition task. Average and entropy pooling show similar performance while max one has higher error.

Among the four scenarios, "out of distribution" is the most common in the real world because a real environment is unpredictable and any kind of sound can happen spanning music, traffic, appliances, animals and others. More experiments are conducted taking into account different scaling factors of the amplitude of noises. Table 6.3.5 presents the results from five different cases of noise with scaling factors x0.25,

Table 6.3.5: Experimental results of residual neural network with different pooling operations when testing dataset is out of distribution and in the presence of different scaling factors of the magnitude of the noise.

Pooling config.	x0.25	x0.5	x0.75	x1	x2
AVG - AVG	0.808	0.683	0.558	0.491	0.368
ENTR - AVG	0.810	0.644	0.558	0.491	0.369
MAX - AVG	**0.819**	0.640	0.527	0.488	0.377
AVG - ENTR	0.794	**0.685**	0.579	0.506	**0.404**
ENTR - ENTR	0.793	0.659	0.539	0.484	0.389
MAX - ENTR	0.790	0.655	0.557	0.508	0.372
AVG - MAX	0.802	0.681	0.578	0.510	0.375
ENTR - MAX	0.764	0.656	0.549	**0.512**	**0.404**
MAX - MAX	0.810	0.654	**0.589**	0.492	0.391

x0.5, x0.75, x1 and x2. The versions of the network AVG-ENTR and ENTR-MAX seem the most robust ones and should be preferred especially when we cannot predict or control the intensity of noise in the real environment.

The 2D convolutional architecture is evaluated in the Google Speech Commands Dataset. This model includes four pooling layers. The experiments include all the variations of the model by combining max and entropy pooling layers. For brevity we refer to the architectures with four letters depending on the type of the pooling layers. For example MEME would be the architecture where the first to the last pooling types are max, entropy, max and entropy. The metrics that are used are the accuracy and the cross entropy error. The experimental environments are the naive approach and the out of distribution scenario. Table 6.3.6 presents the results with the accuracy of each model. The first column represents the naive approach where there is no noise. The two best models are MMMM and MMEM, with the first one performing slightly better. The other four columns show the results when there is noise in the testing environment with various scaling factors. For the case where the amplitude of noise is scaled by 0.25 the most robust model is EMMM. MEMM and MMEM show very similar performance. For the cases with scaling factors 0.5 and 1, the best model is MMEM followed by MEMM in both cases. In the last case with scaling factor 2, the best performance is shown by EMME, followed by MEEM, EEEE and MMEM. Overall MMEM looks the most promising one when the model has to be deployed in an environment with unexpected noises. Now, considering the metric of cross entropy error, the results are slightly different. As shown in table 6.3.7 the models MEEM and EMME show the lowest error. Comparing MMEM which shows the best overall accuracy and MEEM or EMME which show the lowest overall error, it is concluded that MMEM shows relatively low error as well. MEEM and EMME also show descent accuracy, but MEEM seems to outperform EMME in most cases. Thus, the two most robust models with respect to both metrics are MMEM and MEEM.

This work evaluates two deep neural networks evaluated in two different audio classification tasks. The architectures of the networks are built with efficiency in mind, having as less parameters as possible without compromising a lot of performance. This indicates that existing deep neural networks are far from an optimal architecture. Therefore, there is a need to discover formal mathematical methods for designing neural networks. One research direction is to design novel neural layers based on fundamental principles such as the entropy pooling.

Furthermore, the impact of pooling layers on the robustness of the models is examined. A systematic

Table 6.3.6: Results showing the accuracy of the 2D convolutional network with different pooling operations. M stands for max pooling and E for entropy pooling. The first column covers the naive approach and the rest include noise in testing with a scaling factor.

Pool conf.	FFF	x0.25	x0.5	x1	x2
MMMM	**0.932**	0.856	0.782	0.706	0.620
MMEM	0.931	0.875	**0.811**	**0.720**	0.637
MMME	0.926	0.861	0.805	0.707	0.611
EMMM	0.923	**0.876**	0.805	0.677	0.512
MEMM	0.923	0.870	0.810	0.713	0.605
EMEM	0.923	0.836	0.775	0.686	0.556
MEEM	0.917	0.860	0.795	0.717	0.641
EMME	0.916	0.854	0.787	0.712	**0.644**
EEMM	0.916	0.826	0.748	0.645	0.540
EEME	0.915	0.830	0.755	0.624	0.581
EEEM	0.914	0.846	0.785	0.702	0.619
MEME	0.909	0.833	0.746	0.602	0.437
MMEE	0.907	0.843	0.773	0.696	0.634
EEEE	0.901	0.792	0.724	0.668	0.638
EMEE	0.898	0.841	0.776	0.708	0.628
MEEE	0.888	0.816	0.734	0.668	0.616

Table 6.3.7: Results showing the cross-entropy error of the 2D convolutional network with different pooling operations. M stands for max pooling and E for entropy pooling. The first column covers the naive approach and the rest include noise in testing with a scaling factor.

Pool conf.	FFF	x0.25	x0.5	x1	x2
MMMM	**0.289**	0.553	0.797	1.102	1.448
MMEM	0.321	0.587	0.791	1.088	1.380
MMME	0.328	0.583	0.797	1.151	1.513
EMMM	0.325	0.610	0.882	1.260	1.689
MEMM	0.341	0.580	0.786	1.097	1.441
EMEM	0.325	0.733	0.924	1.220	1.595
MEEM	0.362	0.563	0.764	**1.040**	**1.344**
EMME	0.346	**0.539**	**0.746**	1.058	1.406
EEMM	0.362	0.728	0.967	1.295	1.605
EEME	0.374	0.625	0.858	1.646	1.831
EEEM	0.365	0.617	0.804	1.089	1.408
MEME	0.411	0.708	0.967	1.367	1.800
MMEE	0.408	0.620	0.829	1.110	1.405
EEEE	0.405	0.757	0.973	1.224	1.434
EMEE	0.419	0.620	0.808	1.069	1.371
MEEE	0.456	0.663	0.924	1.258	1.582

evaluation process is followed taking into consideration different environmental conditions of noise. The main outcome is that entropy pooling seems promising in making a deep neural network robust to noise. The combination of pooling layers is also shown to demonstrate strong performance. Further experiments are advised to be conducted by exploring the impact of pooling on the performance of other neural architectures as well as on other datasets. Entropy pooling could also be improved. One possible improvement of entropy pooling is to find a more optimal solution. However, this is considered an NP-hard problem and better solutions trade off a lot of computational complexity (Ko et al., 1995).

Deep learning is a research domain that grows very fast and there are several neural layers that could be evaluated using noisy audio data. For future research it is recommended to explore the robustness of audio classifiers using other types of layers such as dropout and compare it with modern approaches like variational dropout (Kingma et al., 2015a) or information dropout (Achille and Soatto, 2018b). Other variational approaches, such as deep variational information bottleneck (Alemi et al., 2017), are shown to be resilient to adversarial attacks and should enhance the performance of neural networks under distribution shifts.

6.3.2 Medical Time Series

Deep learning has demonstrated significant results in various domains including medicine. Because of the unprecedented performance, it is used more and more for a plethora of clinical classification tasks. Real-world medical problems have a relatively small number of available digital records and usually suffer from missing data, imbalanced classes, lack of labels and other similar problems. Furthermore, other elements derived from the clinician's experience and affecting the diagnosis are not included in the dataset. This epistemic uncertainty, as well as the process's inherent aleatory unpredictability, can be addressed using uncertainty quantification methodologies and probabilistic modelling.

This research work utilizes popular deep learning models and integrates the VIB-Pooling operation in order to demonstrate the applicability of the proposed model in medical time series tasks and its robustness when there is uncertainty. The medical time series data are selected from the UCR Time Series Repository Chen et al. (2015). The datasets related to health are ECG200, ECGFiveDays, TwoLeadECG, ECG5000, CinC_ECG_torso, DistalPhalanxOutlineCorrect, DistalPhalanxTW, MedicalImages, ProximalPhalanxOutlineAgeGroup, PhalangesOutlinesCorrect, DistalPhalanxOutlineAgeGroup, NonInvasiveFatalECG_Thorax2, ProximalPhalanxOutlineCorrect and ProximalPhalanxTW. These datasets include time series of various lengths and concern classification tasks predicting health issues. There are five electrocardiogram (ECG) related datasets predicting or identifying health problems such as myocardial infarction. MedicalImages includes histograms of pixel intensity of medical images that are mapped to different human body regions. The rest of the datasets regard bone age assessment which is usually performed in pediatric patients to identify possible growth disorders. An expert usually performs bone age assessment using the Tanner-Whitehouse technique. This entails categorizing each bone into one of seven types based on its developmental stage Cao et al. (2000).

6.3.2.1 Electrocardiogram Data Mining

The electrocardiogram (ECG/EKG) is a non-invasive diagnostic tool used to measure the physiological activities of the heart over time. Many cardiovascular disorders, such as premature contractions of the atria (PAC) or ventricles (PVC), atrial fibrillation (AF), myocardial infarction (MI), and congestive heart failure, can be diagnosed using ECG data (CHF). In the medical field, we have seen the rapid development of portable ECG monitors, such as the Holter monitor (Nikolic et al., 1982) and most recently wearable devices in various healthcare areas, such as the Apple Watch. As a consequence, the volume of ECG data requiring analysis has increased at a rate that human cardiologists are unable to keep up with. Hence, automating and reliably analyzing ECG data has become a prominent study area. Additionally, other developing applications based on ECG data, such as biometric human identification and sleep staging, can be implemented.

Traditional ECG analysis approaches entail the employment of human specialists to apply feature engineering methods based on raw ECG data and then follow rule based AI techniques or machine learning methods to get final results. In fact, expert features are retrieved automatically utilizing computer-based techniques. However, they are generally inadequate because they are restricted by the quality of the data and human expertise (Schläpfer and Wellens, 2017; Guglin and Thatai, 2006).

The advantage of deep learning approaches is that they do not necessitate an explicit feature extraction stage performed by human specialists. Deep learning models extract features automatically during training. Studies have demonstrated that deep learning features are more informative than expert ones for ECG data (Hong et al., 2019). Deep learning algorithms also outperform classical methods on numerous ECG analysis applications, such as disease identification (Clifford et al., 2017) and sleep staging (Ghassemi et al., 2018). A systematic review on deep learning methods for mining ECG data is presented by Hong et al. (2020).

6.3.2.2 Predictive Modelling of Bone Aging

Typically, bone age evaluation entails calculating a patient's age from a radiograph by assessing the growth of the bones of the non-dominant hand. It is used to determine whether a child's bones are developing at an acceptable rate and to track how certain medications affect a patient's skeletal growth. Currently, this task is carried out manually with the use of an atlas based system such as Greulich and Pyle (GP) (Garn, 1959) or a bone scoring method like Tanner and Whitehouse (TW) (Tanner, 1983). Atlas approaches like GP compare the query image to a set of representative hand radiographs collected from participants of various ages. Scoring systems classify each bone into one of several specified stages, which are then used to give an age estimate. Manual techniques take time and are frequently wrong. Previously, automated technologies for determining bone age were proposed. These either try to reproduce the TW or GP approaches (Niemeijer et al., 2003), or they build regression models for chronological age (Thodberg et al., 2008).

6.3.2.3 Experiments and Discussion

The goal of the experiments is to demonstrate that VIB-Pooling improves the performance of standard neural networks when classifying medical time series data. VIB-Pooling is compared against max and average pooling, in order to verify that it can replace these layers successfully. ResNet is selected as a baseline for all the time series tasks and it is adjusted accordingly for each problem and size of the data. The details of the implementation are described below.

ResNet18 includes one pooling operation after the residual blocks and before the last linear layer. Typically this is an average pooling. For the purpose of this experiments the default pooling layer is also replaced by max pooling and VIB-Pooling. Training of the models takes up to 70 epochs with early stop and saving the best model. The metrics that are used are accuracy and Area under the ROC Curve (AUC). All the experiments are run 100 times and table 6.3.8 lists the mean and standard deviation of the two metrics. VIB-Pooling outperforms the other two contenders for the majority of the tasks. It achieves up to 7.5% better accuracy than average pooling with the largest difference taking place for the dataset DistalPhalanxOutlineAgeGroup.

Table 6.3.8: Test results on medical time series using ResNet18. The default ResNet18 architecture includes an average pooling. ResNet15 VIB version replaces the default pooling operation with VIB-Pooling.

Model	ResNet18					
Metric	Acc			AUC		
Test Data	Avg	Max	VIB	Avg	Max	VIB
ECG200	0.767 ± 0.068	0.729 ± 0.101	**0.789** ± **0.047**	0.85 ± 0.081	0.759 ± 0.167	**0.867** ± **0.058**
ECGFiveDays	0.73 ± 0.152	0.702 ± 0.17	**0.735** ± **0.141**	0.863 ± 0.122	0.835 ± 0.155	**0.876** ± **0.101**
TwoLeadECG	0.564 ± 0.093	0.563 ± 0.069	**0.568** ± **0.078**	**0.644** ± **0.144**	0.629 ± 0.14	0.638 ± 0.111
ECG5000	0.931 ± 0.008	0.924 ± 0.059	**0.933** ± **0.007**	0.869 ± 0.017	0.866 ± 0.042	**0.874** ± **0.017**
CinC_ECG_torso	0.878 ± 0.062	0.886 ± 0.886	**0.891** ± **0.063**	0.926 ± 0.041	0.933 ± 0.932	**0.933** ± **0.04**
DistalPhalanxOutlineCorrect	0.575 ± 0.098	**0.612** ± **0.096**	0.602 ± 0.088	0.601 ± 0.072	**0.6** ± **0.094**	0.591 ± 0.062
DistalPhalanxTW	0.529 ± 0.226	0.465 ± 0.192	**0.561** ± **0.206**	0.668 ± 0.171	0.545 ± 0.132	**0.678** ± **0.174**
MedicalImages	0.605 ± 0.045	0.586 ± 0.045	**0.608** ± **0.042**	0.895 ± 0.027	0.878 ± 0.049	**0.898** ± **0.047**
ProximalPhalanxOutlineAgeGroup	**0.764** ± **0.139**	0.718 ± 0.166	0.759 ± 0.142	**0.858** ± **0.143**	0.798 ± 0.183	0.857 ± 0.139
PhalangesOutlinesCorrect	0.679 ± 0.062	**0.732** ± **0.058**	0.694 ± 0.066	0.72 ± 0.076	**0.786** ± **0.063**	0.727 ± 0.088
DistalPhalanxOutlineAgeGroup	0.66 ± 0.177	0.635 ± 0.164	**0.709** ± **0.144**	0.683 ± 0.188	0.642 ± 0.176	**0.764** ± **0.159**
NonInvasiveFatalECG_Thorax2	0.835 ± 0.148	0.797 ± 0.2	**0.852** ± **0.089**	0.98 ± 0.085	0.965 ± 0.118	**0.99** ± **0.049**
ProximalPhalanxOutlineCorrect	0.726 ± 0.045	0.737 ± 0.044	**0.737** ± **0.043**	0.832 ± 0.049	0.84 ± 0.037	**0.835** ± **0.04**
ProximalPhalanxTW	0.574 ± 0.2	0.464 ± 0.18	**0.604** ± **0.188**	0.702 ± 0.177	0.597 ± 0.148	**0.724** ± **0.171**

7 Conclusions

"Look where you want to go, otherwise you will go where you look."
— Unknown

This thesis focused on Machine Learning Approaches for Time Series Problems. The AI tasks that defined our research goals come from three different domains spanning energy disaggregation, speech recognition and analysis of health records. These domains were investigated in depth, especially for the problem of NILM which is a known NP-hard task under the category of blind source separation problems. Original machine learning systems and neural architectures were designed aiming to solve the targeted AI tasks. The proposed frameworks and algorithms, such as Signal2Vec, are not restricted to the targeted tasks but can be used in any other time series problems. Furhtermore, our work contributes to the theoretical foundations when designing deep neural networks using information theoretic principles and bridging the gap between theory and practice in the domain of deep learning.

The original methods that were developed solve time series problems that have arisen in real-world applications with high environmental, societal and economical impact. This chapter summarizes the contributions of this thesis that have been presented in detail in previous chapters. Future directions and currently unanswered research questions are presented with suggestions for future work.

7.1 Contributions

Beginning with Chapter 2 the background knowledge that our work was based on was presented. This includes classic time series representations and the transition to machine learning approaches. Then, distributed representations were discussed with emphasis on the properties that establish optimal representations and methods to learn such representations. One very important property is disentanglement, which has gained a lot of interest the recent years and will occupy the machine learning community for several years ahead. Next, popular deep neural architectures that influenced our work, were presented. Finally, old and recent advancements in the intersection of information theory and deep learning were presented.

In Chapter 3 the time series problems that lead our research milestones were investigated. The selection of the problems was based on environmental, societal and economic criteria. The demand on the industry and the impact in the real-world have driven our motivation and helped us to focus on problems that matter. The first problem is a blind source separation problem, also known as the cocktail party, especially in the domain of audio. In the domain of energy the problem is known as energy disaggregation or non-intrusive load monitoring. NILM was analyzed in depth with reference on previous approaches and emphasizing the transition to machine learning and more specifically deep learning solutions. Obstacles that prevent NILM systems to be deployed in the world were identified with suggestions on how to overcome them. In the same fashion, the internet of things and more specifically internet of audio things (IoAuT) motivated us to pick up two fundamental problems in speech recongition, named speech commands classification and speaker verification. Finally, the third domain of application regards medical AI tasks based on health signals. One of the most popular heath records are ECGs and due to the large impact in health, we selected it as one of our AI tasks. The second one was bone age assesment due to its peculiarity and the different approach of dealing with images as time series data.

Chapter 4 presented a novel benchmark framework in order to tackle the problem of power disaggregation and original machine learning approaches. The novel NILM systems that were presented include a stacking machine learning method, a transfer learning method by imaging time series data and three novel neural networks. The deep neural networks for NILM are online GRU, SAED and Neural Fourier Energy Disaggregation. All the aforementioned approaches try to solve the problem of NILM by training one model which given the main power consumption of a house, predicts the power consumption of a certain appliance. The proposed models were designed with criteria the requirements of a system in production. Two of the architectures demonstrated state-of-the-art performance with the latest one, named Neural Fourier Energy Disaggregator, requiring much less computational cost and storage capacity. SAED is also considered one of the lightest models in the literature, as it has been compared against other NILM models.

In Chapter 5 we introduced a novel time series embedding representation, named Signal2Vec. Signal2Vec is a generic algorithm that can be applied to any time series regardless of the area of the data. The main functionality is that it converts a signal into vectors which in turn can be further compressed into a single vector. The proposed approach was evaluated using energy data with two classification tasks, a multiclass classification and a multilabel one. The former one regards appliance identification given energy signals as inputs. The second one regards the problem of power disaggregation which is now seen as a multilabel task. Signal2Vec was compared against other promising time series representations, most of which were used in NILM for the first time in our work. Our algorithm outperformed all other time series representations in the multilabel NILM task. Our experiments were also extended to compare our approach against the state-of-the-art approach in multilabel NILM which utilizes delay embeddings from chaos theory as representations. Our was superior to the state-of-the-art method.

Chapter 6 focused on the theoretical aspects when designing deep neural networks. More specifically, two novel pooling operations were developed based on information theoretic principles. The first one, named entropy pooling, concerns the principle of max entropy and sees pooling operation as a communication channel. The second one is a variational feature pooling mechanism which is designed in accordance to the information bottleneck framework. Information bottleneck has played a significant role in deep learning theory and has attracted many theoretical researchers as well as practitioners. The operation is called VIB-Pooling and was compared against other classical pooling operations. According to our theoretical contributions, classic pooling mechanisms don't comply with information theory and their success depends on the data or is circumvented by the complexity of the rest of the model architecture. The theoretical outcomes were validated with experiments using classic image classification tasks. The application of both entropy pooling and VIB-Pooling in real-world applications regards audio classification tasks and mining health records accordingly. Entropy pooling showcased very strong robustness on the tasks of speech commands identification and speaker recognition, under out-of-distribution test data. VIB-Pooling demonstrated high performance gains when it replaced other classic pooling operations in popular neural architectures such as ResNet in tasks related to ECG signal classification and bone age estimation.

7.2 Future Work

Time series data emerge from complex systems which cannot be described by linear models. Problems like blind source separation and other time series problems involve dynamic data with characteristics that change over time. Existing machine learning methodologies such as ICA and others heavily depend on the assumption of identically independent distributed (i.i.d.) random variables. These techniques are proved to be inadequate for problems of complex systems. Hence, more sophisticated approaches that go beyond the i.i.d. assumption are needed to be developed. This requires the amalgamation of three separate areas: machine learning, causality and information theory.

Machine learning and causality are two research areas that so far have been evolving independently. Yet, there are many common tools that are used in both subjects. These phenomenal two worlds can be merged into one with the concept of causal factors or disentangled factors of variations. Indeed, causal factors and disentanglement are two sides of the same coin. The former one is an old concept in causality but is not connected to machine learning. The latter one is product of machine learning research, aiming to fix the drawbacks of distributed representations and fulfill the requirements of optimal representations as discussed in Chapter 2. As a consequence, there has already been set a pioneering research direction towards causal representation learning by Schölkopf et al. (2021). As far as information theory is concerned, it has been an integral part of both machine learning and causality. Therefore, it can be considered as the glue of these two separate pieces (Schölkopf, 2019).

Going back to the time series problems that were subject of this thesis, we can say that causal representation learning sounds very promising. A new concept, called independent mechanism analysis (IMA), has been recently proposed by Gresele et al. (2021). A direct application of the IMA approach is the problem of energy disaggregation, where the independent sources of energy consumption have to be disentangled and learned by a machine learning model. Therefore, the work on Signal2Vec model could be further extended by producing a latent space where the factors of energy sources are separated. In addition, problems where robustness is necessary in order to ignore noise or external factors, like at the problems of speech recognition under unpredictable environmental conditions, optimal representations are required. Disentangled factors of variations are optimal representations and learning such representations that will be robust to external nuances is an interesting research direction. Our proposed method named entropy pooling, that was described in Chapter 6, can be further developed beyond the assumption of i.i.d. and produce better features. Finally, from the causality point of view, causal factors are essential in medical applications. It is not adequate to predict a health disorder, the causal factor and the interpretation of such predictions is of equal importance in order to make medical applications ready for production. Researchers are encouraged to extend our work on VIB-Pooling from Chapter 6 and implement a pooling operation that provides interpretable features for critical applications.

BIBLIOGRAPHY

Sadiq H. Abdulhussain, Abd Rahman Ramli, Basheera M. Mahmmod, M. Iqbal Saripan, S.A.R. Al-Haddad, Thar Baker, Wameedh N. Flayyih, and Wissam A. Jassim. 2019. A fast feature extraction algorithm for image and video processing. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Alessandro Achille and Stefano Soatto. 2018a. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980.

Alessandro Achille and Stefano Soatto. 2018b. Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. 2015. Time-series clustering–a decade review. *Information Systems*, 53:16–38.

Rakesh Agrawal, Christos Faloutsos, and Arun Swami. 1993. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer.

Misbah Aiad and Peng Hin Lee. 2016. Unsupervised approach for load disaggregation with devices interactions. *Energy and Buildings*, 116:96–103.

Alex Alemi, Ian Fischer, Josh Dillon, and Kevin Murphy. 2017. Deep variational information bottleneck. In *ICLR*.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. 2018. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168. PMLR.

Alexander A Alemi and Ian Fischer. 2018. Gilbo: One metric to measure them all. *arXiv preprint arXiv:1802.04874*.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1804.06537*.

Shadi Aljawarneh, Aurea Anguera, John William Atwood, Juan A Lara, and David Lizcano. 2019. Particularities of data mining in medicine: lessons learned from patient medical time series data analysis. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–29.

Nguyen Nang An, Nguyen Quang Thanh, and Yanbing Liu. 2019. Deep cnns with self-attention for speaker identification. *IEEE Access*, 7:85327–85337.

Kurt M Anstreicher. 2020. Efficient solution of maximum-entropy sampling problems. *Operations Research*, 68(6):1826–1835.

Diego Ardila, Atilla P Kiraly, Sujeeth Bharadwaj, Bokyung Choi, Joshua J Reicher, Lily Peng, Daniel Tse, Mozziyar Etemadi, Wenxing Ye, Greg Corrado, et al. 2019. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature medicine*, 25(6):954–961.

Johannes Aßfalg, Hans-Peter Kriegel, Peer Kröger, Peter Kunath, Alexey Pryakhin, and Matthias Renz. 2006. Similarity search on time series based on threshold queries. In *International Conference on Extending Database Technology*, pages 276–294. Springer.

Jose Roberto Ayala Solares, Francesca Elisa Diletta Raimondi, Yajie Zhu, Fatemeh Rahimian, Dexter Canoy, Jenny Tran, Ana Catarina Pinho Gomes, Amir H. Payberah, Mariagrazia Zottoli, Milad Nazarzadeh, Nathalie Conrad, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. 2020. Deep learning for electronic health records: A comparative review of multiple deep neural architectures. *Journal of Biomedical Informatics*, 101:103337.

Nebojsa Bacanin, Timea Bezdan, K Venkatachalam, and Fadi Al-Turjman. 2021. Optimized convolutional neural network by firefly algorithm for magnetic resonance image classification of glioma brain tumor grade. *Journal of Real-Time Image Processing*, pages 1–14.

Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE.

M. Baranski and Voss J. 2003. Nonintrusive appliance load monitoring based on an optical sensor. In *2003 IEEE Bologna Power Tech Conference Proceedings,*, volume 4, pages 8 pp. Vol.4–.

Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. 2014a. Nilmtk: An open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th International Conference on Future Energy Systems*, e-Energy '14, page 265–276, New York, NY, USA. Association for Computing Machinery.

Nipun Batra, Rithwik Kukunuri, Ayush Pandey, Raktim Malakar, Rajat Kumar, Odysseas Krystalakos, Mingjun Zhong, Paulo Meira, and Oliver Parson. 2019a. A demonstration of reproducible state-of-the-art energy disaggregation using nilmtk. In *Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pages 358–359.

Nipun Batra, Rithwik Kukunuri, Ayush Pandey, Raktim Malakar, Rajat Kumar, Odysseas Krystalakos, Mingjun Zhong, Paulo Meira, and Oliver Parson. 2019b. Towards reproducible state-of-the-art energy disaggregation. In *Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pages 193–202.

Nipun Batra, Oliver Parson, Mario Berges, Amarjeet Singh, and Alex Rogers. 2014b. A comparison of non-intrusive load monitoring methods for commercial and residential buildings. *CoRR*, abs/1408.6595.

Christian Beckel, Wilhelm Kleiminger, Romano Cicchetti, Thorsten Staake, and Silvia Santini. 2014. The eco data set and the performance of non-intrusive load monitoring algorithms. In *Proceedings of the 1st ACM conference on embedded systems for energy-efficient buildings*, pages 80–89.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540. PMLR.

Anthony J Bell and Terrence J Sejnowski. 1995. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159.

Yoshua Bengio. 2009. *Learning deep architectures for AI.* Now Publishers Inc.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Yoshua Bengio, Paolo Frasconi, and Patrice Simard. 1993. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pages 1183–1188. IEEE.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:.

Vineetha Bettaiah and Heggere S Ranganath. 2014. An analysis of time series representation methods: data mining applications perspective. In *Proceedings of the 2014 ACM Southeast Regional Conference*, pages 1–6.

S. Bharitkar. 2019. Generative feature models and robustness analysis for multimedia content classification. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 105–110.

Roberto Bonfigli, Stefano Squartini, Marco Fagiani, and Francesco Piazza. 2015. Unsupervised algorithms for non-intrusive load monitoring: An up-to-date overview. In *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1175–1180.

Vasileios Bountourakis, Lazaros Vrysis, Konstantinos Konstantoudakis, and Nikolaos Vryzas. 2019. An enhanced temporal feature integration method for environmental sound recognition. In *Acoustics*, volume 1, pages 410–422. Multidisciplinary Digital Publishing Institute.

Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. 2011. Ask the locals: multi-way local pooling for image recognition. In *2011 International Conference on Computer Vision*, pages 2651–2658. IEEE.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

Alon Brutzkus and Amir Globerson. 2019. Why do larger models generalize better? a theoretical perspective via the xor problem. In *International Conference on Machine Learning*, pages 822–830. PMLR.

Danilo Burbano. 2015. Intrusive and non-intrusive load monitoring (a survey) inference and learning approach. *Latin Amer. J. Comput.*, 2(1):45–53.

John Parker Burg. 1975. *Maximum entropy spectral analysis.* Stanford University.

Yuhan Cai and Raymond Ng. 2004. Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 599–610.

Joseph P Campbell. 1997. Speaker recognition: A tutorial. *Proceedings of the IEEE*, 85(9):1437–1462.

Fei Cao, HK Huang, Ewa Pietka, and Vicente Gilsanz. 2000. Digital hand atlas and web-based bone age assessment: system design and implementation. *Computerized medical imaging and graphics*, 24(5):297–307.

K. Carrie Armel, Abhay Gupta, Gireesh Shrimali, and Adrian Albert. 2013. Is disaggregation the holy grail of energy efficiency? the case of electricity. *Energy Policy*, 52:213–234. Special Section: Transition Pathways to a Low Carbon Economy.

Kin-Pong Chan and Ada Wai-Chee Fu. 1999. Efficient time series matching by wavelets. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 126–133. IEEE.

Hsueh-Hsien Chang, Po-Ching Chien, Lung-Shu Lin, and Nanming Chen. 2011. Feature extraction of non-intrusive load-monitoring system using genetic algorithm in smart meters. In *2011 IEEE 8th International Conference on e-Business Engineering*, pages 299–304.

Gal Chechik, Amir Globerson, Naftali Tishby, Yair Weiss, and Peter Dayan. 2005. Information bottleneck for gaussian variables. *Journal of machine learning research*, 6(1).

Lei Chen and Raymond Ng. 2004. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803.

Lei Chen, M Tamer Özsu, and Vincent Oria. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502.

Qiuxia Chen, Lei Chen, Xiang Lian, Yunhao Liu, and Jeffrey Xu Yu. 2007a. Indexable pla for efficient similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 435–446. VLDB Endowment.

Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. 2018. Isolating sources of disentanglement in vaes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2615–2625.

Shuxiao Chen, Edgar Dobriban, and Jane Lee. 2020. A group-theoretic framework for data augmentation. *Advances in Neural Information Processing Systems*, 33.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2180–2188.

Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. 2015. The ucr time series classification archive. `www.cs.ucr.edu/~eamonn/time_series_data/`.

Yueguo Chen, Mario A Nascimento, Beng Chin Ooi, and Anthony KH Tung. 2007b. Spade: On shape-based pattern detection in streaming time series. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 786–795. IEEE.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Edward Choi, Mohammad Taha Bahadori, Joshua A. Kulas, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. 2016a. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3512–3520, Red Hook, NY, USA. Curran Associates Inc.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. 2016b. Doctor ai: Predicting clinical events via recurrent neural networks. In *Proceedings of the 1st Machine Learning for Healthcare Conference*, volume 56 of *Proceedings of Machine Learning Research*, pages 301–318, Northeastern University, Boston, MA, USA. PMLR.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. In *International Conference on Learning Representations*.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*.

Grzegorz Chrupała, Akos Kádár, and Afra Alishahi. 2015. Learning language through pictures. *arXiv preprint arXiv:1506.03694*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *International conference on machine learning*, pages 2067–2075. PMLR.

Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. *arXiv preprint arXiv:1603.00982*.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. 2018. Deep learning for classical japanese literature. *CoRR*, abs/1812.01718.

Gari D Clifford, Chengyu Liu, Benjamin Moody, H Lehman Li-wei, Ikaro Silva, Qiao Li, AE Johnson, and Roger G Mark. 2017. Af classification from a short single lead ecg recording: the physionet/computing in cardiology challenge 2017. in 2017 computing in cardiology (cinc).

climate change act. 2008. Climate change act. *Department of Energy and Climate Change, Tech rep, UK*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Marcella Corduas and Domenico Piccolo. 2008. Time series clustering and classification by the autoregressive metric. *Computational statistics & data analysis*, 52(4):1860–1872.

A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril. 2019. Efficient keyword spotting using dilated convolutions and gating. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6351–6355.

Thomas M Cover and Joy A Thomas. 2012. *Elements of information theory*. John Wiley & Sons.

Kenneth Cukier. 2010. Data, data everywhere. *Economist*, 394(8671):3–5.

Wenyuan Dai, Yuqiang Chen, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2009. Translated learning: Transfer learning across different feature spaces. In *Advances in neural information processing systems*, pages 353–360.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

Sarah Darby. 2006. The effectiveness of feedback on energy consumption. *Environmental Change Institute*.

Amirhossein Esmaili Dastjerdi, Mohammad Kachuee, and Mahdi Shabany. 2017. Non-invasive blood pressure estimation using phonocardiogram. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4.

Leen De Baets, Chris Develder, Tom Dhaene, and Dirk Deschrijver. 2019. Detection of unidentified appliances in non-intrusive load monitoring using siamese neural networks. *International Journal of Electrical Power & Energy Systems*, 104:645–653.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8599–8603. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Michele D'Incecco, Stefano Squartini, and Mingjun Zhong. 2020. Transfer learning for non-intrusive load monitoring. *IEEE Transactions on Smart Grid*, 11(2):1419–1429.

Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552.

Bo Dong, Cristian Lumezanu, Yuncong Chen, Dongjin Song, Takehiko Mizoguchi, Haifeng Chen, and Latifur Khan. 2020. At the speed of sound: Efficient audio scene classification. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pages 301–305.

Roy Dong, Lillian Ratliff, Henrik Ohlsson, and S. Shankar Sastry. 2014. Fundamental limits of nonintrusive load monitoring. In *Proceedings of the 3rd International Conference on High Confidence Networked Systems*, HiCoNS '14, page 11–18, New York, NY, USA. Association for Computing Machinery.

Kenji Doya. 1993. Bifurcations of recurrent neural networks in gradient descent learning. *IEEE Transactions on neural networks*, 1(75):218.

Shiyu Duan, Shujian Yu, Yunmei Chen, and Jose C Principe. 2020. On kernel method–based connectionist models and supervised deep learning without backpropagation. *Neural computation*, 32(1):97–135.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Cian Eastwood and Christopher KI Williams. 2018. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*.

Dominik Egarter, Venkata Pathuri Bhuvana, and Wilfried Elmenreich. 2015a. Paldi: Online load disaggregation via particle filtering. *IEEE Transactions on Instrumentation and Measurement*, 64(2):467–477.

Dominik Egarter, Manfred Pöchacker, and Wilfried Elmenreich. 2015b. Complexity of power draws for load disaggregation. *CoRR*, abs/1501.02954.

Dominik Egarter, Manfred Pöchacker, and Wilfried Elmenreich. 2015c. Complexity of power draws for load disaggregation. *arXiv preprint arXiv:1501.02954*.

Abdel H El-Shaarawi and Walter W Piegorsch. 2001. *Encyclopedia of environmetrics*, volume 1. John Wiley and Sons.

Deniz Erdogmus and Jose C Principe. 2002. An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems. *IEEE Transactions on Signal Processing*, 50(7):1780–1786.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings.

Philippe Esling and Carlos Agon. 2012. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):1–34.

M. Esmaeilpour, P. Cardinal, and A. Lameiras Koerich. 2020. A robust approach for securing audio classification against adversarial attacks. *IEEE Transactions on Information Forensics and Security*, 15:2147–2159.

Amirhossein Esmaili, Mohammad Kachuee, and Mahdi Shabany. 2017. Nonlinear cuffless blood pressure estimation of healthy subjects using pulse transit time and arrival time. *IEEE Transactions on Instrumentation and Measurement*, 66(12):3299–3308.

Fabio Falcini and Giuseppe Lami. 2017. Deep learning in automotive: Challenges and opportunities. In *International Conference on Software Process Improvement and Capability Determination*, pages 279–288. Springer.

et al. Falcon, WA. 2019. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3.

Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. 1994. *Fast subsequence matching in time-series databases*, volume 23. ACM.

Gholamreza Farahani, Seyed Mohammad Ahadi, and Mohammad Mehdi Homayounpour. 2006. Robust feature extraction of speech via noise reduction in autocorrelation domain. In *International Workshop on Multimedia Content Representation, Classification and Security*, pages 466–473. Springer.

Jamil Fayyad, Mohammad A Jaradat, Dominique Gruyer, and Homayoun Najjaran. 2020. Deep learning sensor fusion for autonomous vehicle perception and localization: A review. *Sensors*, 20(15):4220.

Lei Feng, Senlin Shu, Zhuoyi Lin, Fengmao Lv, Li Li, and Bo An. 2021. Can cross entropy loss be robust to label noise? In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 2206–2212.

Adrian Filip et al. 2011. Blued: A fully labeled public dataset for event-based nonintrusive load monitoring research. In *2nd workshop on data mining applications in sustainability (SustKDD)*, page 2012.

Steven Firth, Tom Kane, Vanda Dimitriou, Tarek Hassan, Farid Fouchal, Michael Coleman, and Lynda Webb. 2017. Refit smart home dataset.

Ian Fischer. 2020. The conditional entropy bottleneck. *Entropy*, 22(9):999.

Elias Frentzos, Kostas Gratsias, and Yannis Theodoridis. 2007. Index-based most similar trajectory search. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 816–825. IEEE.

Jon Froehlich, Eric Larson, Sidhant Gupta, Gabe Cohn, Matthew Reynolds, and Shwetak Patel. 2011. Disaggregated end-use energy sensing for the smart grid. *IEEE Pervasive Computing*, 10(1):28–39.

Jingkun Gao, Suman Giri, Emre Can Kara, and Mario Bergés. 2014. Plaid: a public dataset of high-resoultion electrical appliance measurements for load identification research: demo abstract. In *proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 198–199. ACM.

Stanley M Garn. 1959. Radiographic atlas of skeletal development of the hand and wrist. *American journal of human genetics*, 11(3):282.

Mohammad M Ghassemi, Benjamin E Moody, Li-Wei H Lehman, Christopher Song, Qiao Li, Haoqi Sun, Roger G Mark, M Brandon Westover, and Gari D Clifford. 2018. You snooze, you win: the physionet/computing in cardiology challenge 2018. In *2018 Computing in Cardiology Conference (CinC)*, volume 45, pages 1–4. IEEE.

Nikolaos Virtsionis Gkalinikis, Christoforos Nalmpantis, and Dimitris Vrakas. 2020. Attention in recurrent neural networks for energy disaggregation. In *International Conference on Discovery Science*, pages 551–565. Springer.

David Gondek and Thomas Hofmann. 2003. Conditional information bottleneck clustering. In *3rd ieee international conference on data mining, workshop on clustering large data sets*, pages 36–42. Citeseer.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Ian Goodfellow, Honglak Lee, Quoc Le, Andrew Saxe, and Andrew Ng. 2009. Measuring invariances in deep networks. *Advances in neural information processing systems*, 22:646–654.

Benjamin Graham. 2014. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2008. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Daniel Greenfeld and Uri Shalit. 2020. Robust learning with the hilbert-schmidt independence criterion. In *International Conference on Machine Learning*, pages 3759–3768. PMLR.

Luigi Gresele, Julius Von Kügelgen, Vincent Stimper, Bernhard Schölkopf, and Michel Besserve. 2021. Independent mechanism analysis, a new concept? In *Advances in Neural Information Processing Systems*.

Arthur Gretton, Kenji Fukumizu, Choon Hui Teo, Le Song, Bernhard Schölkopf, Alexander J Smola, et al. 2007. A kernel statistical test of independence. In *Nips*, volume 20, pages 585–592. Citeseer.

Ulrich Greveler, Benjamin Justus, and Dennis Loehr. 2012. Forensic content detection through power consumption. In *2012 IEEE International Conference on Communications (ICC)*, pages 6759–6763.

Venkat Gudivada, Amy Apon, and Junhua Ding. 2017. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, 10(1):1–20.

Maya E Guglin and Deepak Thatai. 2006. Common errors in computer electrocardiogram interpretation. *International journal of cardiology*, 106(2):232–237.

Narendhar Gugulothu, Vishnu TV, Pankaj Malhotra, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2017. Predicting remaining useful life using time series embeddings based on recurrent neural networks. *arXiv preprint arXiv:1709.01073*.

Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361.

Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques*. Elsevier.

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. 2020. Contextnet: Improving convolutional neural networks for automatic speech recognition with global context. *arXiv preprint arXiv:2005.03191*.

George William Hart. 1992. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. 2018. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Shenda Hong, Yuxi Zhou, Junyuan Shang, Cao Xiao, and Jimeng Sun. 2020. Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review. *Computers in Biology and Medicine*, 122:103801.

Shenda Hong, Yuxi Zhou, Meng Wu, Junyuan Shang, Qingyun Wang, Hongyan Li, and Junqing Xie. 2019. Combining deep neural networks and engineered features for cardiac arrhythmia detection from ecg recordings. *Physiological measurement*, 40(5):054009.

Radu Horaud. 2020. Audio-visual speech enhancement using conditional variational auto-encoder. *IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING*, page 1.

Eric Horvitz, Johnson Apacible, Raman Sarin, and Lin Liao. 2005. Prediction, expectation, and surprise: methods, designs, and study of a deployed traffic forecasting service. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 275–283.

Bo Hu and Jose C Principe. 2021. Training a bank of wiener models with a novel quadratic mutual information cost function. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3150–3154. IEEE.

David H Hubel and Torsten N Wiesel. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

Patrick Huber, Alberto Calatroni, Andreas Rumsch, and Andrew Paice. 2021. Review on deep neural networks applied to low-frequency nilm. *Energies*, 14(9):2390.

Omer T. Inan, Laurent Giovangrandi, and Gregory T. A. Kovacs. 2006. Robust neural-network-based classification of premature ventricular contractions using wavelet transform and timing interval features. *IEEE Transactions on Biomedical Engineering*, 53(12):2507–2515.

Miquel India, Pooyan Safari, and Javier Hernando. 2019. Self multi-head attention for speaker recognition. *arXiv preprint arXiv:1906.09890*.

Inseon Jang, ChungHyun Ahn, Jeongil Seo, and Younseon Jang. 2017. Enhanced feature extraction for speech detection in media audio. In *INTERSPEECH*, pages 479–483.

Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. 2009. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE.

HS Jayanna and SR Mahadeva Prasanna. 2009. Analysis, feature extraction, modeling and testing techniques for speaker recognition. *IETE Technical Review*, 26(3):181–190.

Yangqing Jia, Chang Huang, and Trevor Darrell. 2012. Beyond spatial pyramids: Receptive field learning for pooled image features. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3370–3377. IEEE.

Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. 2018. Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. 2017. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pages 1724–1732. PMLR.

Linpeng Jin and Jun Dong. 2016. Classification of normal and abnormal ecg records using lead convolutional neural network and rule inference. *Science China Information Sciences*, 60:1–3.

Matthew J. Johnson and Alan S. Willsky. 2013. Bayesian nonparametric hidden semi-markov models. *J. Mach. Learn. Res.*, 14(1):673–701.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *International conference on machine learning*, pages 2342–2350. PMLR.

Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. 2018. Ecg heartbeat classification: A deep transferable representation. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 443–444.

Mohammad Kachuee, Mohammad Mahdi Kiani, Hoda Mohammadzade, and Mahdi Shabany. 2017. Cuffless blood pressure estimation algorithms for continuous health-care monitoring. *IEEE Transactions on Biomedical Engineering*, 64(4):859–869.

Matthias Kahl, Anwar Ul Haq, Thomas Kriechbaumer, and Hans-Arno Jacobsen. 2016. Whited-a worldwide household and industry transient energy data set. In *3rd International Workshop on Non-Intrusive Load Monitoring*.

Prashant V. Kamat. 2007. Meeting the clean energy demand: Nanostructure architectures for solar energy conversion. *The Journal of Physical Chemistry C*, 111(7):2834–2860.

Holger Kantz and Thomas Schreiber. 2004. *Nonlinear time series analysis*, volume 7. Cambridge university press.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Takekazu Kato, Hyun Sang Cho, Dongwook Lee, Tetsuo Toyomura, and Tatsuya Yamazaki. 2009. Appliance recognition from electric current signals for information-energy integrated network in home environments. In *Ambient Assistive Health and Wellness Management in the Heart of the City*, pages 150–157, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vasilios Katos, Dimitrios Vrakas, and Panagiotis Katsaros. 2011. A framework for access control with inference constraints. In *2011 IEEE 35th Annual Computer Software and Applications Conference*, pages 289–297.

Kenji Kawaguchi, Jiaoyang Huang, and Leslie Pack Kaelbling. 2019. Effect of depth and width on local minima in deep learning. *Neural computation*, 31(7):1462–1498.

Jack Kelly and William Knottenbelt. 2015a. Neural nilm: Deep neural networks applied to energy disaggregation. In *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*, pages 55–64.

Jack Kelly and William Knottenbelt. 2015b. The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes. *Scientific data*, 2(1):1–14.

Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001a. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286.

Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001b. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Sigmod Record*, 30(2):151–162.

Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386.

Eamonn J Keogh and Michael J Pazzani. 2000. A simple dimensionality reduction technique for fast similarity search in large time series databases. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 122–133. Springer.

Hyungsul Kim, Manish Marwah, Martin Arlitt, Geoff Lyon, and Jiawei Han. 2011. Unsupervised disaggregation of low frequency power measurements. In *Proceedings of the 2011 SIAM international conference on data mining*, pages 747–758. SIAM.

Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR.

Jihyun Kim, Howon Kim, et al. 2016. Classification performance using gated recurrent unit recurrent neural network on energy disaggregation. In *2016 international conference on machine learning and cybernetics (ICMLC)*, volume 1, pages 105–110. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Durk P Kingma, Tim Salimans, and Max Welling. 2015a. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583.

Durk P Kingma, Tim Salimans, and Max Welling. 2015b. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Tomi Kinnunen and Haizhou Li. 2010. An overview of text-independent speaker recognition: From features to supervectors. *Speech communication*, 52(1):12–40.

Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. 2016. Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, 63(3):664–675.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Andreas Kirsch, Clare Lyle, and Yarin Gal. 2020. Scalable training with information bottleneck objectives. In *Workshop Uncertainty & Robustness in Deep Learning at Int. Conf. on Machine Learning (ICML), online*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Christoph Klemenjak, Anthony Faustine, Stephen Makonin, and Wilfried Elmenreich. 2019. On metrics to assess the transferability of machine learning models in non-intrusive load monitoring. *arXiv preprint arXiv:1912.06200*.

Christoph Klemenjak, Stephen Makonin, and Wilfried Elmenreich. 2020. Towards comparability in non-intrusive load monitoring: on data and performance evaluation. In *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE.

Chun-Wa Ko, Jon Lee, and Maurice Queyranne. 1995. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691.

J Zico Kolter and Tommi Jaakkola. 2012. Approximate inference in additive factorial hmms with application to energy disaggregation. In *Artificial intelligence and statistics*, pages 1472–1482. PMLR.

J Zico Kolter and Matthew J Johnson. 2011. Redd: A public data set for energy disaggregation research. In *Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA*, volume 25, pages 59–62.

Lingpeng Kong, Cyprien de Masson d'Autume, Wang Ling, Lei Yu, Zihang Dai, and Dani Yogatama. 2019. A mutual information maximization perspective of language representation learning. *arXiv preprint arXiv:1910.08350*.

Flip Korn, Hosagrahar V Jagadish, and Christos Faloutsos. 1997. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Acm Sigmod Record*, volume 26, pages 289–300. ACM.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. 2004. Estimating mutual information. *Physical review E*, 69(6):066138.

Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Odysseas Krystalakos, Christoforos Nalmpantis, and Dimitris Vrakas. 2018. Sliding window approach for online energy disaggregation using artificial neural networks. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, pages 1–6.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. 2018. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*.

Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. 2018. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems*, pages 9017–9028.

Lamprini Kyrkou, Christoforos Nalmpantis, and Dimitris Vrakas. 2019. Imaging time-series for nilm. In *International Conference on Engineering Applications of Neural Networks*, pages 188–196. Springer.

Ying-Xun Lai, Chin-Feng Lai, Yueh-Min Huang, and Han-Chieh Chao. 2013. Multi-appliance recognition system with hybrid svm/gmm classifier in ubiquitous smart home. *Information Sciences*, 230:39–55. Mobile and Internet Services in Ubiquitous and Pervasive Computing Environments.

Henning Lange and Mario Bergés. 2016. The neural energy decoder: energy disaggregation by combining binary subcomponents. In *Proceedings of the 3rd international workshop on non-intrusive load monitoring*, volume 78, pages 5571–5589.

Anthony Larcher, Kong Aik Lee, Bin Ma, and Haizhou Li. 2014. Text-dependent speaker verification: Classifiers, databases and rsr2015. *Speech Communication*, 60:56–77.

Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. 2009. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1).

C. Laughman, Kwangduk Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong. 2003. Power signature analysis. *IEEE Power and Energy Magazine*, 1(2):56–63.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. 1990. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. 2016. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pages 464–472.

Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. 2007. Sparse deep belief net model for visual area v2. *Advances in neural information processing systems*, 20:873–880.

Jon Lee. 2006. Maximum entropy sampling. *Encyclopedia of Environmetrics*, 3.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*.

Simon Leglaive, Umut Şimşekli, Antoine Liutkus, Laurent Girin, and Radu Horaud. 2019. Speech enhancement with variational autoencoders and alpha-stable distributions. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 541–545. IEEE.

Athanasios Lentzas, Christoforos Nalmpantis, and Dimitris Vrakas. 2019. Hyperparameter tuning using quantum genetic algorithms. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1412–1416. IEEE.

Athanasios Lentzas and Dimitris Vrakas. 2019. Non-intrusive human activity recognition and abnormal behavior detection on elderly people: A review. *Artificial Intelligence Review*, pages 1–47.

Bowen Li, Kai Huang, Siang Chen, Dongliang Xiong, Haitian Jiang, and Luc Claesen. 2020a. Dfqf: Data free quantization-aware fine-tuning. In *Asian Conference on Machine Learning*, pages 289–304. PMLR.

Sheng Li, Dabre Raj, Xugang Lu, Peng Shen, Tatsuya Kawahara, and Hisashi Kawai. 2019. Improving transformer-based speech recognition systems with compressed structure and speech attributes augmentation. In *INTERSPEECH*, pages 4400–4404.

Yikuan Li, Shishir Rao, Jose Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. 2020b. Behrt: transformer for electronic health records. *Scientific reports*, 10(1):1–12.

Jian Liang, Simon K. K. Ng, Gail Kendall, and John W. M. Cheng. 2010. Load signature study—part i: Basic concept, structure, and methodology. *IEEE Transactions on Power Delivery*, 25(2):551–560.

Znaonui Liang, Gang Zhang, Jimmy Xiangji Huang, and Qmming Vivian Hu. 2014. Deep learning for healthcare decision making with emrs. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 556–559.

Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144.

Jessica Lin, Rohan Khade, and Yuan Li. 2012. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315.

Ralph Linsker. 1988. Self-organization in a perceptual network. *Computer*, 21(3):105–117.

Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Weifeng Liu, Puskal P Pokharel, and Jose C Principe. 2007. Correntropy: Properties and applications in non-gaussian signal processing. *IEEE Transactions on signal processing*, 55(11):5286–5298.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2019. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR.

Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. 2020. Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR.

P Loshin. 2016. Barclays replaces passwords with voice authentication.

David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Stephen Makonin and Fred Popowich. 2015. Nonintrusive load monitoring (nilm) performance evaluation. *Energy Efficiency*, 8(4):809–814.

Stephen Makonin, Fred Popowich, Lyn Bartram, Bob Gill, and Ivan V. Bajić. 2013. Ampds: A public dataset for load disaggregation and eco-feedback research. In *2013 IEEE Electrical Power Energy Conference*, pages 1–6.

Georgios Makridis, Philip Mavrepis, Dimosthenis Kyriazis, Ioanna Polychronou, and Stathis Kaloudis. 2020. Enhanced food safety through deep learning for food recalls prediction. In *International Conference on Discovery Science*, pages 566–580. Springer.

Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2017. Timenet: Pre-trained deep recurrent neural network for time series classification. *arXiv preprint arXiv:1706.08838*.

Simon Malinowski, Thomas Guyet, René Quiniou, and Romain Tavenard. 2013. 1d-sax: A novel symbolic representation for time series. In *International Symposium on Intelligent Data Analysis*, pages 273–284. Springer.

Miranti Indar Mandasari, ML McLaren, and David A van Leeuwen. 2011. Evaluation of i-vector speaker recognition systems for forensic application.

Alan Marchiori, Douglas Hakkarinen, Qi Han, and Lieko Earle. 2011. Circuit-level load monitoring for household energy management. *IEEE Pervasive Computing*, 10(1):40–48.

I. Martin-Morato, M. Cobos, and F. J. Ferri. 2018. On the robustness of deep features for audio event classification in adverse environments. In *2018 14th IEEE International Conference on Signal Processing (ICSP)*, pages 562–566.

Lukas Mauch and Bin Yang. 2015. A new approach for supervised power disaggregation by using a deep recurrent lstm network. In *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 63–67. IEEE.

David McAllester and Karl Stratos. 2020. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 875–884. PMLR.

Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez, Montse Gonzalez Arenas, Kanishka Rao, David Rybach, Ouais Alsharif, Haşim Sak, Alexander Gruenstein, Françoise Beaufays, et al. 2016. Personalized speech recognition on mobile devices. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5955–5959. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

David Minnen, Charles L Isbell, Irfan Essa, and Thad Starner. 2007. Discovering multivariate motifs using subsequence density estimation and greedy mixture learning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 615. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Michael D Morse and Jignesh M Patel. 2007. An efficient and accurate method for evaluating time series similarity. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 569–580.

Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Lara Nahma, Pei Chee Yong, Hai Huyen Dam, and Sven Nordholm. 2019. An adaptive a priori snr estimator for perceptual speech enhancement. *EURASIP Journal on Audio, Speech, and Music Processing*, 2019(1):1–20.

Satoshi Nakamura, Kazuo Hiyane, Futoshi Asano, Takeshi Yamada, and Takashi Endo. 1999. Data collection in real acoustical environments for sound scene understanding and hands-free speech recognition. In *Sixth European Conference on Speech Communication and Technology*.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. 2021. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003.

Christoforos Nalmpantis, Odysseas Krystalakos, and Dimitris Vrakas. 2018. Energy profile representation in vector space. In *SETN '18: 10th Hellenic Conference on Artificial Intelligence*. ACM.

Christoforos Nalmpantis, Athanasios Lentzas, and Dimitris Vrakas. 2019. A theoretical analysis of pooling operation using information theory. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1729–1733. IEEE.

Christoforos Nalmpantis, Nikolaos Virtsionis Gkalinikis, and Dimitris Vrakas. 2022a. Neural fourier energy disaggregation. *Sensors*, 22(2).

Christoforos Nalmpantis and Dimitris Vrakas. 2019a. Machine learning approaches for non-intrusive load monitoring: from qualitative to quantitative comparation. *Artificial Intelligence Review*, 52(1):217–243.

Christoforos Nalmpantis and Dimitris Vrakas. 2019b. Signal2vec: Time series embedding representation. In *International Conference on Engineering Applications of Neural Networks*, pages 80–90. Springer.

Christoforos Nalmpantis and Dimitris Vrakas. 2020. On time series representations for multi-label nilm. *NEURAL COMPUTING & APPLICATIONS*.

Christoforos Nalmpantis and Dimitris Vrakas. 2022. Variational feature pooling based on information bottleneck. In *under review*.

Christoforos Nalmpantis, Lazaros Vrysis, Danai Vlachava, Lefteris Papageorgiou, and Dimitris Vrakas. 2021. Entropy based feature pooling in speech command classification. In *Intelligent Computing*, pages 1083–1091. Springer.

Christoforos Nalmpantis, Lazaros Vrysis, Danai Vlachava, Lefteris Papageorgiou, and Dimitris Vrakas. 2022b. Noise invariant feature pooling for the internet of audio things. In *Multimedia Tools and Applications*.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Phuoc Nguyen, Truyen Tran, Nilmini Wickramasinghe, and Svetha Venkatesh. 2017. `Deepr`: A convolutional net for medical records. *IEEE Journal of Biomedical and Health Informatics*, 21(1):22–30.

Meindert Niemeijer, Bram van Ginneken, Casper A Maas, Frederik JA Beek, and Max A Viergever. 2003. Assessing the skeletal age from a hand radiograph: automating the tanner-whitehouse method. In *Medical Imaging 2003: Image Processing*, volume 5032, pages 1197–1205. International Society for Optics and Photonics.

George Nikolic, RL Bishop, and JB Singh. 1982. Sudden death recorded during holter monitoring. *Circulation*, 66(1):218–225.

Morteza Noshad, Yu Zeng, and Alfred O Hero. 2019. Scalable mutual information estimation using dependence graphs. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2962–2966. IEEE.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Amichai Painsky and Naftali Tishby. 2017. Gaussian lower bound for the information bottleneck limit. *J. Mach. Learn. Res.*, 18:213–1.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Francesca Paradiso, Federica Paganelli, Dino Giuli, and Samuele Capobianco. 2016. Context-based energy disaggregation in smart homes. *Future Internet*, 8(1):4.

Francesca Paradiso, Federica Paganelli, Antonio Luchetta, Dino Giuli, and Pino Castrogiovanni. 2013. Ann-based appliance recognition from low-frequency energy monitoring data. In *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE.

Sonal Parasrampuria and Jawanna Henry. 2019. Hospitals' use of electronic health records data, 2015-2017. *ONC Data Brief*, 46:1–13.

Oliver Parson, Siddhartha Ghosh, Mark Weal, and Alex Rogers. 2011. Using hidden markov models for iterative non-intrusive appliance monitoring.

Oliver Parson, Siddhartha Ghosh, Mark Weal, and Alex Rogers. 2012. Non-intrusive load monitoring using prior models of general appliance types. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 356–362. AAAI Press.

Oliver Parson, Siddhartha Ghosh, Mark Weal, and Alex Rogers. 2014. An unsupervised training method for non-intrusive appliance load monitoring. *Artificial Intelligence*, 217:1–19.

Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Razvan Pascanu, Yann N Dauphin, Surya Ganguli, and Yoshua Bengio. 2014. On the saddle point problem for non-convex optimization. *arXiv preprint arXiv:1405.4604*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR.

Ayesha Pervaiz, Fawad Hussain, Huma Israr, Muhammad Ali Tahir, Fawad Riasat Raja, Naveed Khan Baloch, Farruh Ishmanov, and Yousaf Bin Zikria. 2020. Incorporating noise robustness in speech command recognition by noise augmentation of training data. *Sensors*, 20(8):2326.

Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. 2016. Deepcare: A deep dynamic memory model for predictive medicine. In *Advances in Knowledge Discovery and Data Mining*, pages 30–41, Cham. Springer International Publishing.

Arnab Poddar, Md Sahidullah, and Goutam Saha. 2018. Speaker verification with short utterances: a review of challenges, trends and opportunities. *IET Biometrics*, 7(2):91–101.

Huseyin Polat, Wenliang Du, Sahin Renckes, and Yusuf Oysal. 2010. Private predictions on hidden markov models. *Artificial Intelligence Review*, 34(1):53–72.

Ryan Poplin, Avinash V Varadarajan, Katy Blumer, Yun Liu, Michael V McConnell, Greg S Corrado, Lily Peng, and Dale R Webster. 2018. Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, 2(3):158–164.

François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7-8):789–816.

Yuanbin Qu, Peihan Liu, Wei Song, Lizhen Liu, and Miaomiao Cheng. 2020. A text generation and prediction system: Pre-training on new corpora using bert and gpt-2. In *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 323–326.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

FA Rezaur rahman Chowdhury, Quan Wang, Ignacio Lopez Moreno, and Li Wan. 2018. Attention-based models for text-dependent speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5359–5363. IEEE.

Marc Ranzato, Christopher Poultney, Sumit Chopra, Yann LeCun, et al. 2007. Efficient learning of sparse representations with an energy-based model. *Advances in neural information processing systems*, 19:1137.

Chotirat Ratanamahatana, Eamonn Keogh, Anthony J Bagnall, and Stefano Lonardi. 2005. A novel bit level time series representation with implication of similarity search and clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 771–777. Springer.

Melani Rey, Volker Roth, and Thomas Fuchs. 2014. Sparse meta-gaussian information bottleneck. In *International Conference on Machine Learning*, pages 910–918. PMLR.

Karl Ridgeway and Michael C Mozer. 2018. Learning deep disentangled embeddings with the f-statistic loss. In *NeurIPS*.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

A. G. Ruzzelli, C. Nicolas, A. Schoofs, and G. M. P. O'Hare. 2010. Real-time recognition and profiling of appliances through a single electricity sensor. In *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Lalitha Sankar, S. Raj Rajagopalan, Soheil Mohajer, and H.V. Poor. 2013. Smart meter privacy: A theoretical framework. *IEEE Transactions on Smart Grid*, 4(2):837–846.

Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. 2019. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020.

Omid Sayadi, Mohammad B. Shamsollahi, and Gari D. Clifford. 2010. Robust detection of premature ventricular contractions using a wave-based bayesian framework. *IEEE Transactions on Biomedical Engineering*, 57(2):353–362.

Patrick Schäfer. 2015. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530.

Patrick Schäfer and Mikael Högqvist. 2012. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 516–527. ACM.

Patrick Schäfer and Ulf Leser. 2017. Fast and accurate time series classification with weasel. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 637–646. ACM.

Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010*, pages 92–101. Springer.

Jürg Schläpfer and Hein J Wellens. 2017. Computer-interpreted electrocardiograms: benefits and limitations. *Journal of the American College of Cardiology*, 70(9):1183–1192.

Jürgen Schmidhuber. 1992. Learning factorial codes by predictability minimization. *Neural computation*, 4(6):863–879.

Bernhard Schölkopf. 2019. Causality for machine learning. *arXiv preprint arXiv:1911.10500*.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. 2021. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634.

Ohad Shamir, Sivan Sabato, and Naftali Tishby. 2010. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2018. Efficient attention: Attention with linear complexities. *CoRR*, abs/1812.01243.

M Shetty. 2015. Icici bank to roll out voice authentication. *Aovailable at: wwwtimesofindiaindiatimescom/business/india-business/ICICI-Bank-to-rollout-voice-authentication/articleshow/46818823cms*, 6.

Michael C Shewry and Henry P Wynn. 1987. Maximum entropy sampling. *Journal of applied statistics*, 14(2):165–170.

Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. 2017. Deep ehr: a survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604.

Suwon Shon, Hao Tang, and James Glass. 2019. Voiceid loss: Speech enhancement for speaker verification. *arXiv preprint arXiv:1904.03601*.

Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

V Anne Smith, Jing Yu, Tom V Smulders, Alexander J Hartemink, and Erich D Jarvis. 2006. Computational inference of neural information flow networks. *PLoS computational biology*, 2(11):e161.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. 2018. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333. IEEE.

Roman A Solovyev, Maxim Vakhrushev, Alexander Radionov, Irina I Romanova, Aleksandr A Amerikanov, Vladimir Aliev, and Alexey A Shvets. 2020. Deep learning approaches for understanding simple speech commands. In *2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO)*, pages 688–693. IEEE.

Eduardo D Sontag et al. 1998. Vc dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences*, 168:69–96.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

D Srinivasan, WS Ng, and AC Liew. 2005. Neural-network-based signature recognition for harmonic source identification. *IEEE transactions on power delivery*, 21(1):398–405.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

DJ Strouse and David J Schwab. 2017. The deterministic information bottleneck. *Neural computation*, 29(6):1611–1630.

Cathie Sudlow, John Gallacher, Naomi Allen, Valerie Beral, Paul Burton, John Danesh, Paul Downey, Paul Elliott, Jane Green, Martin Landray, et al. 2015. Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS medicine*, 12(3):e1001779.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Nikolaos Symeonidis, Christoforos Nalmpantis, and Dimitris Vrakas. 2019. A benchmark framework to evaluate energy disaggregation solutions. In *Engineering Applications of Neural Networks*, pages 19–30, Cham. Springer International Publishing.

Seyed Mostafa Tabatabaei, Scott Dick, and Wilsun Xu. 2016. Toward non-intrusive load monitoring via multi-label classification. *IEEE Transactions on Smart Grid*, 8(1):26–40.

James Mourilyan Tanner. 1983. Assessment of skeletal maturity and prediction of adult height. *TW 2 Method*, pages 50–106.

Hans Henrik Thodberg, Sven Kreiborg, Anders Juul, and Karen Damgaard Pedersen. 2008. The bonexpert method for automated determination of skeletal maturity. *IEEE transactions on medical imaging*, 28(1):52–66.

Hans Henrik Thodberg, Sven Kreiborg, Anders Juul, and Karen Damgaard Pedersen. 2009. The bonexpert method for automated determination of skeletal maturity. *IEEE Transactions on Medical Imaging*, 28(1):52–66.

Sawitchaya Tippaya, Suchada Sitjongsataporn, Tele Tan, Masood Mehmood Khan, and Kosin Chamnongthai. 2017. Multi-modal visual features-based video shot boundary detection. *IEEE Access*, 5:12563–12575.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE.

Ljupčo Todorovski and Sašo Džeroski. 2003. Combining classifiers with meta decision trees. *Machine learning*, 50(3):223–249.

Truyen Tran, Tu Dinh Nguyen, Dinh Phung, and Svetha Venkatesh. 2015. Learning vector representation of medical objects via emr-driven nonnegative restricted boltzmann machines (enrbm). *Journal of Biomedical Informatics*, 54:96–105.

Frederik Träuble, Elliot Creager, Niki Kilbertus, Francesco Locatello, Andrea Dittadi, Anirudh Goyal, Bernhard Schölkopf, and Stefan Bauer. 2021. On disentangled representations learned from correlated data. In *International Conference on Machine Learning*, pages 10401–10412. PMLR.

Nikolaos Tsipas, Lazaros Vrysis, Charalampos Dimoulas, and George Papanikolaou. 2015. Mirex 2015: Methods for speech/music detection and classification. *Proc. Music information retrieval evaluation eXchange (MIREX)*.

Luca Turchet, György Fazekas, Mathieu Lagrange, Hossein S Ghadikolaei, and Carlo Fischione. 2020. The internet of audio things: state-of-the-art, vision, and challenges. *IEEE Internet of Things Journal*.

Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. 2014. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4052–4056. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR (Poster)*, 2(3):4.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015a. Grammar as a foreign language. *Advances in neural information processing systems*, 28:2773–2781.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Nikolaos Virtsionis-Gkalinikis, Christoforos Nalmpantis, and Dimitris Vrakas. 2021. Saed: self-attentive energy disaggregation. *Machine Learning*, pages 1–20.

Janani Viswanathan, N Saranya, and Abinaya Inbamani. 2021. Deep learning applications in medical imaging: Introduction to deep learning-based intelligent systems for medical applications. In *Deep Learning Applications in Medical Imaging*, pages 156–177. IGI Global.

Michail Vlachos, George Kollios, and Dimitrios Gunopulos. 2002. Discovering similar multidimensional trajectories. In *Proceedings 18th international conference on data engineering*, pages 673–684. IEEE.

Lazaros Vrysis, Iordanis Thoidis, Charalampos Dimoulas, and George Papanikolaou. 2020a. Experimenting with 1d cnn architectures for generic audio classification. In *Audio Engineering Society Convention 148*. Audio Engineering Society.

Lazaros Vrysis, Nikolaos Tsipas, Iordanis Thoidis, and Charalampos Dimoulas. 2020b. 1d/2d deep cnns vs. temporal feature integration for general audio classification. *Journal of the Audio Engineering Society*, 68(1/2):66–77.

Lazaros Vrysis, Nikolaos Tsipas, Iordanis Thoidis, and Charalampos Dimoulas. 2020c. Enhanced temporal feature integration in audio semantics. *Journal of the Audio Engineering Society*, 68(1/2):66–77.

Kun-Ching Wang. 2020. Robust audio content classification using hybrid-based smd and entropy-based vad. *Entropy*, 22(2):183.

Zhiguang Wang and Tim Oates. 2015. Imaging time-series to improve classification and imputation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Zhiyue J Wang. 2021. Probing an ai regression model for hand bone age determination using gradient-based saliency mapping. *Scientific reports*, 11(1):10610.

Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

Aleksander Wieczorek, Mario Wieser, Damian Murezzan, and Volker Roth. 2018. Learning sparse latent representations with the deep copula information bottleneck. *arXiv preprint arXiv:1804.06216*.

Matt Wytock and J Zico Kolter. 2014. Contextually supervised source separation with application to energy disaggregation. In *Twenty-eighth AAAI conference on artificial intelligence*.

Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

Yilun Xu, Peng Cao, Yuqing Kong, and Yizhou Wang. 2019. L_dmi: A novel information-theoretic loss function for training deep nets robust to label noise. In *NeurIPS*, pages 6222–6233.

Weizhong Yan and Lijie Yu. 2015. On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. In *Proceedings of the annual conference of the prognostics and health management society*.

Rui Yao, ZeQing Zeng, and Ping Zhu. 2016. A priori snr estimation and noise estimation for speech enhancement. *EURASIP journal on advances in signal processing*, 2016(1):1–15.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. 2018. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412.

Shujian Yu, Francesco Alesiani, Xi Yu, Robert Jenssen, and Jose C Principe. 2021. Measuring dependence with matrix-based entropy functional. *arXiv preprint arXiv:2101.10160*.

Gu yuan Lin, Shih chiang Lee, Jane Yung jen Hsu, and Wan rong Jih. 2010. Applying power meters for appliance recognition on the electric panel. In *2010 5th IEEE Conference on Industrial Electronics and Applications*, pages 2254–2259.

Thomas Zaslavsky. 1975. *Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes: Face-count formulas for partitions of space by hyperplanes*, volume 154. American Mathematical Soc.

Michael Zeifman. 2012. Disaggregation of home energy display data using probabilistic approach. *IEEE Transactions on Consumer Electronics*, 58(1):23–31.

Michael Zeifman and Kurt Roth. 2011. Nonintrusive appliance load monitoring: Review and outlook. *IEEE Transactions on Consumer Electronics*, 57(1):76–84.

Matthew D Zeiler and Rob Fergus. 2013. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

M. Zeng and N. Xiao. 2019. Effective combination of densenet and bilstm for keyword spotting. *IEEE Access*, 7:10767–10775.

Chaoyun Zhang, Mingjun Zhong, Zongzuo Wang, Nigel Goddard, and Charles Sutton. 2018a. Sequence-to-point learning with neural networks for non-intrusive load monitoring. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021a. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021b. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115.

Zixing Zhang, Jürgen Geiger, Jouni Pohjalainen, Amr El-Desoky Mousa, Wenyu Jin, and Björn Schuller. 2018b. Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Trans. Intell. Syst. Technol.*, 9(5).

Mingjun Zhong, Nigel Goddard, and Charles Sutton. 2015. Latent bayesian melding for integrating individual and population models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3618–3626.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. 2015. Object detectors emerge in deep scene cnns. In *ICLR*.

Yi-Tong Zhou and Rama Chellappa. 1988. Computation of optical flow using a neural network. In *ICNN*, pages 71–78.

Yingke Zhu, Tom Ko, David Snyder, Brian Mak, and Daniel Povey. 2018. Self-attentive speaker embeddings for text-independent speaker verification. In *Interspeech*, volume 2018, pages 3573–3577.

Tehseen Zia, Dietmar Bruckner, and Adeel Zaidi. 2011. A hidden markov model based procedure for identifying household electric loads. In *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, pages 3218–3223.

Ahmed Zoha, Alexander Gluhak, Muhammad Ali Imran, and Sutharshan Rajasegarar. 2012. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12):16838–16866.

A List of Publications

A.1 Publications in Journals

1. Nalmpantis, C. and Vrakas, D. (2018). *Machine learning approaches for non-intrusive load monitoring: from qualitative to quantitative comparation*. Artificial Intelligence Review, 1-27.

2. Nalmpantis, C. and Vrakas, D. (2020). *On time series representations for multi-label NILM*. Neural Computing and Applications, 1-16.

3. Gkalinikis, N. V., Nalmpantis, C. and Vrakas, D. (2021). *SAED: Self-Attentive Energy Disaggregation*. Machine Learning, 1-20.

4. Nalmpantis, C., Gkalinikis, N. V. and Vrakas, D. (2022). *Neural Fourier Energy Disaggregation*. Sensors 22, no. 2: 473.

5. Nalmpantis, C., Vrysis, L., Vlachava, D., Papageorgiou, L. and Vrakas, D. (2022) *Noise invariant feature pooling for the Internet of Audio Things*. Multimedia Tools and Applications.

A.2 Papers under Review in Journals

1. Nalmpantis, C. and Vrakas, D. *Variational Feature Pooling Based on the Principle of Information Bottleneck*. Under review in IEEE Transactions on Neural Networks and Learning Systems.

A.3 Publications in Conferences

1. Nalmpantis, C., Krystalakos, O. and Vrakas, D. (2018). *Energy profile representation in vector space*. In Proceedings of the 10th Hellenic Conference on Artificial Intelligence (p. 22). ACM

2. Krystalakos, O., Nalmpantis, C. and Vrakas, D. (2018). *Sliding Window Approach for Online Energy Disaggregation Using Artificial Neural Networks*. In Proceedings of the 10th Hellenic Conference on Artificial Intelligence (p. 7). ACM.

3. Nalmpantis C. and Vrakas D. (2019). *Signal2Vec: Time Series Embedding Representation*. In: Macintyre J., Iliadis L., Maglogiannis I., Jayne C. (eds) Engineering Applications of Neural Networks. EANN 2019. Communications in Computer and Information Science, vol 1000. Springer, Cham.

4. Kyrkou L., Nalmpantis C. and Vrakas D. (2019). *Imaging Time-Series for NILM*. In: Macintyre J., Iliadis L., Maglogiannis I., Jayne C. (eds) Engineering Applications of Neural Networks. EANN 2019. Communications in Computer and Information Science, vol 1000. Springer, Cham.

5. Symeonidis N., Nalmpantis C. and Vrakas D. (2019). *A Benchmark Framework to Evaluate Energy Disaggregation Solutions*. In: Macintyre J., Iliadis L., Maglogiannis I., Jayne C. (eds) Engineering Applications of Neural Networks. EANN 2019. Communications in Computer and Information Science, vol 1000. Springer, Cham.

6. Nalmpantis C., Lentzas A and Vrakas D. (2019). *A Theoretical Analysis of Pooling Operation Using Information Theory*. In 31st International Conference on Tools with Artificial Intelligence.

7. Gkalinikis, N. V., Nalmpantis, C. and Vrakas, D. (2020). *Attention in Recurrent Neural Networks for Energy Disaggregation*. In International Conference on Discovery Science (pp. 551-565). Springer, Cham.

8. Nalmpantis, C., Vrysis, L., Vlachava, D., Papageorgiou, L. and Vrakas, D. (2021). *Entropy Based Feature Pooling in Speech Command Classification*. In Intelligent Computing (pp. 1083-1091). Springer, Cham.

A.4 Other Publications (not included in this thesis)

1. Lentzas A, Nalmpantis C. and Vrakas D. (2019). *Hyperparameter tuning using Quantum Genetic Algorithms*. In 31st International Conference on Tools with Artificial Intelligence.

2. Gkalinikis, N. V., Nalmpantis, C. and Vrakas, D. (2022). *Torch-NILM: An effective deep learning toolkit for Non Intrusive Load Monitoring in Pytorch*. Energies

B ΠΕΡΙΛΗΨΗ ΔΙΑΤΡΙΒΗΣ

B.1 Εισαγωγή

Αυτή η διατριβή παρουσιάζει πρωτότυπη έρευνα στη διασταύρωση των περιοχών της μηχανικής μάθησης και των χρονοσειρών. Τα δεδομένα χρονοσειρών βρίσκονται παντού στη φύση ή στην καθημερινή μας ζωή. Κάθε ακολουθία δεδομένων όπου η σειρά από άποψη χρόνου έχει σημασία αποτελείται από μια χρονοσειρά. Αυτός ο τύπος δεδομένων έχει μοναδικές ιδιότητες. Τα δεδομένα χρονοσειρών προκύπτουν από τη δυναμική σύνθετων συστημάτων και είναι υψηλών διαστάσεων. Αυτές οι πτυχές αναδεικνύουν πολλές νέες προκλήσεις κατά την προσπάθεια εξαγωγής χρήσιμων πληροφοριών από δεδομένα χρονοσειρών χρησιμοποιώντας μεθόδους μηχανικής μάθησης.

Εστιάζουμε τόσο σε πρακτικές όσο και σε θεωρητικές πτυχές των μεθόδων μηχανικής μάθησης όταν ασχολούμαστε με δεδομένα χρονοσειρών. Οι τρεις βασικοί πυλώνες της συνεισφοράς μας περιλαμβάνουν νέες αρχιτεκτονικές μηχανικής μάθησης, αναπαραστάσεις ενσωμάτωσης χρονοσειρών και σχεδιασμό μοντέλων βαθιάς μάθησης με βάση τις θεωρητικές αρχές της πληροφορίας. Οι ερευνητικοί μας στόχοι καθοδηγούνται από τρία κοινά προβλήματα χρονοσειρών, όπως ο διαχωρισμός τυφλών πηγών, η ταξινόμηση και η παλινδρόμηση. Οι τομείς των δεδομένων που χρησιμοποιούνται σε αυτό το έργο καλύπτουν ενέργεια, ομιλία και υγεία. Όσον αφορά τον πρώτο τομέα, επιλέξαμε το πρόβλημα της μη παρεμβατικής παρακολούθησης φορτίου (NILM). Από τον τομέα της ομιλίας τα προβλήματα που αντιμετωπίζονται είναι η ταξινόμηση εντολών ομιλίας και η αναγνώριση ομιλητή. Τέλος, στον τομέα της υγείας πραγματοποιήσαμε πειράματα σχετικά με ηλεκτροκαρδιογραφήματα εξόρυξης (ECGs) και εκτίμηση οστικής ηλικίας. Μία από τις πρώτες συνεισφορές μας είναι μια ολοκληρωμένη διερεύνηση των προαναφερθέντων πραγματικών προβλημάτων, όπου εντοπίσαμε ερευνητικές ευκαιρίες.

Ο πρώτος πυλώνας των επιστημονικών μας συνεισφορών περιλαμβάνει νέες λύσεις μηχανικής εκμάθησης για το πρόβλημα του NILM. Προτείνουμε ένα νέο σύστημα μηχανικής μάθησης στοίβαξης, έναν μηχανισμό μάθησης μεταφοράς που χρησιμοποιεί τεχνικές απεικόνισης για χρονοσειρές και τρεις νέες νευρωνικές αρχιτεκτονικές. Οι μέθοδοί μας συγκρίνονται με τις υπάρχουσες και επιδεικνύουν αποτελέσματα αιχμής. Επιπλέον, αντιμετωπίζουμε διάφορα ζητήματα αξιολόγησης εισάγοντας νέα μέτρα/προσεγγίσεις αξιολόγησης και ανάπτυξη νέων πλαισίων αναφοράς.

Η δεύτερη σημαντική συμβολή αυτής της έρευνας αφορά τις αναπαραστάσεις χρονοσειρών. Η αναπαράσταση χρονοσειρών είναι μια ενεργή ερευνητική κατεύθυνση τις τελευταίες δεκαετίες. Με την άνοδο του Διαδικτύου των πραγμάτων (IoT) και την έκρηξη δεδομένων χρονοσειρών από διάφορους αισθητήρες, είναι προφανές ότι είναι επιτακτική μία αποτελεσματική αναπαράσταση χωρίς απώλεια χρήσιμων πληροφοριών. Προτείνουμε μια νέα αναπαράσταση ενσωμάτωσης χρονοσειρών, που ονομάζεται Signal2Vec. Το Signal2Vec είναι μια μη εποπτευόμενη προσέγγιση για τη μετατροπή μιας δεδομένης χρονοσειράς σε διανύσματα. Ο προτεινόμενος αλγόριθμος συγκρίνεται με τις υπάρχουσες μεθόδους αναπαράστασης χρονοσειρών από τον τομέα της επεξεργασίας σήματος και της θεωρίας του χάους. Οι εργασίες αξιολόγησης είναι μια ταξινόμηση πολλών κατηγοριών ενεργειακών δεδομένων και το πρόβλημα του NILM το οποίο θεωρείται ως προσέγγιση ταξινόμησης πολλαπλών ετικετών. Το Signal2Vec ξεπερνά τις άλλες μεθόδους και επιδεικνύει αποτελέσματα τελευταίας τεχνολογίας στις προαναφερθείσες εργασίες.

Η τρίτη επιστημονική μας προσθήκη αγγίζει δύο ερευνητικούς κλάδους, τη βαθιά μάθηση και τη θεωρία της πληροφορίας. Η θεωρία της πληροφορίας είναι ένα ισχυρό εργαλείο για τη μηχανική μάθηση από τα πρώτα βήματα του πεδίου. Σήμερα, οι αρχές της θεωρίας της πληροφορίας χρησιμοποιούνται για να γεφυρωθεί το χάσμα μεταξύ θεωρίας και πράξης στη βαθιά μάθηση. Παρά την επιτυχία της βαθιάς μάθησης σε πολλούς τομείς, ο σχεδιασμός τέτοιων μοντέλων βασίζεται στη διαδικασία της δοκιμής και του λάθους. Μια αυστηρή μαθηματική θεωρία υπερπαραμετροποιημένων μοντέλων εξακολουθεί να λείπει. Οι προσπάθειές μας επικεντρώνονται σε ένα δημοφιλές στοιχείο των σύγχρονων νευρωνικών αρχιτεκτονικών, που ονομάζεται pooling και περιλαμβάνει τόσο θεωρητικά όσο και εμπειρικά αποτελέσματα. Μελετάμε τις υπάρχουσες λειτουργίες pooling μέσα από το πρίσμα της θεωρίας πληροφοριών και προσπαθούμε να απαντήσουμε στο ερώτημα ποια συνάρτηση θα πρέπει να επιλεγεί κατά το σχεδιασμό ενός νέου νευρωνικού δικτύου με μια εργασία και ορισμένα δεδομένα. Στη συνέχεια, αναπτύσσουμε δύο νέες λειτουργίες pooling που βασίζονται στην αρχή της μέγιστης εντροπίας και στην αρχή της συμφόρησης πληροφορίας (IB). Τα θεωρητικά μας αποτελέσματα επαληθεύονται μέσω πειραμάτων. Οι προτεινόμενες μέθοδοι συγκρίνονται με άλλες και παρουσιάζουν ανώτερη απόδοση σε εργασίες ταξινόμησης και παλινδρόμησης, κατά την επίλυση προβλημάτων στους τομείς της ομιλίας και της υγείας.

Αυτή η επιστημονική εργασία είναι ένα βήμα προς πιο αξιόπιστες και αποτελεσματικές μεθόδους μηχανικής μάθησης για προβλήματα χρονοσειρών. Ελπίζουμε ότι τα ευρήματά μας θα αποτελέσουν κίνητρο για τους μελλοντικούς ερευνητές και θα χρησιμεύσουν ως εργαλεία για μηχανικούς σε βιομηχανικές εφαρμογές υψηλού αντίκτυπου.

B.2 Προβλήματα Χρονοσειρών

B.2.1 Το Πρόβλημα Επιμερισμού Ηλεκτρικής Ισχύος

Σε ένα σύστημα NILM, υπάρχει ένας έξυπνος μετρητής συνδεδεμένος με τον κεντρικό ηλεκτρικό εξοπλισμό μιας κατοικίας και συλλέγει δεδομένα κατανάλωσης ενέργειας. Οι εγγραφές περιλαμβάνουν τη συνολική κατανάλωση της κατοικίας. Στη συνέχεια, τα μοντέλα κατανομής ενέργειας μπορούν να εκτιμήσουν την κατανάλωση κάθε συσκευής που λειτουργεί στο σπίτι. Τα τρία βασικά βήματα της όλης διαδικασίας περιγράφονται από το πλαίσιο NILM (Carrie Armel et al., 2013; Burbano, 2015) και απεικονίζονται στην Εικ. B.2.1.

Το πρώτο βήμα του πλαισίου NILM είναι η απόκτηση δεδομένων. Αναφέρεται στη μεθοδολογία που συλλέγονται τα ενεργειακά αρχεία. Σε αυτό το βήμα η συσκευή που χρησιμοποιείται για τη λήψη των μετρήσεων παίζει σημαντικό ρόλο. Ένα από τα πιο σημαντικά χαρακτηριστικά των αισθητήρων είναι ο ρυθμός δειγματοληψίας που επηρεάζει την απόδοση των αλγορίθμων διαχωρισμού αργότερα. Ο εξοπλισμός είναι συνήθως έξυπνοι μετρητές που γίνονται πολύ δημοφιλείς σε πολλές χώρες. Αυτός ο εξοπλισμός έχει χαμηλό κόστος και είναι εύκολος στην εγκατάσταση. Όσον αφορά τις ιδιότητες των καταγεγραμμένων δεδομένων, οι πιο σημαντικές είναι ο τύπος ισχύος, η ανάλυση επιπέδου ισχύος και η συχνότητα δειγματοληψίας. Οι τύποι ισχύος είναι ενεργός και άεργος ισχύς και οι έξυπνοι μετρητές μπορούν να καταγράφουν έναν από αυτούς ή και τους δύο. Η δεύτερη ιδιότητα είναι η μικρότερη μέτρηση ισχύος που μπορούν να ανιχνεύσουν οι αισθητήρες. Η τελευταία ιδιότητα μπορεί να ταξινομηθεί σε δύο κύριες κατηγορίες χαμηλής συχνότητας που είναι έως 1 κΗζ και υψηλής συχνότητας που κυμαίνονται από 10 έως 100 ΜΗζ (Carrie Armel et al., 2013). Αυτά τα ακίνητα παίζουν

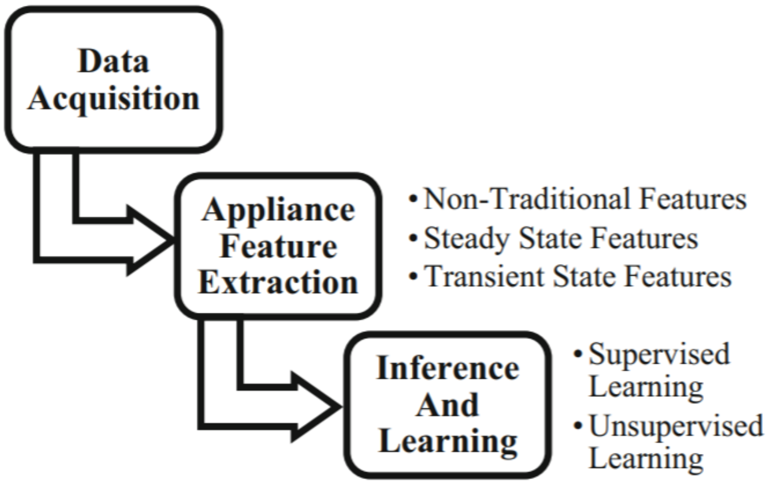


Figure B.2.1: Τα κύρια τρία βήματα του πλαισίου NILM.

σημαντικό ρόλο στην απόδοση των μοντέλων διαχωρισμού και στην ιδιωτικότητα των κατοίκων. Τα θεμελιώδη όρια ενός συστήματος ΝΙΛΜ διερευνώνται σε βάθος από το Dong et al. (2014). Οι συγγραφείς εξάγουν ένα ανώτερο όριο για την πιθανότητα διάκρισης σεναρίων με βάση ένα σύστημα NILM. Το άνω όριο εξαρτάται από τα χαρακτηριστικά των συλλεγόμενων ενεργειακών δεδομένων.

Το δεύτερο βήμα του πλαισίου NILM είναι η εξαγωγή χαρακτηριστικών της συσκευής. Η ιεραρχική ταξινόμηση αυτών των χαρακτηριστικών απεικονίζεται στο Σχ. B.2.2 και περιγράφεται λεπτομερώς από τους Zoha et al. (2012). Σύμφωνα με τους συγγραφείς, η ομάδα χαρακτηριστικών σταθερής κατάστασης απαιτεί χαμηλή συχνότητα δειγματοληψίας και συνιστά μια λύση χαμηλότερου κόστους. Ένα σύστημα που χρησιμοποιεί χαρακτηριστικά μεταβατικής κατάστασης τείνει να είναι πιο ακριβό επειδή απαιτεί συγκεκριμένο υλικό. Η απόδοση γενικά βελτιώνεται όταν εμπλέκονται συσκευές με επικαλυπτόμενα χαρακτηριστικά σταθερής κατάστασης. Λαμβάνοντας υπόψη τον υπάρχον εξοπλισμό στην αγορά, ένα σύστημα που χρησιμοποιεί χαρακτηριστικά σταθερής κατάστασης είναι πιο εφικτό.

Υπάρχει μια άλλη κατηγορία χαρακτηριστικών, που ονομάζεται μη παραδοσιακή. Μη παραδοσιακά θεωρούνται τα χαρακτηριστικά που δεν μπορούν να ομαδοποιηθούν στις άλλες δύο παραδοσιακές κατηγορίες. Αυτά περιλαμβάνουν πιο σύνθετα χαρακτηριστικά, όπως ο συνδυασμός παραδοσιακών, χαρακτηριστικά περιβάλλοντος, συμπεριφορά ή δείκτες των συσκευών. Οι Kim et al. (2011) προτείνουν ένα υπό όρους παραγοντικό κρυφό semi-Markov μοντέλο που χρησιμοποιεί μη παραδοσιακά χαρακτηριστικά. Αυτά τα χαρακτηριστικά περιλαμβάνουν την ώρα της ημέρας και την ημέρα της εβδομάδας, με βάση το γεγονός ότι η χρήση της συσκευής έχει χρονικά μοτίβα. Οι Wytock and Kolter (2014) προτείνουν μια εποπτευόμενη τεχνική με βάση τα συμφραζόμενα με συναρτήσεις ακτινικής βάσης σε θερμοκρασία και ώρα της ημέρας. Σύμφωνα με την προαναφερθείσα έρευνα, τα χρονικά χαρακτηριστικά δείχνουν ισχυρή ώθηση απόδοσης αλγορίθμων κατανομής ενέργειας. Μια προσέγγιση χωρίς επίβλεψη που λαμβάνει υπόψη τις αλληλεπιδράσεις συσκευών προτείνεται από τους Aiad and Lee (2016). Ζητήματα χαμηλής ποιότητας ισχύος μπορεί να οδηγήσουν σε διαταραχή των αρμονικών και των ρευμάτων παρεμβολής, λόγω της λειτουργίας άλλων συσκευών. Ενσωμάτωση αυτών των αλληλεπιδράσεων σε ένα παραγοντικό Hidden Markov Model, δείχνει βελτιωμένη ακρίβεια διαχωρισμού για τις συσκευές που εμπλέκονται σε αμοιβαίες αλληλεπιδράσεις. Οι Lange and Bergés (2016) χρησιμοποιούν έναν νευρωνικό αποκωδικοποιητή για την εξαγωγή δυαδικών υποσυστατικών. Τα επιμέρους εξαρτήματα χρησιμοποιούνται για να συμπεράνουμε την κατανάλωση ενέργειας των συσκευών.

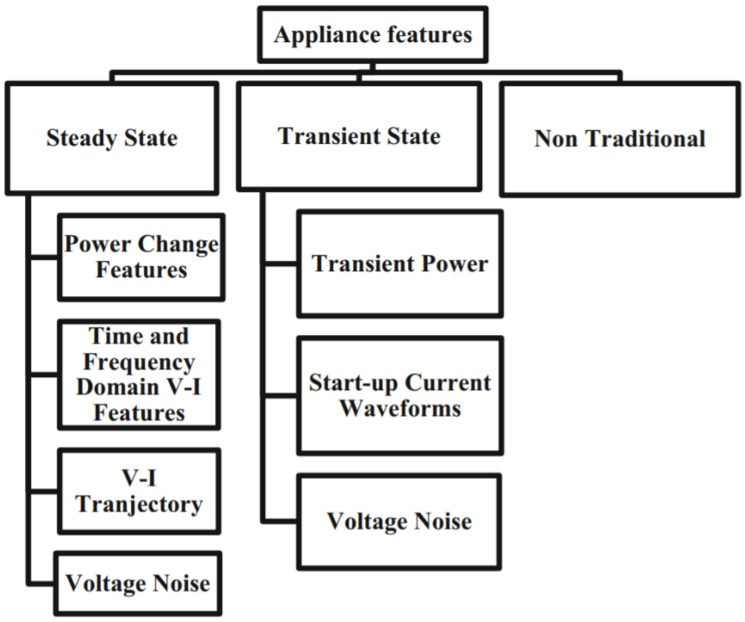


Figure B.2.2: Ιεραρχική ταξινόμηση των χαρακτηριστικών ηλεκτρικών συσκευών.

Το τρίτο βήμα του πλαισίου NILM αφορά το συμπέρασμα και τη μάθηση. Δεδομένων των εξαγόμενων χαρακτηριστικών του προηγούμενου βήματος, το μοντέλο διαχωρίζει το αρχικό σήμα στα στοιχεία του. Η τεχνολογία που χρησιμοποιείται για τη μοντελοποίηση του διαχωριστή βασίζεται σε δύο κύριες προσεγγίσεις: τη βελτιστοποίηση και τη μηχανική εκμάθηση. Στη βιβλιογραφία υπάρχουν πολλοί αλγόριθμοι βελτιστοποίησης που αντιμετωπίζουν το πρόβλημα της διάσπασης ισχύος (Baranski and J, 2003; Liang et al., 2010; Baranski and J, 2003). Η υπολογιστική πολυπλοκότητα του προβλήματος περιορίζει την πρακτική εφαρμογή προσεγγίσεων βελτιστοποίησης μόνο σε μικρό αριθμό συσκευών. Ένας μεγάλος αριθμός συσκευών και ένα πολύπλοκο περιβάλλον καθιστά πολύ δύσκολο το έργο της αντιστοίχισης υπογραφών.

Οι σύγχρονες προσεγγίσεις είναι κατά προσέγγιση λύσεις που βασίζονται σε μεθόδους μηχανικής μάθησης. Το ιδανικό σύστημα NILM που βασίζεται στη μηχανική μάθηση είναι ένας αλγόριθμος χωρίς επίβλεψη που μπορεί να προσδιορίσει τον αριθμό των ηλεκτρικών συσκευών από μόνος του και επομένως απαιτείται ελάχιστη είσοδος από τον χρήστη (Zeifman and Roth, 2011). Ωστόσο, η μάθηση χωρίς επίβλεψη είναι πολύ δύσκολη και οι εποπτευόμενες λύσεις αποδίδουν καλύτερα και είναι πιο πρακτικές, δεδομένου ότι υπάρχουν αρκετά επισημασμένα δεδομένα. Οι εποπτευόμενες λύσεις NILM περιλαμβάνουν μεθόδους όπως νευρωνικά δίκτυα (Ruzzelli et al., 2010; Srinivasan et al., 2005), support vector machines (SVM) (Kato et al., 2009; yuan Lin et al., 2010) hidden Markov model (HMM) (Zia et al., 2011), Μπεϋζιανές προσεγγίσεις (Marchiori et al., 2011; Zhong et al., 2015), ομαδοποίηση Hart (1992); Laughman et al. (2003) και συνδυασμός διαφόρων μεθόδων (Chang et al., 2011; Lai et al., 2013; **?**).

Οι περισσότερες σύγχρονες προσεγγίσεις βασίστηκαν σε κρυφά μοντέλα Markov, μέχρι την επανάσταση των βαθιών νευρωνικών δικτύων. Από το 2012, τα βαθιά νευρωνικά δίκτυα έχουν επιδείξει άνευ προηγουμένου απόδοση σε πολλά πεδία που καλύπτουν την όραση υπολογιστή (Krizhevsky and Hinton, 2009), την αναγνώριση ομιλίας (Deng et al., 2013), την επεξεργασία φυσικής γλώσσας (Collobert and Weston, 2008) και άλλα. Οι λύσεις βαθιάς μάθησης στον τομέα του NILM έχουν ξεπεράσει τις λύσεις που βασίζονται στο HMM και οι νέες νευρωνικές αρχιτεκτονικές δεν έχουν ακόμη διερευνηθεί.

B.2.2 Προβλήματα Αναγνώρισης Φωνής

Το Internet of Audio Things (IoAuT) είναι ένα αναδυόμενο υποπεδίο του Internet of Things (IoT) και έχει προσελκύσει πολλούς ερευνητές από διαφορετικούς κλάδους. Βρίσκεται στη διασταύρωση του διαδικτύου των πραγμάτων, της αναγνώρισης ήχου, της μηχανικής μάθησης και της αλληλεπίδρασης ανθρώπου-υπολογιστή (Turchet et al., 2020).

Από την ανθρώπινη σκοπιά, ο λόγος είναι ο εγγενής τρόπος επικοινωνίας. Ωστόσο, οι περισσότερες διεπαφές χρήστη από μηχανή σε άνθρωπο έχουν περιοριστεί σε πολύ περιορισμένες φωνητικές αλληλεπιδράσεις ή άλλες μεθόδους, όπως οθόνες αφής. Το IoAuT αντιμετωπίζει τις διεπιστημονικές προκλήσεις που πρέπει να επιλυθούν προκειμένου να ανοίξει ο δρόμος για νέες ευκαιρίες και εφαρμογές. Οι Turchet et al. (2020) παρουσιάζουν μια ταξινόμηση αυτών των προκλήσεων με βάση τις ακόλουθες κατηγορίες: συνδεσιμότητα, διαλειτουργικότητα και τυποποίηση, μηχανική ανάλυση περιεχομένου ήχου, συλλογή δεδομένων και αναπαράσταση περιεχομένου ήχου, υπολογιστική άκρη, συγχρονισμός, απόρρητο και ασφάλεια και σχεδιασμός ηχητικών πραγμάτων. Στο πλαίσιο της μηχανικής ανάλυσης περιεχομένου ήχου και υπολογιστών ακμών, οι κύριες προκλήσεις που αντιμετωπίζουμε είναι μεγάλος όγκος δεδομένων, παρουσία θορύβου, περιορισμοί υπολογιστικών πόρων και λανθάνουσα κατάσταση κατά την εξαγωγή συμπερασμάτων.

Το IoAuT θα παράγει τεράστιο όγκο δεδομένων ήχου που θα είναι δύσκολο να καθαριστούν. Η εκμάθηση από θορυβώδη δεδομένα είναι ένα κοινό πρόβλημα για πολλές εργασίες μηχανικής εκμάθησης. Ακόμη και όταν ένα μοντέλο εκπαιδεύεται με θορυβώδη δεδομένα, δεν υπάρχει καμία εγγύηση ότι θα είναι στιβαρό όταν αναπτυχθεί στον πραγματικό κόσμο (Martin-Morato et al., 2018; Bharitkar, 2019; Esmaeilpour et al., 2020). Επιπλέον, οι συσκευές IoT έχουν περιορισμένη αποθήκευση, γεγονός που καθιστά απαγορευτική την ανάπτυξη ενός υπερσύγχρονου νευρωνικού δικτύου. Τα σύγχρονα βαθιά νευρωνικά δίκτυα συνήθως εκπαιδεύονται και δοκιμάζονται χρησιμοποιώντας πολύ ισχυρές GPU και επομένως δεν είναι κατάλληλα για λειτουργία σε ενσωματωμένες συσκευές με περιορισμένους υπολογιστικούς και ενεργειακούς πόρους. Τέλος, για να δημιουργηθεί μια εφαρμογή IoAuT ώστε να ανταποκρίνεται στις απαιτήσεις των τελικών χρηστών, πρέπει να λειτουργεί με συνδεσιμότητα χαμηλού εύρους ζώνης και σε πραγματικό χρόνο.

Ο στόχος αυτής της έρευνας είναι να αναπτύξει αποτελεσματικές λύσεις βαθιάς μάθησης που να είναι ανθεκτικές σε εξωτερικούς παράγοντες σε εφαρμογές ταξινόμησης ήχου. Οι εργασίες για την αξιολόγηση των προτεινόμενων μοντέλων είναι η αναγνώριση ομιλητών και η αναγνώριση εντολών ομιλίας. Η πρώτη αφορά την αναγνώριση πέντε διαφορετικών ομιλητών χρησιμοποιώντας το δημοφιλές σύνολο δεδομένων "Speaker Recognition Dataset Prominent Leaders Speeches". Η τελευταία στοχεύει στον εντοπισμό δέκα φωνητικών εντολών από το δημοφιλές σύνολο δεδομένων "Google Speech Commands" που παρέχεται από την Google.

B.2.3 Εξόρυξη Δεδομένων Υγείας

Όλο και μεγαλύτερος όγκος δεδομένων παράγεται στην ιατρική βιομηχανία αυτές τις μέρες. Είναι δυνατό να αντληθούν χρήσιμες πληροφορίες για την υγεία ενός ασθενούς παρακολουθώντας μια ποικιλία φυσιολογικών σημάτων που εκπέμπονται από διάφορα όργανα. Η μελέτη των φυσιολογικών σημάτων απαιτεί τη χρήση εξειδικευμένων μεθόδων όπως η μηχανική μάθηση. Αυτή η διαδικασία έχει μια σειρά από προεκτάσεις και ζητήματα, ειδικά όταν εφαρμόζεται σε δεδομένα που αποκτήθηκαν από ιατρικές εξετάσεις ασθενών, οι οποίες είναι κατά κύριο λόγο χρονοσειρές (Aljawarneh et al., 2019).

Στον τομέα της υγειονομικής περίθαλψης ακριβείας, ακριβείς και εξατομικευμένες προβλέψεις, προλήψεις και παρεμβάσεις στοχεύουν στη βελτίωση της φροντίδας των ασθενών. Έχει σημειωθεί σημαντική πρόοδος προς εξατομικευμένες προβλέψεις στην καρδιαγγειακή ιατρική, την ακτινολογία, τη νευρολογία, τη δερματολογία και την οφθαλμολογία, για να αναφέρουμε μόνο μερικές, χάρη στις πρόσφατες εξελίξεις στη βαθιά μάθηση. Για παράδειγμα, οι Ardila et al. (2019) δείχνουν ότι η βαθιά μάθηση μπορεί να προβλέψει με ακρίβεια τον κίνδυνο καρκίνου του πνεύμονα από τις αξονικές τομογραφίες ενός ασθενούς. Οι Poplin et al. (2018) καταδεικνύουν ότι η βαθιά μάθηση μπορεί να προβλέψει με ακρίβεια μια ποικιλία παραγόντων καρδιαγγειακού κινδύνου από μια φωτογραφία βυθού του αμφιβληστροειδούς. Μαζί με τις προόδους στους αλγόριθμους βαθιάς μάθησης, ένας σημαντικός συντελεστής σε αυτήν την επιτυχία ήταν η τεράστια αύξηση του αριθμού πολυτροπικών βιοϊατρικών δεδομένων, συμπεριλαμβανομένων, ενδεικτικά, μεγάλων βάσεων δεδομένων όπως η UK Biobank (Sudlow et al., 2015) και δεδομένα υγείας που συλλέγονται τακτικά, όπως ως ηλεκτρονικά αρχεία υγείας (EHR) (Shickel et al., 2017).

Η υιοθέτηση συστημάτων EHR έχει επιταχυνθεί τα τελευταία χρόνια. το ποσοστό των νοσοκομείων στις Ηνωμένες Πολιτείες και το Ηνωμένο Βασίλειο που έχουν εφαρμόσει συστήματα EHR ξεπερνά πλέον το 84% και το 94%, αντίστοιχα (Parasrampuria and Henry, 2019). Ως αποτέλεσμα, τα συστήματα EHR ενός μεγάλου ιατρικού οργανισμού είναι πλέον πιθανό να συλλέγουν δεδομένα από εκατομμύρια άτομα κατά τη διάρκεια αρκετών ετών. Το EHR κάθε ατόμου μπορεί να ενσωματώσει δεδομένα από ποικίλες πηγές και ως εκ τούτου να περιλαμβάνει «έννοιες» όπως διαγνώσεις, παρεμβάσεις, εργαστηριακές εξετάσεις και κλινικές αφηγήσεις. Κάθε παρουσία μιας έννοιας μπορεί να αντιπροσωπεύει ένα μόνο ή πολλά σημεία δεδομένων. Για παράδειγμα, μια μεμονωμένη νοσηλεία μπορεί να δημιουργήσει χιλιάδες σημεία δεδομένων για ένα άτομο, ενώ μια διάγνωση μπορεί να αντιπροσωπεύει ένα μόνο σημείο δεδομένων. Αυτό καθιστά τα EHR μεγάλης κλίμακας μια απαράμιλλη πηγή πληροφοριών και δεδομένων για την εκπαίδευση μοντέλων μηχανικής εκμάθησης που απαιτούν δεδομένα (Li et al., 2020b).

Η κλασσική μέθοδος έρευνας EHR αντιπροσωπεύει τα άτομα ως πολυδιάστατα διανύσματα, που συχνά ονομάζονται χαρακτηριστικά. Αυτή η μέθοδος εξαρτάται σε μεγάλο βαθμό από την τεχνογνωσία των ανθρώπων στον τομέα. Οι πρόσφατες εξελίξεις στη βαθιά μάθηση έχουν αναπτύξει μοντέλα που μπορούν να μάθουν αναπαραστάσεις από ακατέργαστα ή ελάχιστα επεξεργασμένα δεδομένα, με ελάχιστη ή καθόλου συμβολή ειδικών. Αυτό επιτυγχάνεται μέσω μιας ιεραρχικής δομής επιπέδων, καθένα από τα οποία χρησιμοποιεί έναν μεγάλο αριθμό απλών γραμμικών και μη γραμμικών πράξεων για να χαρτογραφήσει τις αντίστοιχες εισόδους τους σε μια αναπαράσταση. Η πρόοδος από επίπεδο σε επίπεδο αναμένεται να οδηγήσει σε μια τελική αναπαράσταση που είναι χρήσιμη για μια δεδομένη εργασία.

Οι Liang et al. (2014) έδειξαν ότι τα βαθιά νευρωνικά δίκτυα ξεπερνούν τα SVM και τα δέντρα αποφάσεων που αναμιγνύονται με τη μη αυτόματη μηχανική χαρακτηριστικών σε μια ποικιλία εργασιών πρόβλεψης και συνόλων δεδομένων. Κατά την πρόβλεψη του κινδύνου αυτοκτονίας ατόμων μέσω του EHR, οι Tran et al. (2015) υποστήριξαν τη χρήση περιορισμένων μηχανών Boltzmann (RBM) για την εκμάθηση μιας κατανεμημένης αναπαράστασης του EHR, η οποία αποδείχθηκε ότι υπερέχει της εξαγωγής χαρακτηριστικών από ανρθρώπους.

Αυτές οι πρώιμες ερευνητικές εργασίες σχετικά με την εφαρμογή της βαθιάς μάθησης στο EHR απέτυχαν να λάβουν υπόψη τις αποχρώσεις των δεδομένων EHR. Σε μια προσπάθεια να το αντιμετωπίσει, οι Nguyen et al. (2017) ανέπτυξαν το Deepr, ένα μοντέλο συνελικτικού νευρωνικού δικτύου για την πρόβλεψη της πιθανότητας επανεισδοχής. Οι συγγραφείς θεώρησαν το ιατρικό ιστορικό κάποιου ως μια σειρά από έννοιες και πρόσθεσαν μια συγκεκριμένη λέξη μεταξύ κάθε σειράς διαδοχικών επισκέψεων για να υποδείξουν τη διαφορά ώρας. Οι Choi et al. (2016b) πρότειναν ένα μοντέλο ρηχού επαναλαμβανόμενου νευρωνικού δικτύου για την πρόβλεψη των διαγνώσεων και των φαρμάκων που είναι πιθανό να συναντηθούν στην επόμενη επίσκεψη. Και οι δύο προσεγγίσεις χρησιμοποίησαν κάποια μορφή ενσωμάτωσης για να μετατρέψουν μη αριθμητικές ιατρικές έννοιες σε έναν αλγεβρικό χώρο στον οποίο θα μπορούσαν να λειτουργήσουν τα μοντέλα ακολουθιών.

Μία από τις επόμενες βελτιώσεις που έγιναν στο EHR στοχεύουν την αποτύπωση μακροπρόθεσμων εξαρτήσεων μεταξύ γεγονότων. Μερικά αντιπροσωπευτικά παραδείγματα περιλαμβάνουν βασικές διαγνώσεις όπως ο διαβήτης που μπορεί να παραμείνει παράγοντας κινδύνου σε όλη τη ζωή ενός ατόμου, ακόμη και δεκαετίες μετά την πρώτη εμφάνισή του. Οι Pham et al. (2016) ανέπτυξαν το DeepCare, μια αρχιτεκτονική μακροπρόθεσμης μνήμης (LSTM) σε συνδυασμό με τον μηχανισμό Attention που ξεπέρασε τις παραδοσιακές προσεγγίσεις μηχανικής μάθησης, το LSTM και το απλό RNN σε εφαρμογές όπως η πρόβλεψη διαβήτη. Οι Choi et al. (2016a) καθιέρωσαν ένα μοντέλο που βασίζεται στον μηχανισμό προσοχής αντίστροφου χρόνου για την εδραίωση προηγούμενων επισκέψεων με αντίκτυπο χρησιμοποιώντας μια αρχιτεκτονική RNN με το όνομα RETAIN, για την πρόβλεψη καρδιακής ανεπάρκειας. Το RETAIN ξεπέρασε την πλειονότητα των μοντέλων τη στιγμή της δημοσίευσής του και χρησίμευσε ως καλό σημείο εκκίνησης για την ιατρική έρευνα βαθιάς μάθησης (Ayala Solares et al., 2020).

B.3 Λύσεις Μηχανικής Μάθησης για το Πρόβλημα του NILM

Αυτή η ενότητα παρουσιάζει τις κύριες συνεισφορές μας στην επίλυση του προβλήματος του NILM. Αρχικά παρουσιάζεται ένα νέο πλαίσιο αναφοράς των συστημάτων NILM αναλύοντας τη σημασία μιας τυποποιημένης διαδικασίας αξιολόγησης. Το πλαίσιο εισήχθη για πρώτη φορά στη μελέτη μας (Symeonidis et al., 2019) και χρησιμοποιείται για την αξιολόγηση συστημάτων NILM επόμενης γενιάς. Παράλληλα, εξετάζονται μέθοδοι μηχανικής εκμάθησης στοίβαξης για την επίλυση του προβλήματος της διάσπασης ισχύος. Στη συνέχεια, υπάρχει μια σειρά από πρωτότυπες αρχιτεκτονικές νευρωνικών δικτύων που αντιμετωπίζουν αυτό το πρόβλημα ως εργασία παλινδρόμησης και προβλέποντας την κατανάλωση ενέργειας μεμονωμένων συσκευών.

Η πρώτη νευρωνική αρχιτεκτονική, η οποία υπήρξε ένα από τα μοντέλα τελευταίας τεχνολογίας, ονομάζεται online-GRU (Krystalakos et al., 2018). Αυτή η μελέτη εισήγαγε την προσέγγιση κυλιόμενου παραθύρου σε αντίθεση με προηγούμενες μεθόδους που χρησιμοποιούσαν προσεγγίσεις από αλληλουχία σε ακολουθία ή από ακολουθία σε σημείο. Επιπλέον, αποδίδει ισάξια ή καλύτερα σε σύγκριση με άλλα μοντέλα τελευταίας τεχνολογίας και έχει λιγότερες από τις μισές παραμέτρους ενός προηγούμενου επαναλαμβανόμενου νευρωνικού δικτύου. Στη συνέχεια, η έρευνά μας επικεντρώθηκε στη μεταφορά γνώσης από άλλους τομείς όπως η αναγνώριση εικόνων και ήταν η πρώτη φορά που οι τεχνικές χρονοσειρών απεικόνισης εισήχθησαν στην έρευνα NILM (Kyrkou et al., 2019).

Τα επόμενα βήματά μας προέρχονται από ένα βασικό πρακτικό πρόβλημα κατά την ανάπτυξη συστημάτων NILM σε πραγματικό χρόνο σε συσκευές ενσωμάτωσης. Πράγματι, τέτοιες συσκευές έχουν περιορισμένους υπολογιστικούς και αποθηκευτικούς πόρους και τα σύγχρονα μοντέλα βαθιάς εκμάθησης τείνουν να έχουν τεράστιο μέγεθος και απαιτούν μεγάλη υπολογιστική ισχύ. Το μοντέλο Self-attentive Energy disagregator (SAED) (Gkalinikis et al., 2020; Virtsionis-Gkalinikis et al., 2021) είναι ένα νέο νευρωνικό δίκτυο που έχει πολύ μικρό μέγεθος και είναι πολύ γρήγορο στην εξαγωγή προβλέψεων.

Η τελευταία μας συνεισφορά, σε αυτήν την ενότητα, είναι ένα υπερσύγχρονο νευρωνικό δίκτυο που αποδίδει ισοδύναμα με άλλα τελευταίας τεχνολογίας μοντέλα διαχωρισμού ενέργειας, αλλά έχει πολύ μικρότερο μέγεθος και επιδεικνύει πολύ υψηλή ταχύτητα συμπερασμάτων (**?**). Στη συνέχεια παρουσιάζεται μία εκτεταμένη περίληψη των σημαντικότερων από τις επιστημονικές συνεισφορές που αναφέρθηκαν παραπάνω.

Device	Category1		Category2		Category3		Category4	
	Train	Test	Train	Test	Train	Test	Train	Test
DW	1	1	1	2	1,2	5	1,2	2
FZ	1	1	1	2	1,2,4	5	1,2,4	3
KT	1	1	1	5	1,2,4	5	1,2,4	2
MW	1	1	1	2	1,2	5	1,2	1
WM	1	1	1	4	1,5	2	1,5	3

Table B.3.1: Κτίρια που χρησιμοποιούνται για εκπαίδευση και δοκιμές. Στις κατηγορίες 1-3, το UK-DALE χρησιμοποιήθηκε τόσο για εκπαίδευση όσο και για δοκιμές. Στην κατηγορία 4, το UK-DALE χρησιμοποιήθηκε μόνο για εκπαίδευση. Για τη δοκιμή χρησιμοποιήθηκε το DW και το KT από το σύνολο REFIT, ενώ το REDD χρησιμοποιήθηκε για τη δοκιμή FZ, MW και WM.

B.3.1 Μέθοδολογία και Σύνολα Δεδομένων

Όσον αφορά την αξιολόγηση και σύγκριση του προτεινόμενου μοντέλου με τα υπάρχοντα, η διαδικασία ευθυγραμμίζεται με το πλαίσιο αναφοράς που προτείνεται από στην εργασία μας (Symeonidis et al., 2019) και περιλαμβάνει τέσσερα βασικά σενάρια. Στην πρώτη περίπτωση τα μοντέλα εκπαιδεύονται και δοκιμάζονται στο ίδιο σπίτι σε διαφορετικές χρονικές περιόδους. Τα δεδομένα της δοκιμής είναι χρονολογικά μετά τα δεδομένα εκπαίδευσης. Ως εκ τούτου, αναμένεται μικρή ή καθόλου μετατόπιση κατανομής πιθανότητας. Τα μοντέλα με χαμηλή απόδοση σε αυτά τα πειράματα θεωρούνται αδύναμα επειδή αυτή είναι η πιο εύκολη περίπτωση αξιολόγησης. Στο δεύτερο σενάριο αναμένεται μια μετατόπιση κατανομής των δεδομένων, επειδή τα δεδομένα δοκιμής ανήκουν σε διαφορετικούς οίκους που δεν φαίνονται κατά τη διάρκεια της εκπαίδευσης. Τα διαφορετικά πρότυπα κατανάλωσης ενέργειας μπορούν να αποδοθούν στις συνήθειες των κατοίκων και στην ποικιλία των συσκευών. Το τρίτο και το τέταρτο σενάριο εξετάζουν τις δυνατότητες εκμάθησης των μοντέλων σε πολλά κτίρια και δοκιμές στο ίδιο και διαφορετικό σύνολο δεδομένων. Οι τέσσερις κατηγορίες του δείκτη αναφοράς συνοψίζονται ως εξής: μεμονωμένο κτίριο NILM, μάθηση και γενίκευση μεμονωμένου κτιρίου στο ίδιο σύνολο δεδομένων, εκμάθηση πολλαπλών κτιρίων και γενίκευση στο ίδιο σύνολο δεδομένων και γενίκευση σε διαφορετικό σύνολο δεδομένων. Ο πίνακας **??** παρουσιάζει τις λεπτομέρειες των συνόλων δεδομένων και των αντίστοιχων κατοικιών που επιλέγονται για κάθε κατηγορία του πλαισίου συγκριτικής αξιολόγησης.

Τα πειράματα αυτής της εργασίας βασίζονται σε τρία δημόσια σύνολα δεδομένων: UK-DALE (Kelly and Knottenbelt, 2015b), REDD (Kolter and Johnson, 2011) και REFIT (Firth et al., 2017). Το UK-DALE και το REFIT περιέχουν δεδομένα από το Ηνωμένο Βασίλειο και το REDD από τις ΗΠΑ. Το REFIT περιλαμβάνει 20 σπίτια και μια ευρύτερη γκάμα συσκευών. Πέντε οικιακές συσκευές χρησιμοποιούνται για την αξιολόγηση των μοντέλων διαχωρισμού: πλυντήριο πιάτων (DW), ψυγείο (FZ), βραστήρας (KT), φούρνος μικροκυμάτων (MW) και πλυντήριο ρούχων (WM).

B.3.2 Μετρήσεις Αξιολόγησης

Οι πιο συνηθισμένες μετρήσεις κατά την αξιολόγηση της απόδοσης ενός συστήματος NILM είναι η μετρική F_1 και το μέσο απόλυτο σφάλμα (MAE). Η βαθμολογία F_1 αντιστοιχεί στην ανίχνευση του εάν μια συγκεκριμένη συσκευή καταναλώνει ενέργεια. Υπολογίζεται χρησιμοποιώντας την Εξίσωση (B.1) που είναι ο αρμονικός μέσος όρος Ακρίβειας και Ανάκλησης. Η Ακρίβεια και η Ανάκληση περιγράφονται στις Εξισώσεις (B.2) και (B.3), αντίστοιχα. Η μετρική MAE μετρά πόσο αποκλίνει η προβλεπόμενη κατανάλωση ενέργειας από την πραγματική. Μετριέται σε Watts και η εξίσωσή της περιγράφεται από την (B.4) όπου T είναι το μήκος της προβλεπόμενης ακολουθίας, y_t' η εκτιμώμενη κατανάλωση ηλεκτρικής ενέργειας και y_t η πραγματική τιμή της ενεργού κατανάλωσης τη στιγμή t.

$$F_1 = 2\frac{Precision \times Recall}{Precision + Recall} \tag{B.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{B.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{B.3}$$

$$MAE = \frac{1}{T}\sum |y_t' - y_t| \tag{B.4}$$

Το πλαίσιο αναφοράς που χρησιμοποιείται σε αυτή την εργασία περιλαμβάνει δοκιμές και σε δεδομένα που δεν έχει δει το μοντέλο. Για να ποσοτικοποιηθούν οι δυνατότητες γενίκευσης των μοντέλων, χρησιμοποιείται η μέτρηση απώλειας γενίκευσης (G-loss) Klemenjak et al. (2019). Η απώλεια G υπολογίζεται από τις εξισώσεις (B.5) ή (B.6), ανάλογα με το αν η βασική μέτρηση είναι F_1 ή MAE. Ο δείκτης *u* σημαίνει unseen και *s* για γνωστά δεδομένα. Όσο μεγαλύτερη είναι η απώλεια G τόσο χειρότερη είναι η γενίκευση. Η μέση απόδοση γενίκευσης μπορεί να υπολογιστεί χρησιμοποιώντας τη μέση απώλεια γενίκευσης (MGL) σύμφωνα με την Εξίσωση (B.7). Επιπλέον, η μέση βαθμολογία F_1 και η μέση απώλεια λαμβάνονται επίσης υπόψη χρησιμοποιώντας τις Εξισώσεις (B.8) και (B.9).

$$G - loss = 100(1 - \frac{F1_u}{F1_s}) \tag{B.5}$$

$$G - loss = 100(\frac{MAE_u}{MAE_s} - 1) \tag{B.6}$$

$$MGL = \frac{1}{N}\sum_i^N G - loss_i \tag{B.7}$$

$$AUH = \frac{1}{N}\sum_i^N F1_{ui} \tag{B.8}$$

$$EUH = \frac{1}{N}\sum_i^N MAE_{ui} \tag{B.9}$$

B.3.3 Αρχιτεκτονικές Νευρωνικών Δικτύων

Το μοντέλο Sequence-to-point (S2P) είναι ένα συνελικτικό νευρωνικό δίκτυο που προτείνεται από τους ?. Η αρχική αρχιτεκτονική του δικτύου δέχεται ως είσοδο μια ακολουθία μεγέθους 599. Αποτελείται από πέντε επίπεδα συνέλιξης και τη συνάρτηση μη γραμμικής ενεργοποίησης ReLU. Μια γραμμική συνάρτηση ενεργοποίησης ορίζει το τελικό επίπεδο. Οι λεπτομέρειες των στρωμάτων παρουσιάζονται στο σχ. B.3.1.

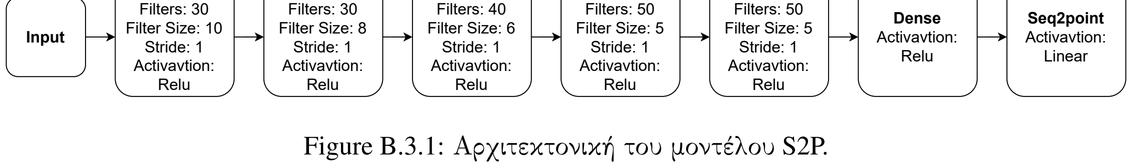


Figure B.3.1: Αρχιτεκτονική του μοντέλου S2P.

Το παράθυρο GRU (WGRU) είναι ένα από τα πρώτα νευρωνικά δίκτυα που φτιάχτηκε στα πλαίσια αυτής της διατριβής από τους Krystalakos et al. (2018) και το κύριο χαρακτηριστικό του είναι το επαναλαμβανόμενο επίπεδο GRU (Cho et al., 2014b). Πριν από την έξοδο, υπάρχει ένα συνελικτικό στρώμα, δύο διπλής κατεύθυνσης επίπεδα GRU και ένα πυκνό στρώμα. Η προσέγγιση εγκατάλειψης χρησιμοποιείται για την αποφυγή υπερβολικής προσαρμογής. Το Srivastava et al. (2014) χρησιμοποιείται μεταξύ των επιπέδων. Η είσοδος είναι ένα συρόμενο παράθυρο με ματιά. Το σχ. B.3.2 απεικονίζει τις αρχιτεκτονικές λεπτομέρειες.

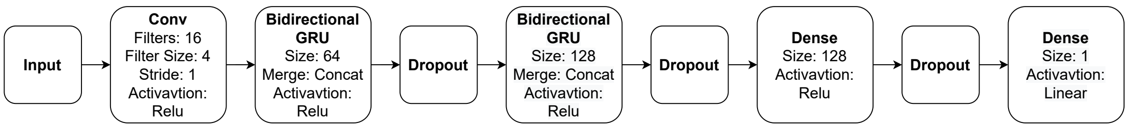


Figure B.3.2: Αρχιτεκτονική του μοντέλου WGRU.

Το νευρωνικό δίκτυο Self-attentive energy disaggregator (SAED) βασίζεται στον μηχανισμό attention και αναπτύχθηκε στα πλαίσια της διατριβής αυτής από τους Gkalinikis et al. (2020). Είναι ένα νευρωνικό δίκτυο που είναι υπολογιστικά αποδοτικό. Μπορεί να εκπαιδευτεί έως και ξ7,5 πιο γρήγορα από το WGRU και να κάνει προβλέψεις έως και ξ6,5 πιο γρήγορα. Ένα συνελικτικό στρώμα ακολουθείται από τον μηχανισμό attention στην αρχιτεκτονική. Ο μηχανισμός αυτός χωρίζεται σε δύο τύπους: additive attention και dot attention. Στη συνέχεια, υπάρχει ένα αμφίδρομο στρώμα GRU, ακολουθούμενο από ένα πυκνό στρώμα. Ολόκληρη η αρχιτεκτονική απεικονίζεται στο Σχ. B.3.3.

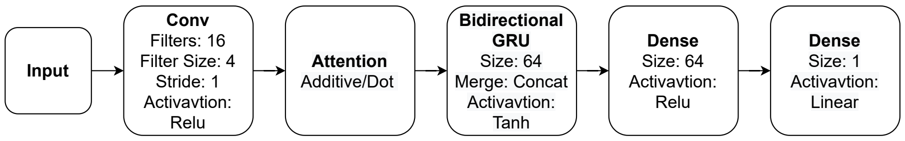
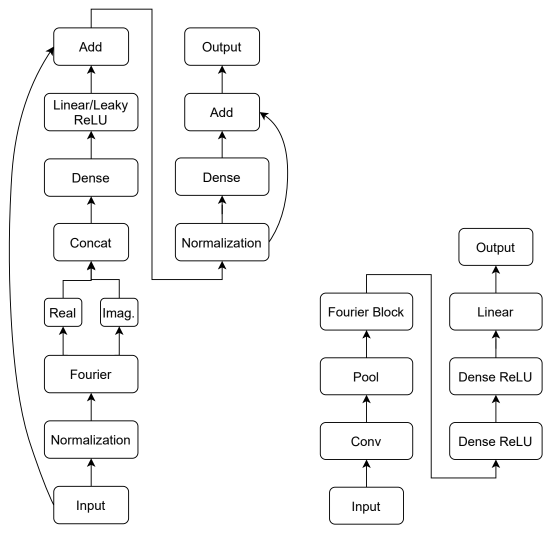


Figure B.3.3: Αρχιτεκτονική του μοντέλου SAED.

Το νευρωνικό δίκτυο Neural Fourier Energy Discagregation (NFED) είναι μια μοναδική νευρωνική αρχιτεκτονική που προτείνεται σε αυτήν την έρευνα. Από όσο γνωρίζουν οι συγγραφείς, αυτή είναι η πρώτη φορά που ένα νευρωνικό δίκτυο βασισμένο στο μετασχηματισμό Fourier έχει προταθεί για το πρόβλημα NILM. Το θεμελιώδες στοιχείο του δικτύου είναι μια αρχιτεκτονική γνωστή ως μπλοκ Fourier, η οποία απεικονίζεται στο σχήμα B.3.4a. Η είσοδος του μπλοκ είναι ένας κανονικοποιημένος τανυστής. Μετά από αυτό, χρησιμοποιείται ο μετασχηματισμός Fourier. Τα πραγματικά και τα φανταστικά μέρη περνούν μέσα από ένα πυκνό στρώμα. Η συνάρτηση ενεργοποίησης μπορεί να είναι είτε γραμμική είτε μη γραμμική. Έχει παρατηρηθεί ότι η συνάρτηση relu βελτιώνει την απόδοση για ορισμένες συσκευές, όπως το πλυντήριο πιάτων. Στη συνέχεια, η αρχική είσοδος συνδυάζεται με την έξοδο του πυκνού στρώματος μέσω ενός υπολειπόμενου στρώματος. Υπάρχει ένα άλλο στρώμα κανονικοποίησης, ακολουθούμενο από ένα στρώμα γραμμικής πυκνότητας. Η είσοδος του πυκνού στρώματος προσαρτάται ως υπολειπόμενη σύνδεση στην έξοδο του, με αποτέλεσμα την τελική έξοδο του μπλοκ. Η αρχιτεκτονική από άκρο σε άκρο του NFED απεικονίζεται στο σχήμα B.3.4b. Ξεκινά με ένα συνελικτικό επίπεδο και στη συνέχεια περιλαμβάνει μια τεχνική συγκέντρωσης μέσης ισχύος 1D. Ακολουθεί ένα μπλοκ Fourier, η έξοδος του οποίου δρομολογείται μέσω δύο μη γραμμικών πυκνών στρωμάτων με συνάρτηση ενεργοποίησης relu. Τέλος, ένα γραμμικό επίπεδο παρέχει την έξοδο του δικτύου.



(a) Fourier block. (b) NFED architecture.

Figure B.3.4: Το προτεινόμενο νευρωνικό δίκτυο NFED μαζί με την αρχιτεκτονική του Fourier block.

B.3.4 Περίληψη Πειραματικών Αποτελεσμάτων

Τα αποτελέσματα που παράγονται από τα πειράματα της αρχιτεκτονικής online-GRU μαζί με τα αποτελέσματα του RNN από το Neural NILM (Kelly and Knottenbelt, 2015a) φαίνονται στο σχήμα B.3.6. Ένα παράδειγμα των εξόδων της προτεινόμενης αρχιτεκτονικής GRU και του short sequence2point απεικονίζεται στο Σχήμα B.3.5, όπου απεικονίζονται η πραγματική και η προβλεπόμενη κατανάλωση συσκευών από τα μοντέλων. Γενικά, δεν υπάρχει μοντέλο σαφώς ανώτερο από τα άλλα. Τα αποτελέσματα είναι μικτά, με ορισμένα μοντέλα να επιτυγχάνουν καλύτερα αποτελέσματα ανάλογα με τις συσκευές-στόχους και τη μέτρηση. Το πλεονέκτημα του προτεινόμενου μοντέλου είναι ότι σε γενικές γραμμές παρουσιάζει αποτελέσματα τεχνολογίας αιχμής με πολύ λιγότερες παραμέτρους.

Το Neural NILM RNN αποδίδει πολύ καλά στην ανίχνευση συμβάντων συσκευών δύο καταστάσεων όπως το ψυγείο και ο βραστήρας. Ωστόσο, αποτυγχάνει να προβλέψει την κατανάλωση ρεύματος, κάτι που γίνεται προφανές από το Σχετικό Σφάλμα στην Ολική Ενέργεια του βραστήρα. Τα δίκτυα που λαμβάνουν ως είσοδο το συρόμενο παράθυρο επιτυγχάνουν καλύτερα αποτελέσματα στις συσκευές πολλαπλών καταστάσεων όπως το πλυντήριο πιάτων και το πλυντήριο ρούχων. Αυτό υποδηλώνει τη χρήση πληροφοριών σχετικά με η κατανάλωση στα προηγούμενα ή/και επόμενα χρονικά σημεία, είναι χρήσιμη για να αναγνωρίσει το δίκτυο τη συμπεριφορά της συσκευής.

Τα πειραματικά αποτελέσματα του δικτύου GRU δείχνουν ότι έχει την ίδια ή καλύτερη απόδοση από την αρχιτεκτονική LSTM και στις πέντε συσκευές. Η απόδοση των δύο δικτύων είναι παρόμοια για το πλυντήριο. Για τις υπόλοιπες συσκευές το δίκτυο GRU είναι ανώτερο. Λαμβάνοντας υπόψη ότι οι νευρώνες LSTM είναι υπολογιστικά πιο ακριβοί και απαιτούν περισσότερη μνήμη, συνάγεται το συμπέρασμα ότι τα δίκτυα GRU είναι καλύτερα κατάλληλα για το έργο της διάσπασης ισχύος.

Το δίκτυο short sequence2point φαίνεται να αποδίδει σχεδόν το ίδιο με το δίκτυο GRU. Ο βραστήρας αποτελεί εξαίρεση, όπου το seq2point επιτυγχάνει σαφώς καλύτερες βαθμολογίες σε όλες τις μετρήσεις. Κρίνοντας από τα πειραματικά αποτελέσματα, το sequence2point και το online GRU είναι κατάλληλα για διαχωρισμό σε πραγματικό χρόνο, όπου ο χρήστης πρέπει να λάβει αποτελέσματα με ελάχιστο λανθάνοντα χρόνο.

Παρακάτω περιγράφονται τα πιο σημαντικά αποτελέσματα τα οποία αφορούν τα συγκριτικά τεστ για το μοντέλο NFED. Τα S2P και WGRU, δύο ισχυρά βασικά μοντέλα, χρησιμοποιούνται για την

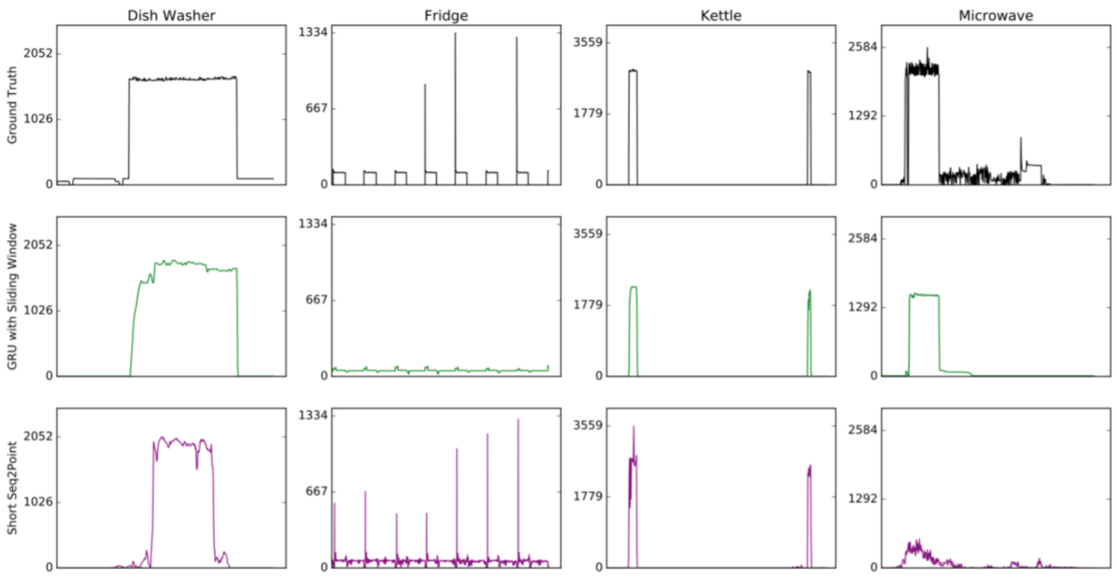


Figure B.3.5: Παραδείγματα εξόδων των δικτύων GRU και Short sequence2point .

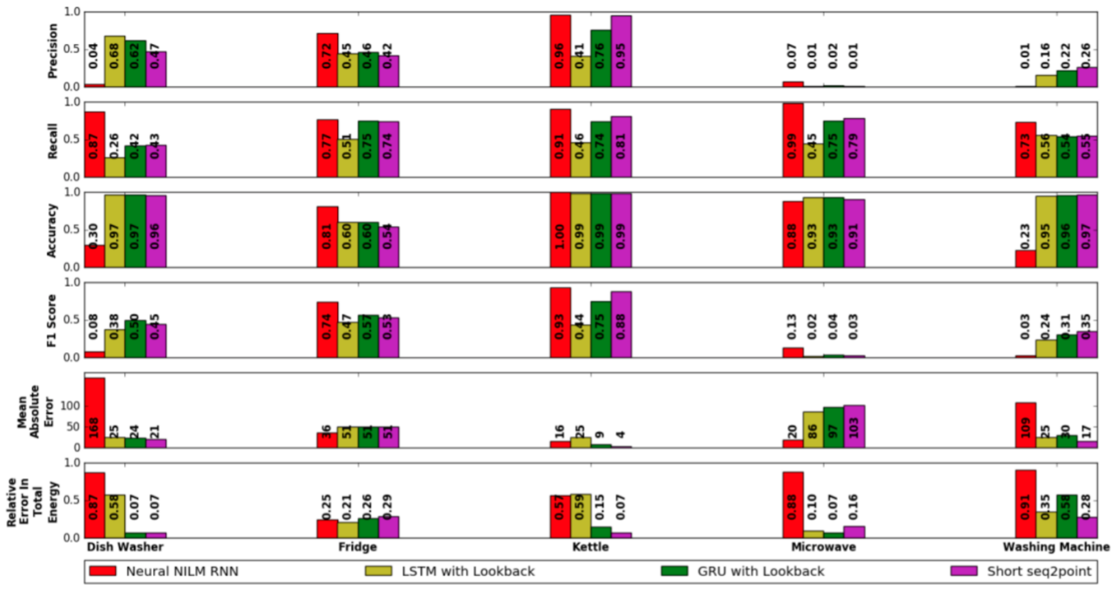
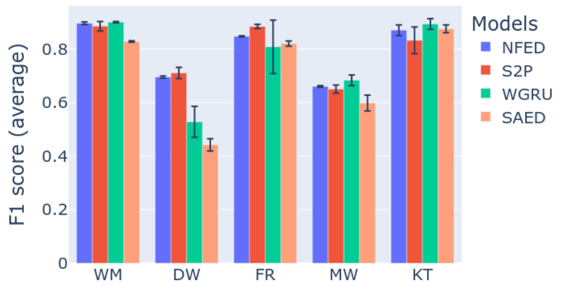
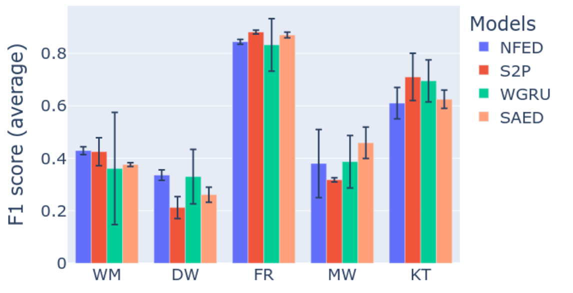


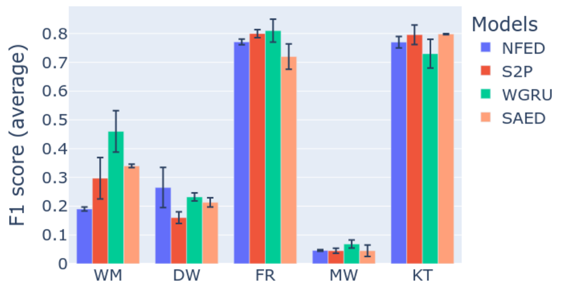
Figure B.3.6: Πειραματικά αποτελέσματα συμπεριλαμβανομένων των εκδόσεων παραθύρου ανασκόπησης του LSTM και του sequence2point, της προτεινόμενης ηλεκτρονικής αρχιτεκτονικής GRU και του αρχικού LSTM Kelly and Knottenbelt (2015a).



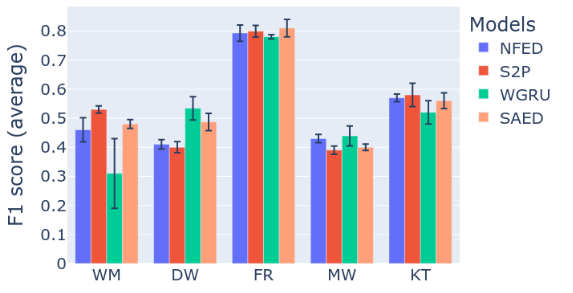
(a) Κατηγορία 1: Ένα σπίτι μάθησης NILM.



(b) Κατηγορία 2: Εκμάθηση και γενίκευση ενός κτιρίου στο ίδιο σύνολο δεδομένων.



(c) Κατηγορία 3: Εκμάθηση πολλαπλών κτιρίων και γενίκευση στο ίδιο σύνολο δεδομένων.

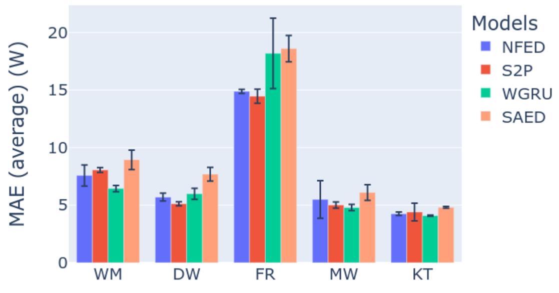


(d) Κατηγορία 4: Γενίκευση σε διαφορετικό σύνολο δεδομένων.

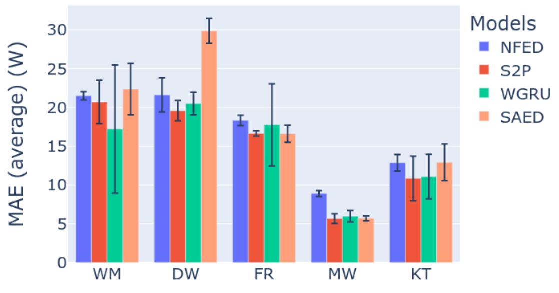
Figure B.3.7: Συγκριτικά αποτελέσματα για τα μοντέλα NFED, S2P, WGRU και SAED.

αξιολόγηση και τη σύγκριση του NFED. Και τα δύο αυτά μοντέλα έχουν υψηλό σκορ $F1$ και χαμηλό MAE. Τα μειονεκτήματα είναι ότι το S2P περιέχει μεγάλο αριθμό παραμέτρων, με αποτέλεσμα ένα αρκετά μεγάλο εκπαιδευμένο μοντέλο. Το WGRU έχει λίγες παραμέτρους, αλλά είναι αργό, καθώς αποτελείται κυρίως από επαναλαμβανόμενες μονάδες που κάνουν διαδοχικές λειτουργίες και όχι ταυτόχρονες. Ένα τρίτο μοντέλο βάσης, το SAED, είναι πιο αδύναμο διαχωρισμό, αλλά πολύ ελαφρύ και παρέχει καλή απόδοση γενίκευσης χάρη στον μηχανισμό προσοχής. Τα τέσσερα μοντέλα ελέγχονται και συγκρίνονται για τις ακόλουθες συσκευές χρησιμοποιώντας τη μεθοδολογία αναφοράς που αναφέρθηκε στις προηγούμενες ενότητες: πλυντήριο πιάτων, πλυντήριο ρούχων, ψυγείο, βραστήρας και φούρνος μικροκυμάτων. Οι μετρήσεις αξιολόγησης είναι η βαθμολογία $F1$ και η MAE. Το σχήμα B.3.7 δείχνει τα ευρήματα της βαθμολογίας $F1$. Τα αποτελέσματα για το MAE είναι συγκρίσιμα και φαίνονται στο σχήμα B.3.9.

Ξεκινώντας με την πρώτη ομάδα πειραμάτων, η οποία αξιολογεί τα μοντέλα σε αόρατα μελλοντικά δεδομένα από ένα μόνο κτίριο, το προτεινόμενο μοντέλο λαμβάνει την υψηλότερη ή τη δεύτερη υψηλότερη βαθμολογία $F1$ για όλες τις συσκευές. Από το σχήμα B.3.7a είναι σαφές ότι το NFED όχι μόνο αποδίδει σταθερά, αλλά έχει επίσης τη μικρότερη τυπική απόκλιση σε πολλές επαναλήψεις του ίδιου πειράματος. Όπως φαίνεται στο σχήμα B.3.7b, παρόμοια αποτελέσματα λαμβάνονται για τη δεύτερη κατηγορία πειραμάτων, όπου τα δεδομένα δοκιμής προέρχονται από διαφορετικό κτίριο. Η NFED είναι πολύ ανταγωνιστική σε αυτές τις δύο κατηγορίες πειραμάτων, όσον αφορά τις συσκευές

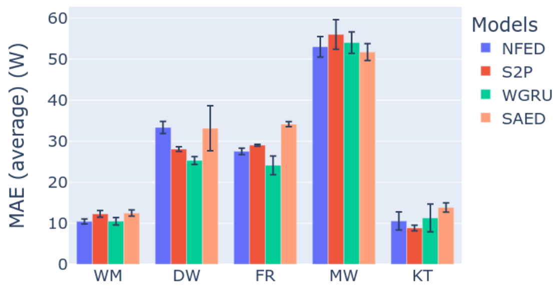


(a) Κατηγορία 1: Αποτελέσματα MAE.

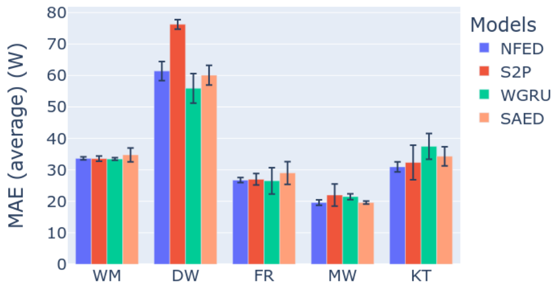


(b) Κατηγορία 2: Αποτελέσματα MAE.

Figure B.3.8: Αποτελέσματα Κατηγοριών συγκριτικής αξιολόγησης 1-2 για τα μοντέλα NFED, S2P, WGRU και SAED.



(a) Κατηγορία 3: Αποτελέσματα MAE.



(b) Κατηγορία 4: Αποτελέσματα MAE.

Figure B.3.9: Αποτελέσματα Κατηγοριών συγκριτικής αξιολόγησης 3-4 για τα μοντέλα NFED, S2P, WGRU και SAED.

πλυντηρίου πιάτων και ρούχων. Τα S2P και WGRU είναι τα επόμενα, με το S2P να έχει χαμηλότερες τυπικές αποκλίσεις αλλά τη χειρότερη απόδοση στην περίπτωση του πλυντηρίου πιάτων στην κατηγορία 2. Όσον αφορά το ψυγείο, και τα τέσσερα μοντέλα έχουν καλή απόδοση, με τα S2P και NFED να καταλαμβάνουν την πρώτη και τη δεύτερη θέση με μια μικρή διαφορά. Όσον αφορά τους φούρνους μικροκυμάτων, το WGRU και το NFED είναι τα καλύτερα μοντέλα στην κατηγορία 1, αλλά το SAED καταλαμβάνει την πρώτη θέση στην κατηγορία 2. Αυτό μπορεί να αποδοθεί στην ισχυρή ικανότητα γενίκευσης του SAED. Όσον αφορά τον βραστήρα της πρώτης κατηγορίας, όλα τα μοντέλα σκοράρουν πάνω από 80% $F1$. Στην κατηγορία 2, η απόδοση υποβαθμίζεται, με τις S2P και WGRU να έχουν τις καλύτερες επιδόσεις, ακολουθούμενες από τις NFED και SAED.

Οι δύο τελευταίες κατηγορίες του δείκτη αναφοράς είναι οι πιο δύσκολες. Στην κατηγορία 3, ένα μοντέλο επιχειρεί να μάθει από πολλά κτίρια, κάτι που είναι δύσκολο έργο γιατί μπορεί να υπάρχουν περισσότερα μοτίβα για μάθηση. Τα δύο κτίρια εκπαίδευσης μπορεί να έχουν διαφορετικό αριθμό συσκευών με διαφορετική συμπεριφορά κατανάλωσης ενέργειας. Στη συνέχεια πραγματοποιείται δοκιμή σε ένα άγνωστο σπίτι. Ως αποτέλεσμα, το μοντέλο είναι επιρρεπές στο να μάθει τις ομοιότητες μεταξύ των δύο σπιτιών εκπαίδευσης και η δοκιμή βασίζεται σε αυτές τις μαθημένες αναπαραστάσεις. Ομοίως, η κατηγορία 4 είναι πιο δύσκολη επειδή τα δεδομένα δοκιμών προέρχονται από διαφορετικό δίκτυο ηλεκτρικής ενέργειας. Παρά το γεγονός ότι η κατηγορία 4 είναι γενικά πιο δύσκολη, το τελικό αποτέλεσμα εξαρτάται σε μεγάλο βαθμό από την πραγματική πολυπλοκότητα του κτιρίου δοκιμών, όπως πόσες συσκευές διαθέτει. Συνολικά, και για τις δύο κατηγορίες εκπαίδευσης πολλαπλών κτιρίων, το προτεινόμενο μοντέλο έχει καλύτερη απόδοση ή είναι στο ίδιο επίπεδο με τα άλλα μοντέλα. Το SAED παρουσιάζει ισχυρή γενίκευση λόγω της χαμηλής μείωσης της απόδοσής του από θήκες μεμονωμένων κτιρίων. Ο πίνακας **??** περιέχει πρόσθετες πληροφορίες σχετικά με την απόδοση γενίκευσης των μοντέλων. Αν και οι δύο ισχυρές γραμμές βάσης, το S2P και το WGRU, είναι ανταγωνιστικές, κανένα δεν είναι σταθερά το καλύτερο μοντέλο. Λόγω της πολυπλοκότητας αυτών των δοκιμών, ο καθορισμός του καλύτερου μοντέλου για μια συγκεκριμένη συσκευή είναι δύσκολος. Από τα σχήματα B.3.7c και B.3.7d ένα εύλογο συμπέρασμα που μπορεί να εξαχθεί είναι ότι το NFED αποδίδει πάνω ή πάνω από τις γραμμές βάσης. Τα δύο μοντέλα με την καλύτερη συνολική απόδοση είναι το NFED και το S2P.

Λόγω της πολυπλοκότητας του προβλήματος, η σύγκριση μοντέλων NILM είναι δύσκολη. Στον πραγματικό κόσμο, τα δεδομένα είναι εκτός διανομής. Αυτό είναι ένα θεμελιώδες άλυτο ζήτημα στη μηχανική εκμάθηση. Το πλαίσιο αναφοράς που χρησιμοποιείται σε αυτή τη μελέτη προσομοιώνει το προαναφερθέν πρόβλημα και τα αποτελέσματα δείχνουν ότι κανένα μοντέλο δεν μπορεί να έχει παρόμοια απόδοση σε δεδομένα εκτός διανομής. Επιπλέον, στον πραγματικό κόσμο, είναι κρίσιμο να ληφθούν υπόψη οι ιδιότητες του μοντέλου, όπως το μέγεθός του και το πόσο γρήγορα μπορεί να τρέξει σε διάφορους υπολογιστικούς πόρους. Μια λογική τεχνική για τη σύγκριση διαφορετικών μοντέλων NILM είναι να ληφθούν υπόψη όλα τα πειραματικά ευρήματα του δείκτη αναφοράς και τα χαρακτηριστικά των μοντέλων.

Το σχήμα B.3.10 απεικονίζει ένα διάγραμμα αράχνης για την θήκη του πλυντηρίου πιάτων. Παρέχει τη βαθμολογία $F1$ για καθεμία από τις τέσσερις κατηγορίες του βενςημαρκ, καθώς και πληροφορίες σχετικά με το μέγεθος των μοντέλων, την ταχύτητα συμπερασμάτων σε μια GPU και την απόδοση συμπερασμάτων σε μια CPU. Όλες οι ποσότητες έχουν τις βέλτιστες τιμές τους στον εξωτερικό χώρο του δίσκου. Όσο πιο κοντά βρίσκεστε στο κέντρο, τόσο χειρότερο είναι το αποτέλεσμα. Όπως μπορεί να φανεί, το NFED αποδίδει θαυμάσια και στις τέσσερις κατηγορίες, καθιστώντας το εξαιρετικό διεκδικητή για ανάπτυξη σε πραγματικό κόσμο. Όταν ληφθούν υπόψη και οι άλλες ιδιότητες, το NFED είναι το τρίτο ταχύτερο μοντέλο. Το S2P θα ήταν κατάλληλος υποψήφιος εάν η ταχύτητα είναι κρίσιμη αλλά όχι σε βάρος της ακρίβειας. Από την άλλη πλευρά, το S2P είναι το μεγαλύτερο σε μέγεθος μοντέλο, γεγονός που θα το καθιστούσε απαγορευτικά ακριβό εάν ο χώρος αποθήκευσης είναι περιορισμένος. Κάθε μία από τις πέντε συσκευές που εξετάστηκαν σε αυτήν την έρευνα αναλύεται με τον ίδιο τρόπο και τα αντίστοιχα διαγράμματα αράχνης μπορούν να βρεθούν στην αναλυτική περιγραφή της διδακτορικής διατριβής. Συνοψίζοντας, κάθε μοντέλο έχει μια σειρά από πλεονεκτήματα και μειονεκτήματα και μπορεί να ταιριάζει ή να μην ταιριάζει ανάλογα με τη σημασία των παραγόντων και τις συνολικές ανάγκες του συστήματος.

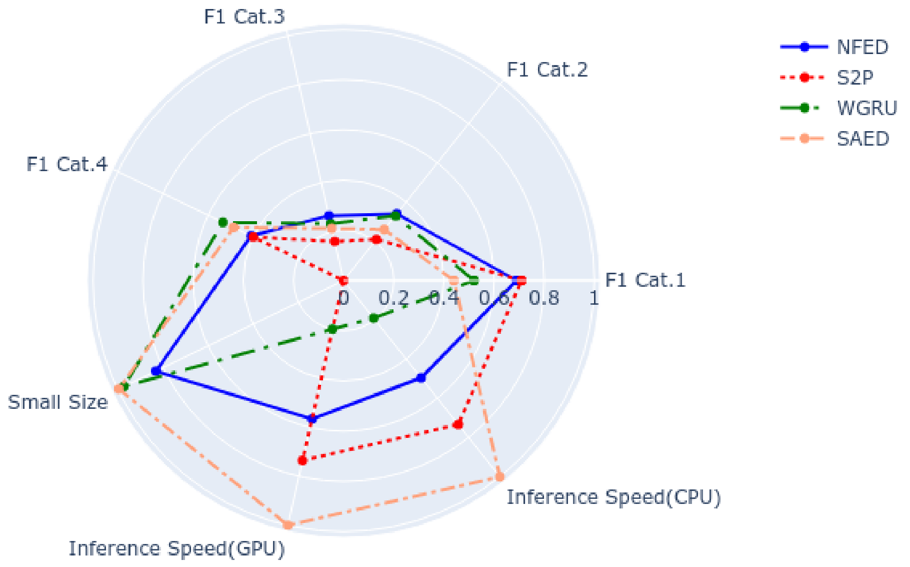


Figure B.3.10: Διάγραμμα που συνοψίζει τις δυνατότητες των μοντέλων να ξεχωρίζουν ένα πλυντήριο πιάτων.

Μπορεί να είναι δύσκολο να σχεδιαστεί ένα μη παρεμβατικό σύστημα παρακολούθησης φορτίου. Οι απαιτήσεις συστήματος ενδέχεται να διαφέρουν ανάλογα με το περιβάλλον στόχο στο οποίο θα εφαρμοστεί το μοντέλο. Λόγω της αφθονίας των πόρων, η εκτέλεση μοντέλων κατανομής ενέργειας στο cloud μπορεί να είναι πιο ευέλικτη. Μια λύση cloud, από την άλλη πλευρά, μπορεί να είναι πολύ ακριβή καθώς το σύστημα επεκτείνεται. Η εναλλακτική είναι η εκτέλεση τέτοιων μοντέλων σε μια ενσωματωμένη συσκευή, η οποία έχει περιορισμένους πόρους. Αυτή η εργασία εισάγει το NFED, ένα μοναδικό νευρωνικό δίκτυο που είναι κατάλληλο και για τις δύο λύσεις. Το NFED χρησιμοποιεί λίγο χώρο, έχει γρήγορο χρόνο συμπερασμάτων και παρέχει συγκρίσιμα ή υψηλότερα αποτελέσματα απόδοσης. Η χρήση του μετασχηματισμού Fourier, ο οποίος μπορεί να υπολογιστεί γρήγορα και δεν έχει παραμέτρους εκμάθησης, είναι το κλειδί για την αποτελεσματικότητα του NFED.

B.4 Αναπαραστάσεις Χρονοσειρών σε Χώρους Εμβύθισης

Ο αλγόριθμος Signal2vec Nalmpantis and Vrakas (2019b) είναι ένας νέος αλγόριθμος, που αντιστοιχίζει οποιαδήποτε χρονική σειρά σε ένα διανυσματικό χώρο. Είναι μια γενίκευση του word2vec

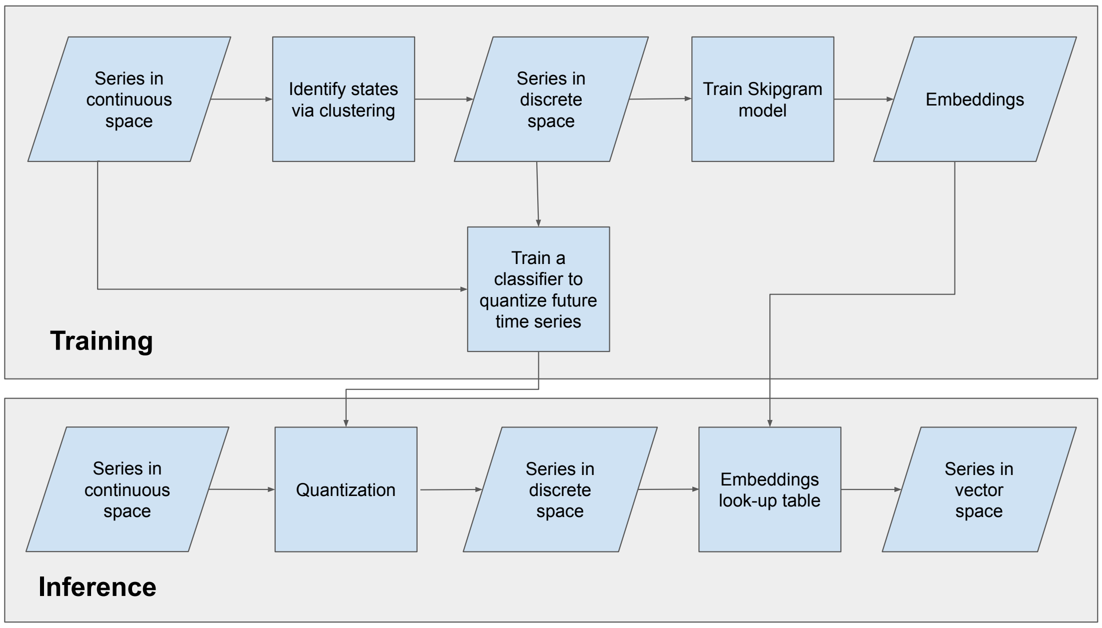


Figure B.4.1: Απεικόνιση του αλγορίθμου Signal2Vec.

Mikolov et al. (2013) και επεκτείνει την εφαρμογή του σε ακολουθίες σε συνεχή χώρο. Οι δύο βασικές έννοιες του Signal2vec περιλαμβάνουν μια διαδικασία κβαντοποίησης και το μοντέλο skip-gram Mikolov et al. (2013). Το τελευταίο είναι υπεύθυνο για την κατασκευή του χώρου εμθύθισης. Η διαδικασία εκμάθησης του χώρου εμβύθισης είναι χωρίς επίβλεψη, υπολογιστικά αποδοτική και επεκτάσιμη. Το κύριο πλεονέκτημα σε σύγκριση με άλλους αλγόριθμους αναπαράστασης χρονοσειρών είναι ότι οι ενσωματώσεις δημιουργούνται μόνο μία φορά και μπορούν να επαναχρησιμοποιηθούν από άλλα συστήματα μηχανικής εκμάθησης που εφαρμόζονται σε άγνωστα σύνολα δεδομένων.

Μια απεικόνιση ολόκληρης της ροής εργασίας του Signal2Vec παρουσιάζεται στο Σχήμα B.4.1. Η δεδομένη χρονική σειρά κβαντίζεται μέσω ομαδοποίησης και κάθε σύμπλεγμα ορίζει μία διακριτή τιμή. Ο αριθμός των συμπλεγμάτων υπολογίζεται χρησιμοποιώντας τη βαθμολογία σιλουέτας, η οποία λέει ποια αντικείμενα ταιριάζουν καλά στο σύμπλεγμα Rousseeuw (1987). Ο χώρος αναζήτησης του αριθμού των συστάδων περιορίζεται επίσης από έναν ελάχιστο και έναν μέγιστο επιθυμητό αριθμό συστάδων. Στον τομέα της ενέργειας, τα υποθετικά όρια του χώρου μπορούν να εκτιμηθούν με τον υπολογισμό του αριθμού των πιθανών ενεργειακών καταστάσεων χρησιμοποιώντας την πολυπλοκότητα των αναλήψεων ισχύος Egarter et al. (2015c). Μόλις βρεθούν τα συμπλέγματα, μια συνάρτηση εκπαιδεύεται να αντιστοιχίζει οποιαδήποτε χρονική σειρά σε μια διακριτή ακολουθία διακριτικών. Αυτή η συνάρτηση μπορεί να είναι ένας ταξινομητής, όπως οι κ-πλησιέστεροι γείτονες, ο οποίος εκπαιδεύεται μόνο μία φορά και στη συνέχεια μπορεί να χρησιμοποιηθεί για να αντιστοιχίσει οποιαδήποτε νέα χρονική σειρά στον πεπερασμένο διακριτοποιημένο χώρο.

Στην ορολογία του word2vec, το token είναι μια λέξη, ο πεπερασμένος χώρος των tokens ονομάζεται λεξιλόγιο και μια ακολουθία tokens ονομάζεται corpus. Για να εφαρμόσουμε το μοντέλο skip-gram πρέπει να ορίσουμε ένα context.

Definition B.4.1. *Context:* Δεδομένης μιας ακολουθίας διακριτικών και ενός συρόμενου παραθύρου μήκους W, με περιττό W, ας είναι το TKN_{target} το μεσαίο διακριτικό του παραθύρου. Στη συνέχεια, το πλαίσιο αποτελείται από τα υπόλοιπα tokens μέσα στο παράθυρο.

Ο στόχος του μοντέλου skip-gram είναι να προβλέψει το πλαίσιο στο οποίο δίνεται ένα συγ-

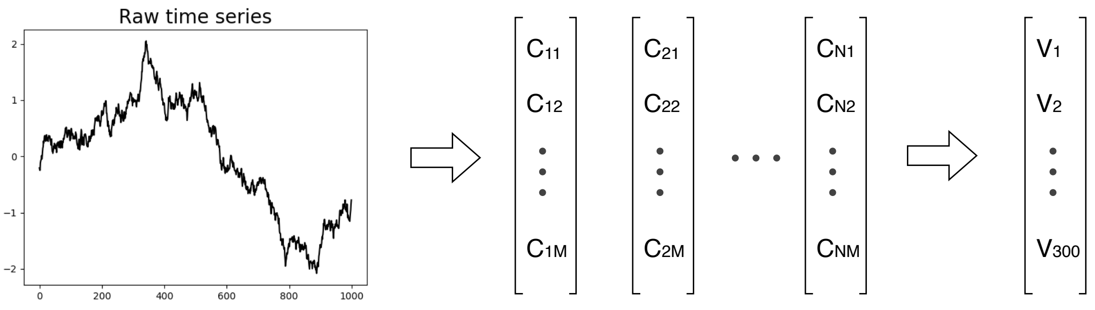


Figure B.4.2: Αντιστοίχιση χρονολογικής σειράς σε ένα διάνυσμα.

κεκριμένο token. Η αρχιτεκτονική είναι ένα ρηχό νευρωνικό δίκτυο με ένα κρυφό στρώμα και εκπαιδεύεται με ζεύγη tokens. Το ένα token είναι ο στόχος και το άλλο ανήκει στο πλαίσιο. Το δίκτυο εκπαιδεύει τα βάρη του κρυφού στρώματος από τις συχνότητες που εμφανίζει κάθε σύζευξη. Ο επίσημος ορισμός της αντικειμενικής συνάρτησης δίνεται παρακάτω.

Έστω $TKN_1, TKN_2, ...TKN_n$ μια ακολουθία N κουπονιών εκπαίδευσης. Στη συνέχεια, η αντικειμενική συνάρτηση προσπαθεί να μεγιστοποιήσει τη μέση πιθανότητα καταγραφής σύμφωνα με τον τύπο:

$$1/N \sum_{t=1}^{N} \sum_{-c \leq i \leq c, i \neq 0} \log p(TKN_{t+i}|TKN_t) \tag{B.10}$$

, όπου c είναι το context εκπαίδευσης.

Η συνάρτηση απώλειας του νευρωνικού δικτύου είναι Noise Contrastive Estimation (NCE) Gutmann and Hyvärinen (2012). Ο αλγόριθμος Adagrad Duchi et al. (2011) είναι το επιλεγμένο εργαλείο βελτιστοποίησης που βασίζεται σε κλίση, με ρυθμό εκμάθησης 0,001. Το μέγεθος κάθε ενσωμάτωσης είναι 300 και το πλαίσιο είναι 6 tokens.

Definition B.4.2. *Signal2Vec:* Έστω μια χρονική σειρά S μήκους N, μια συνάρτηση αντιστοίχισης $f : R \rightarrow T^n$ και μια συνάρτηση αναζήτησης $g : T^n \rightarrow V^{n*M}$, με T ένα πεπερασμένο σύνολο n tokens και V ένας διανυσματικός χώρος M διαστάσεων με n διανύσματα. Ορίζουμε το Signal2Vec ως την αντιστοίχιση μιας χρονοσειράς σε ένα διανυσματικό χώρο με τον ακόλουθο τύπο:

$$Signal2Vec = g \circ f : R \rightarrow V^{n*M} \tag{B.11}$$

Σύμφωνα με τον ορισμό του Signal2Vec, ο κατασκευασμένος διανυσματικός χώρος έχει τόσα διανύσματα όσα και το μέγεθος του συνόλου των tokens, με κάθε διάνυσμα να έχει 300 διαστάσεις. Έτσι, μια χρονοσειρά αντιστοιχίζεται σε μια ακολουθία tokens, η οποία χαρτογραφείται σε μια ακολουθία διανυσμάτων 300 διαστάσεων. Μια τέτοια αναπαράσταση δεν είναι πολύ χρήσιμη καθώς επιδεινώνει το πρόβλημα της έκρηξης διαστάσεων. Η έννοια της αναπαράστασης του μέσου διανύσματος λύνει αυτό το πρόβλημα.

Definition B.4.3. Μέση διανυσματική αναπαράσταση *:* Αφήστε μια χρονοσειρά S μήκους N να αντιστοιχιστεί σε μια ακολουθία N διανυσμάτων με M διαστάσεων το καθένα. Έστω αυτά τα διανύσματα $\hat{C}_i, i = 1, ..., N$. Η μέση διανυσματική αναπαράσταση $S2V_{avg}$ της χρονοσειράς S ορίζεται από:

$$S2V_{avg} = 1/N \sum_{i=1}^{N} \hat{C}_i \tag{B.12}$$

Algorithm 4 SIGNAL2VEC TRAINING Εκπαιδεύει έναν ταξινομητή για κβαντοποίηση και δημιουργεί έναν διανυσματικό χώρο της δεδομένης χρονοσειράς.

```
Input: A time-series $ts\_data$ and the desired minimum and maximum states $min\_states$ and $max\_states$
       respectively.
Output: A trained classifier which can be used to quantize relevant time-series and a dictionary to map the
        quantized values to vectors.
Data: Signal2vec training data.
$labels \leftarrow dict()$
$scores \leftarrow dict()$
$selected\_states \leftarrow 0$
$max\_score \leftarrow 0$
for $states \leftarrow min\_states$ to $max\_states$ do
    $clustering\_model \leftarrow KMeans(n\_clusters \leftarrow states)$
    $clustering\_model.fit(ts\_data)$
    $labels[states] \leftarrow clustering\_model.labels$
    $scores[states] \leftarrow silhouette\_score(ts\_data, labels)$
    if $scores[states] > max\_score$ then
        $max\_score \leftarrow scores[states]$
        $selected\_states \leftarrow states$
    end
end
$tokens \leftarrow labels[selected\_states]$
$classifier \leftarrow KNeighborsClassifier(n\_neighbors = selected\_states)$
$classifier.fit(ts\_data, tokens)$
$vocabulary \leftarrow set(tokens)$
$tokens\_sequence \leftarrow classifier.predict(ts\_data)$
$embeddings\_dict \leftarrow SkipGram.train(tokens\_sequence, vocabulary)$
return $classifier, embeddings\_dict$
```

Algorithm 5 SIGNAL2VEC INFERENCE Μετατρέπει μια δεδομένη χρονική σειρά σε διάνυσμα.

```
Input: A time-series $ts\_data$, a $classifier$ to quantize the given time series and a dictionary
       $embeddings\_dict$ to map the discrete sequence to a sequence of vectors.
Output: A vector representation $S2V_{avg}$ of the given time series.
Data: A time-series or a segment of any size.
$tokens\_sequence \leftarrow classifier.predict(ts\_data)$
$embeddings\_sequence \leftarrow list()$
foreach $token$ in $tokens\_sequence$ do
    $embeddings\_sequence.append(embeddings\_dict[token])$
end
$S2V_{avg} \leftarrow average(embeddings\_sequence)$
return $S2V_{avg}$
```

Συνεπώς, μια χρονική σειρά μήκους N μπορεί να αναπαρασταθεί από ένα μόνο διάνυσμα M, διαστάσεων όπως στο σχήμα B.4.2. Το πλεονέκτημα αυτής της αντιστοίχισης είναι ότι η τελική αναπαράσταση είναι ανεξάρτητη από το αρχικό μήκος N. Σε περίπτωση που μια χρονοσειρά είναι

Table B.4.1: Macro f1 score for 12 appliances.

TS size	**Signal2vec**	**Raw**	**SAX**	**PAA**
1 day	0.730	0.730	0.682	**0.814**
3 days	**0.878**	0.732	0.748	0.869
5 days	**0.930**	0.681	0.767	0.897
7 days	**0.942**	0.693	0.766	0.920

πολύ μεγάλη, μπορεί να χωριστεί σε ίσα τμήματα και η τελική διανυσματική αναπαράσταση θα είναι η συνένωση των μέσων διανυσμάτων κάθε τμήματος.

Οι πλήρεις αλγόριθμοι εκπαίδευσης και συμπερασμάτων του Signal2Vec περιγράφονται από τους Αλγόριθμος 4 και Αλγόριθμος 5 αντίστοιχα. Η εκπαίδευση του Signal2Vec είναι γρήγορη επειδή το νευρωνικό δίκτυο είναι πολύ ρηχό και δεν επιβραδύνει την εξαγωγή συμπερασμάτων επειδή συμβαίνει μόνο μία φορά. Επομένως, ο χρόνος συμπερασμάτων εξαρτάται κυρίως από τον χρόνο συμπερασμάτων του ταξινομητή καθώς η αντιστοίχιση των tokens σε ενσωματώσεις είναι $O(1)$. Έτσι, η χρονική πολυπλοκότητα συμπερασμάτων του k-NN είναι $O(N*S)$, όπου N είναι το μήκος της χρονοσειράς υπό μετατροπή και S είναι ο αριθμός των δειγμάτων που εκπαιδεύτηκε ο ταξινομητής. Λαμβάνοντας υπόψη ότι το S δεν αλλάζει, συμπεραίνουμε ότι η χρονική πολυπλοκότητα της διαδικασίας συμπερασμάτων Signal2Vec είναι γραμμική. Η επιλογή ενός ταχύτερου ταξινομητή αποτελεί αντικείμενο μελλοντικής έρευνας και συνιστάται για μελλοντικές εφαρμογές.

Το Signal2Vec εφαρμόζεται σε δύο εργασίες μηχανικής μάθησης: μια ταξινόμηση πολλαπλών κλάσεων και μια ταξινόμηση πολλαπλών ετικετών. Αυτή η πρόταση παρουσιάζει δύο πειράματα πραγματικού κόσμου: ταξινόμηση συσκευών και ενεργειακή κατανομή. Το πρώτο διερευνά την ικανότητα του προτεινόμενου πλαισίου να ταξινομεί αποτελεσματικά τα σήματα από διάφορες συσκευές. Η δεύτερη είναι μια απλή μέθοδος για την επίλυση ενός προβλήματος τυφλού διαχωρισμού πηγής, όπως η διάσπαση ενέργειας, με το Signal2vec.

Ο πίνακας B.4.1 συγκρίνει το Signal2vec με ακατέργαστα δεδομένα και δύο άλλες αναπαραστάσεις, το SAX Lin et al. (2007) και το PAA Keogh and Pazzani (2000). Τα αποτελέσματα περιλαμβάνουν χρονοσειρές με μεγέθη 1, 5 και 7 ημερών, χρησιμοποιώντας 3 διασταυρούμενη επικύρωση. Άλλοι ταξινομητές έχουν επίσης δοκιμαστεί. Ανεξάρτητα από την αναπαράσταση, το τυχαίο δάσος έδειξε τα καλύτερα αποτελέσματα. Το SAX και το PAA έχουν συντονιστεί για να λάβουν την καλύτερη δυνατή βαθμολογία f1. Συνολικά το Signal2vec είναι ανώτερο, όχι μόνο δίνει τα καλύτερα αποτελέσματα, αλλά και η διάσταση του τελικού αντιπροσωπευτικού διανύσματος είναι ανεξάρτητη από το μήκος της χρονοσειράς. Το Signal2vec ξεπερνιέται από το PAA, μόνο για μικρές χρονικές σειρές όπως το μέγεθος 1 ημέρας. Μια πιθανή εξήγηση είναι ότι τα χρονικά μοτίβα είναι πιο σημαντικά σε μικρές περιόδους, επειδή μια συσκευή χρησιμοποιείται σπάνια. Κατά τον υπολογισμό του μέσου διανύσματος χάνονται τυχόν χρονικά μοτίβα, ενώ στο PAA διατηρούνται. Για μελλοντικά πειράματα μπορούν να αξιολογηθούν πιο εξελιγμένες μέθοδοι π.χ. σταθμισμένο μέσο διάνυσμα. Τέλος, είναι αξιοσημείωτο ότι για μεγαλύτερες χρονοσειρές τα αποτελέσματα για τα ανεπεξέργαστα δεδομένα χειροτερεύουν. Αυτό εξηγείται από την έκρηξη διαστάσεων, την οποία αντιμετωπίζει το signal2vec συμπιέζοντας όλες τις πληροφορίες σε ένα ενιαίο διάνυσμα σταθερών διαστάσεων.

Ο στόχος των επόμενων πειραμάτων είναι να δείξουν ότι η μείωση διαστάσεων των ενεργειακών δεδομένων οδηγεί σε υπολογιστικά ελαφριές λύσεις για την εργασία του NILM πολλαπλών ετικετών. Οι εξεταζόμενες αναπαραστάσεις χρονοσειρών είναι: PAA, SAX, 1d-SAX, DFT, SFA, BOSS, WEASEL και Signal2Vec. Τα μοντέλα μηχανικής εκμάθησης που επιλέγονται για την αξιολόγηση περιορίζονται σε ελαφριά μοντέλα που υποστηρίζουν ταξινόμηση πολλαπλών ετικετών. Αυτά τα μοντέλα περιλαμ-

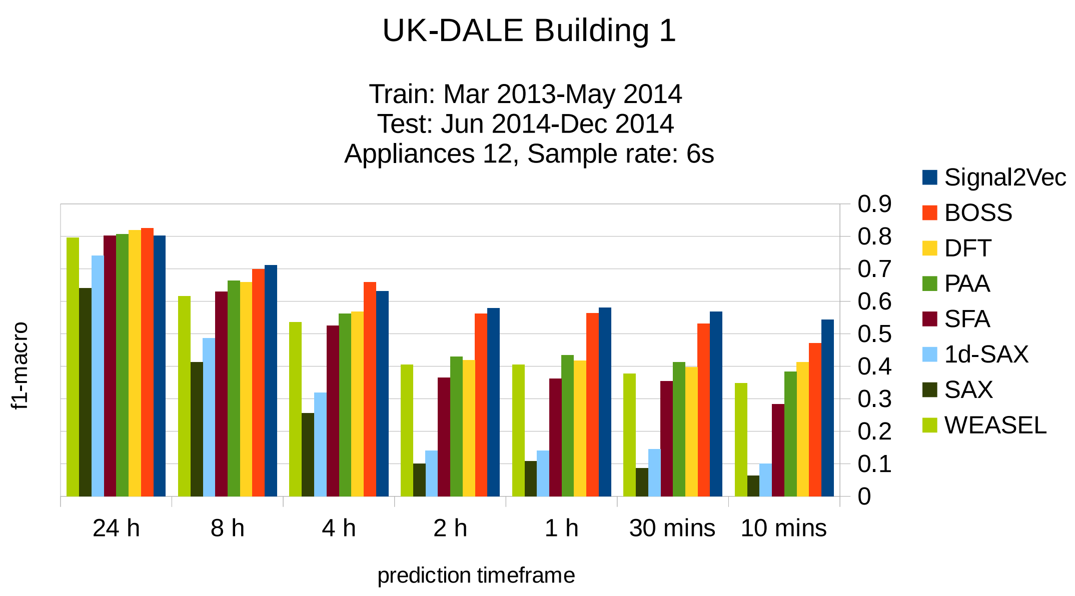


Figure B.4.3: Οικιακές ηλεκτρικές συσκευές: φούρνος, φούρνος μικροκυμάτων, πλυντήριο πιάτων, ψυγείο, βραστήρας, πλυντήριο ρούχων, τοστιέρα, βραστήρας, τηλεόραση, πιστολάκι μαλλιών, ηλεκτρική σκούπα, φως.

βάνουν δέντρα απόφασης, επιπλέον δέντρα, τυχαία δάση και νευρωνικά δίκτυα τροφοδοσίας. Όσον αφορά την αξιολόγηση, οι μετρήσεις που χρησιμοποιούνται σε εργασίες ταξινόμησης πολλαπλών ετικετών είναι οι μικρο και οι μακροοικονομικές βαθμολογίες f1.

Μετά την επιλογή του μοντέλου, εκπαιδεύονται και ελέγχονται τα καλύτερα μοντέλα μηχανικής εκμάθησης που έχουν συνδυαστεί με τις αντίστοιχες παραμετροποιημένες χρονοσειρές προσέγγισης. Η εκπαίδευση πραγματοποιείται στο σπίτι 1 του συνόλου δεδομένων UK-DALE από τον Μάρτιο του 2013 έως τον Μάιο του 2014. Οι δοκιμές πραγματοποιούνται στο ίδιο σπίτι για τις 12 συσκευές κατά την περίοδο Ιουνίου 2014 έως Δεκεμβρίου 2014. Αυτή είναι μια πολύ μεγάλη περίοδος δοκιμών, σε αντίθεση με την πλειονότητα των πειραμάτων της βιβλιογραφίας, όπου τα συστήματα NILM δοκιμάζονται για περίοδο μιας εβδομάδας ή μερικών μηνών. Τα αποτελέσματα παρουσιάζονται στο σχήμα B.4.3. Οι δύο πιο αποτελεσματικές αναπαραστάσεις χρονοσειρών είναι οι BOSS και Signal2Vec, με τη δεύτερη να δείχνει συνολικά την καλύτερη απόδοση. Τα ακόλουθα δύο μοντέλα είναι DFT και PAA, που παρουσιάζουν αρκετά ισχυρά αποτελέσματα, ειδικά όταν το χρονικό παράθυρο είναι μεγάλο.

Αξίζει να σημειωθεί ότι το Signal2Vec είναι η μόνη αναπαράσταση που υποστηρίζει τη μεταφορά εκμάθησης. Εκπαιδεύτηκε με χρονικό πλαίσιο ίσο με 6 δευτερόλεπτα και τα ίδια διανύσματα χρησιμοποιούνται για παράθυρα με διάρκεια από 10 λεπτά έως μία ημέρα. Όλες οι άλλες αναπαραστάσεις προσαρμόζονται και παραμετροποιούνται για κάθε διαφορετικό χρονικό πλαίσιο. Η μόνη παράμετρος του Signal2Vec είναι ο αριθμός των μέσων διανυσμάτων ανά χρονικό πλαίσιο που είναι 1 για μικρά παράθυρα και 5 για εκείνα που υπερβαίνουν τις 2 ώρες. Μια άλλη αξιοσημείωτη παρατήρηση είναι ότι το Signal2Vec όχι μόνο ξεπερνά τον αλγόριθμο BOSS, αλλά χρησιμοποιεί επίσης ένα μικρότερο νευρωνικό δίκτυο με σχεδόν τους μισούς νευρώνες.

Μια άλλη σημαντική παράμετρος στην ενεργειακή διάσπαση είναι ο αριθμός των συσκευών που αναγνωρίζονται. Αυτό το σετ πειραμάτων περιλαμβάνει 4 έως 12 διαφορετικές συσκευές. Το σχήμα 5.3.3 παρουσιάζει ορισμένα αντιπροσωπευτικά αποτελέσματα με χρονική περίοδο μία ώρα. Οι 12

Table B.4.2: Συγκρίνοντας το multi-NILM με τα προτεινόμενα μοντέλα των Tabatabaei et al.

Dataset	House	RAkEL Time	MLkNN Time	RAkEL Wavelet	MLkNN Wavelet	NN Signal2Vec	NN BOSS
REDD	1	0.476	0.490	0.450	0.504	**0.518**	0.369
REDD	3	0.372	0.414	0.312	0.434	**0.446**	0.170
UK-DALE	1	0.288	0.423	0.337	0.430	**0.492**	0.439

Table B.4.3: Σύγκριση ανά συσκευή στο House 1 REDD

Appliance	RAkEL Time	MLkNN Time	RAkEL Wavelet	MLkNN Wavelet	NN Signal2Vec	NN BOSS
oven	0.0	0.03	**0.12**	0.08	0.11	0.0
refrigerator	0.27	**0.43**	0.19	**0.43**	**0.43**	0.41
microwave	0.22	0.33	0.27	0.33	**0.45**	0.21
washer dryer	0.09	0.29	0.11	0.28	**0.34**	0.04
bath GFI	0.27	0.5	0.31	0.48	**0.55**	0.24
sockets	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
light	0.39	0.43	0.4	0.41	**0.46**	0.22

συσκευές είναι αυτές που αναφέρθηκαν προηγουμένως. Το σετ των 9 συσκευών περιέχει: φούρνο μικροκυμάτων, πλυντήριο πιάτων, ψυγείο, βραστήρα, πλυντήριο στεγνωτήριο, τοστιέρα, τηλεόραση, πιστολάκι μαλλιών και ηλεκτρική σκούπα. Το σετ των 6 είναι: φούρνος, πλυντήριο πιάτων, ψυγείο, φούρνος μικροκυμάτων, βραστήρας και τοστιέρα. Τέλος, το σετ των 4 συσκευών περιλαμβάνει αυτές που χρησιμοποιούνται κυρίως σε ερευνητικές εργασίες NILM: πλυντήριο πιάτων, ψυγείο, φούρνο μικροκυμάτων και βραστήρα. Το Signal2vec επιτυγχάνει και πάλι την καλύτερη βαθμολογία μακροεντολής f1, με το BOSS να ακολουθεί. Το PAA, το DFT και το WEASEL είναι πολύ κοντά, το SFA ακολουθεί με μέση απόδοση, ενώ το SAX και το 1d-SAX δίνουν τα χειρότερα αποτελέσματα.

Σύμφωνα με τη βιβλιογραφία, τα μοντέλα που προτείνουν οι Tabatabaei et al. χρησιμοποιούνται συχνά ως ισχυρή βάση για την αξιολόγηση λύσεων αιχμής. Δυστυχώς, αυτά τα πειράματα δεν λαμβάνουν πάντα υπόψη όλες τις παραμέτρους μιας εγκατάστασης συστήματος NILM και μερικές φορές η σύγκριση με τα αποτελέσματα μιας άλλης ερευνητικής εργασίας δεν είναι δίκαιη. Προκειμένου να επιτευχθεί μια αντικειμενική αξιολόγηση τα πειράματα που περιγράφονται από τους Tabatabaei et al. αναπαράγονται όσο το δυνατόν πιο κοντά αναπτύσσοντας τα μοντέλα από την αρχή. Διεξάγονται επίσης πρόσθετα πειράματα για την ενίσχυση των αποτελεσμάτων.

Η λύση των Tabatabaei et al. περιλαμβάνει δύο πολύ δημοφιλείς μεθόδους πολλαπλών ετικετών, που ονομάζονται MLkNN και RAkEL. Και τα δύο αξιολογούνται εξάγοντας χαρακτηριστικά είτε στον τομέα χρόνου είτε στον τομέα συχνότητας. Στον τομέα χρόνου, η μέθοδος που χρησιμοποιείται είναι ενσωματώσεις χρονικής καθυστέρησης Kantz and Schreiber (2004). Έχει δύο παραμέτρους τη χρονική καθυστέρηση και τις διαστάσεις των ενσωματώσεων. Η πρώτη παράμετρος προσδιορίζεται χρησιμοποιώντας τη μέθοδο αμοιβαίας ενημέρωσης με χρονική καθυστέρηση. Το δεύτερο διαμορφώνεται μέσω της μεθόδου των ψευδών πλησιέστερων γειτόνων. Στον τομέα συχνότητας το σήμα μετασχηματίζεται χρησιμοποιώντας το κυματίδιο Haar.

Τα αποτελέσματα των πειραμάτων συνοψίζονται στον Πίνακα B.4.2 χρησιμοποιώντας τη μετρική macro f1-score ως μέτρηση. Το καλύτερο μοντέλο πολλαπλών ετικετών NILM, βασισμένο στο Signal2Vec και ένα ρηχό νευρωνικό δίκτυο, ξεπέρασε όλα τα άλλα μοντέλα και στα τρία σπίτια. Είναι πολύ σημαντικό να τονίσουμε ότι ο διανυσματικός χώρος που χρησιμοποιείται από το Signal2Vec, για τα πειράματα στο σύνολο δεδομένων REDD, είναι ο ίδιος που εκπαιδεύεται στο σύνολο δεδομένων UK-DALE. Αυτό είναι απόδειξη των δυνατοτήτων γενίκευσης των κατασκευασμένων διανυσμάτων. Οι ιδιότητες που έχουν μάθει από την κατανάλωση ενέργειας σε ένα σπίτι στο Ηνωμένο Βασίλειο, ισχύουν και για σπίτια από τις ΗΠΑ. Το δεύτερο καλύτερο μοντέλο είναι το MLkNN που χρησιμοποιεί κυματίδια, αλλά μόνο για σπίτια από το REDD. Στο UK-DALE το δεύτερο καλύτερο μοντέλο είναι αυτό που βασίζεται στο BOSS. Από την άλλη το BOSS δεν δείχνει πολύ συνεπή συμπεριφορά καθώς δείχνει τα χειρότερα αποτελέσματα για το σπίτι 3 από το REDD. Γενικά, το RAkEL δείχνει χαμηλότερη βαθμολογία f1 σε σύγκριση με το MLkNN. Ένα πιο λεπτομερές δείγμα των πειραματικών αποτελεσμάτων παρουσιάζεται στον Πίνακα B.4.3. Τα αποτελέσματα προέρχονται από το σπίτι 1 του REDD και αντιπροσωπεύουν τη βαθμολογία f1 κάθε συσκευής. Το Signal2Vec επιτυγχάνει τις καλύτερες βαθμολογίες για όλες τις συσκευές, εκτός από τον φούρνο. Το RAkEL στον τομέα συχνότητας αποδίδει την καλύτερη βαθμολογία για τον φούρνο.

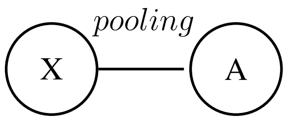


Figure B.5.1: Markov αλυσίδα λειτουργίας pooling .

B.5 Αρχές της Θεωρίας Πληροφορίας στη Βαθιά Μάθηση

Η έννοια της συγκέντρωσης χαρακτηριστικών (feature pooling) χρονολογείται από ένα σημαντικό άρθρο σχετικά με σύνθετα κύτταρα στον οπτικό φλοιό του Hubel and Wiesel (1962). Χρησιμοποιείται σε πολλές χειροποίητες μεθόδους μηχανικής χαρακτηριστικών όπως η μέθοδος SIFT Lowe (2004) και η μεθοδος HOG Dalal and Triggs (2005). Ειδικά το max Jarrett et al. (2009) και average pooling LeCun et al. (1990, 1998) χρησιμοποιούνται συχνά σε συνελικτικά νευρωνικά δίκτυα. Η στοχαστική συγκέντρωση έδειξε επίσης υπερσύγχρονα αποτελέσματα βαθιών συνελικτικών νευρωνικών δικτύων Zeiler and Fergus (2013). Οι Graham (2014) παρουσιάζουν μια εναλλακτική εκδοχή της στοχαστικής συγκέντρωσης που ονομάζεται κλασματική max-pooling. Η κύρια ιδέα πίσω από αυτήν την τεχνική είναι ότι οι περιοχές συγκέντρωσης δεν έχουν προκαθορισμένο μέγεθος, αλλά μπορούν να επιλεγούν με στοχαστικό τρόπο.

Η μέθοδος συγκέντρωση είναι αναπόσπαστο μέρος αρκετών αρχιτεκτονικών νευρωνικών δικτύων και υπάρχει πληθώρα παραδειγμάτων που υποδηλώνουν τα οφέλη της. Ωστόσο, οι περισσότερες μελέτες είναι εμπειρικές και λείπει ένα πλήρες θεωρητικό πλαίσιο. Στόχος αυτής της έρευνας είναι να ρίξει φως στη δυναμική της λειτουργίας συγκέντρωσης, χρησιμοποιώντας θεωρητικές έννοιες της πληροφορίας.

B.5.1 Θεωρητική Ανάλυση της Διαδικασίας **Pooling**

Ας ορίσουμε τον στόχο της συγκέντρωσης από την άποψη της θεωρίας της πληροφορίας.

Definition B.5.1. Έστω ένα βαθύ νευρωνικό δίκτυο με μία λειτουργία συγκέντρωσης μετά από κρυφό στρώμα i. Υποδηλώστε τα χαρακτηριστικά εξόδου του κρυφού στρώματος i ως τυχαία μεταβλητή X και την έξοδο της λειτουργίας συγκέντρωσης ως τυχαία μεταβλητή A. Τότε το κανάλι συγκέντρωσης είναι το κανάλι πληροφοριών που ορίζεται από τον τελεστή συγκέντρωσης.

Ένα κανάλι συγκέντρωσης μπορεί να αναπαρασταθεί από την αλυσίδα Markov στο Σχήμα B.5.1. Η αμοιβαία πληροφορία μεταξύ των δύο τυχαίων μεταβλητών X και A είναι:

$$I(X;A) = H(A) - H(A|X) \tag{B.13}$$

, όπου H(A) είναι η εντροπία και H(A|X) η υπό όρους εντροπία. Η εξίσωση (B.13) ισχύει τόσο για διακριτές όσο και για συνεχείς τυχαίες μεταβλητές, επειδή η αμοιβαία πληροφορία μεταξύ δύο τυχαίων μεταβλητών είναι το όριο της αμοιβαίας πληροφορίας μεταξύ των κβαντισμένων εκδόσεων τους Cover and Thomas (2012). Στη συνέχεια, η χωρητικότητα του καναλιού ορίζεται ως εξής Cover and Thomas (2012):

$$C = \max_{p(x)} I(X;A) \tag{B.14}$$

Το ιδανικό κανάλι θα επέτρεπε τη μεταφορά όλων των πληροφοριών, χωρίς απώλειες. Στην περίπτωση της συγκέντρωσης, ο στόχος είναι να διατηρηθούν όσο το δυνατόν περισσότερες σχετικές πληροφορίες. Όπως συζητήθηκε προηγουμένως, η μέτρηση της συνάφειας των πληροφοριών θα απαιτούσε προηγούμενη γνώση της εργασίας, διαφορετικά η καλύτερη εκτίμησή μας θα ήταν απλώς μια εικασία. Κατά συνέπεια, το καλύτερο κανάλι είναι αυτό που δεν κάνει υποθέσεις και αυτό ερμηνεύεται ως το κανάλι με τη μέγιστη χωρητικότητα. Από το (B.13) είναι προφανές ότι το I(X;A) είναι μέγιστο όταν το H(A) μεγιστοποιείται και το H(A|X) ελαχιστοποιείται. Η ακόλουθη πρόταση περιγράφει τη συμπεριφορά της αμοιβαίας πληροφόρησης για την περίπτωση ενός καναλιού συγκέντρωσης και θα βοηθήσει στον καθορισμό της μέγιστης χωρητικότητάς του.

Proposition B.5.1. Έστω οι τυχαίες μεταβλητές *X* και *A* η είσοδος και η έξοδος ενός καναλιού συγκέντρωσης και *f* η συνάρτηση που περιγράφει τον τελεστή συγκέντρωσης. Η συνάρτηση *f* μπορεί να είναι ντετερμινιστική ή στοχαστική και η αμοιβαία πληροφορία $I(X;A)$ δίνεται από:

(i) Για το *f* που είναι ντετερμινιστικό:

$$I(X;A) = H(A) \tag{B.15}$$

(ii) Για το *f* που είναι στοχαστικό:

$$I(X;A) = H(A) - H(\overrightarrow{r}) \tag{B.16}$$

, όπου το $\overrightarrow{r}$ είναι μια γραμμή του πίνακα μετάβασης *p(A|X)*, με γραμμές που αντιπροσωπεύουν το *A* και στήλες που αντιπροσωπεύουν το *X*.

Proof. Η πρώτη πρόταση της πρότασης B.5.1 αποδεικνύεται με την εκτίμηση του H(A|X) από το (B.13), όταν η συγκέντρωση είναι μια ντετερμινιστική συνάρτηση. Στη συνέχεια, η υπό όρους εντροπία αναπτύσσεται αναλυτικά ως εξής:

$$H(A|X) = \sum_x p(x) H(A|X=x) \tag{B.17}$$

$$H(A|X) = \sum_x p(x) H(1) = 0 \tag{B.18}$$

Ως εκ τούτου, για την περίπτωση μιας ντετερμινιστικής συνάρτησης, η αμοιβαία πληροφορία εξαρτάται μόνο από την εντροπία των χαρακτηριστικών εξόδου και περιγράφεται από την εξίσωση (B.15).

Για τη δεύτερη πρόταση, όπου η συγκέντρωση είναι μια στοχαστική συνάρτηση, ας εξετάσουμε τον πίνακα μετάβασης p(A|X), με τις γραμμές που αντιπροσωπεύουν το A και τις στήλες που αντιπροσωπεύουν το X. Έστω $\overrightarrow{r}$ μια γραμμή του πίνακα μετάβασης, τότε το (B.13) γίνεται (B.16). □

Τα αποτελέσματα σχετικά με τη χωρητικότητα καναλιού της λειτουργίας συγκέντρωσης μπορούν να χρησιμοποιηθούν για την κατανόηση του μηχανισμού της μέγιστης, μέσης και στοχαστικής συγκέντρωσης. Χρησιμοποιώντας τις εξισώσεις που προέκυψαν στην προηγούμενη ενότητα και τις στατιστικές ιδιότητες των μεταβλητών X και A, θα εξηγηθεί πώς λειτουργούν αυτές οι πράξεις.

Οι στατιστικές ιδιότητες της εισόδου X είναι οι ίδιες για κάθε περίπτωση συγκέντρωσης και περιγράφονται σύμφωνα με αυτές τις γραμμές. Έστω το $\overline{x}$ να υποδηλώνει τον μέσο όρο της αναπαράστασης του κοινού χαρακτηριστικού X_N με μέγεθος N και σ_X την τυπική απόκλιση. Οι αντίστοιχοι τύποι είναι:

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{B.19}$$

$$\sigma_X = \frac{1}{N} \sum_{i=1}^{N} (x_i - \overline{x})^2 \tag{B.20}$$

Η μέγιστη συγκέντρωση είναι μια συνάρτηση που με δεδομένο ένα σύνολο τιμών, επιλέγει τη μεγαλύτερη. Η ομαδοποίηση εφαρμόζεται χρησιμοποιώντας πυρήνες, οπότε έστω το X_r μια κοινή αναπαράσταση χαρακτηριστικών ενός πυρήνα με στοιχεία r, τότε η συνάρτηση είναι:

$$f_{max}(X_r) = max(x_1, x_2, ..., x_r) \tag{B.21}$$

Είναι μια ντετερμινιστική συνάρτηση επιλογής και η αμοιβαία πληροφορία μεταξύ της εισόδου και της εξόδου, σύμφωνα με το (B.15), είναι ίση με H(A). Η εξέταση των στατιστικών ιδιοτήτων του max pooling θα αποσαφηνίσει τώρα πώς επηρεάζει το H(A).

Remark B.5.1. Η χωρητικότητα του καναλιού μέγιστης συγκέντρωσης δεν επηρεάζεται από τον χειριστή μέγιστου και αυξάνεται όταν η ιδιότητα των χαρακτηριστικών είναι υψηλή.

Proof. Εξ ορισμού, είναι προφανές ότι ο μέσος όρος θα αυξηθεί, επομένως $\overline{x}_A \geq \overline{x}$. Για να αξιολογηθεί η τυπική απόκλιση, χρειάζεται περισσότερη επεξεργασία. Χρησιμοποιώντας πυρήνες , η τυπική απόκλιση στο (B.20) αναπτύσσεται ως εξής:

$$\sigma_X = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{r}\sum_{j=1, x_{ij}\neq a_i}^{r}(x_{ij}-\overline{x})^2 + \frac{1}{M*r}\sum_{i=1}^{M}(a_i-\overline{x})^2 \tag{B.22}$$

όπου M είναι ο αριθμός των πυρήνων, r ο αριθμός των στοιχείων κάθε πυρήνα και a_i η μέγιστη τιμή κάθε πυρήνα. Υποτίθεται ότι δεν υπάρχει επικάλυψη κατά τη συγκέντρωση και άρα $N = r * M$. Η απόδειξη με επικάλυψη είναι παρόμοια. Στη συνέχεια, η τυπική απόκλιση της μέγιστης συγκέντρωσης δίνεται από:

$$\sigma_A = \frac{1}{M}\sum_{i=1}^{M}(a_i-\overline{x}_A)^2 \tag{B.23}$$

Συγκρίνοντας τα (B.22) και (B.23), συμπεραίνεται ότι η αλλαγή της τυπικής απόκλισης εξαρτάται από την κατανομή των τιμών των χαρακτηριστικών εισόδου. Η είσοδος καθορίζει επίσης την καρδινικότητα του A, η οποία επηρεάζει επίσης την εντροπία. Από τον ορισμό της εντροπίας, υψηλότερη καρδινικότητα σημαίνει υψηλότερη εντροπία. □

Έτσι, η μέγιστη λειτουργία δεν ελέγχει την εντροπία της εξόδου και δεν υπάρχει καμία εγγύηση ότι θα μεγιστοποιήσει την ικανότητα συγκέντρωσης. Η χωρητικότητα του καναλιού αποδεικνύεται ότι εξαρτάται από την κατανομή των χαρακτηριστικών εισόδου και την ιδιότητα της εξόδου A. Αυτό είναι σε καλή συμφωνία με το εύρημα των Boureau et al. (2010), ότι η μέγιστη συγκέντρωση είναι καλά υιοθετημένη σε σπάνιες ενεργοποιημένες λειτουργίες.

Η μέση συγκέντρωση αναλύεται με τον ίδιο τρόπο, επειδή ο υπολογισμός του μέσου όρου είναι μια ντετερμινιστική συνάρτηση.

$$f_{avg}(X_r) = \frac{1}{r}\sum_{j=1}^{r}x_j \tag{B.24}$$

Επομένως, η αμοιβαία πληροφορία περιγράφεται από την (B.15).

Remark B.5.2. Η έξοδος του μέσου καναλιού συγκέντρωσης τείνει να είναι ομοιόμορφη.

Η απόδειξει μπορεί να βρεθεί στο αντίστοιχο κεφάλαιο της διατριβής.

Μετά την πρόταση B.5.2, η κατανομή εξακολουθεί να εξαρτάται από τα δεδομένα εισόδου. Τα μέγιστα μοναδικά στοιχεία της τελικής διανομής είναι M και συμβαίνει όταν κάθε πυρήνας δίνει μια μοναδική τιμή. Ο μηχανισμός της μέσης συγκέντρωσης δεν επηρεάζει τον αριθμό των μοναδικών τιμών εξόδου και επομένως δεν εγγυάται τη μεγιστοποίηση της εντροπίας.

Στην πράξη, τις περισσότερες φορές η παραγωγή της μέσης συγκέντρωσης θα επαναλαμβανόταν σπάνια. Ένα καλό παράδειγμα είναι η ειδική περίπτωση εικόνων που απεικονίζουν μοτίβα όπως μια σκακιέρα. Με πυρήνες μεγέθους 2ξ2 και υποθέτοντας ένα μοτίβο σκακιέρας με τιμές μηδέν και ένα, η μέση τιμή θα είναι πάντα η ίδια.

Η περίπτωση της στοχαστικής συγκέντρωσης είναι διαφορετική, αφού είναι μια μη ντετερμινιστική συνάρτηση επιλογής. Επομένως, χρησιμοποιώντας το δεύτερο μέρος της πρότασης B.5.1, εξάγουμε την παρακάτω πρόταση.

Proposition B.5.2. Η χωρητικότητα του καναλιού στοχαστικής συγκέντρωσης παραμετροποιείται ασθενώς από τη στοχαστική συνάρτηση.

Η απόδειξει μπορεί να βρεθεί στο αντίστοιχο κεφάλαιο της διατριβής.

Η αμοιβαία πληροφορία της στοχαστικής συγκέντρωσης εξαρτάται τόσο από το H(A) όσο και από το H(A|X). O πρώτος όρος εξακολουθεί να είναι απαλλαγμένος από οποιαδήποτε παράμετρο του καναλιού και εξαρτάται από τα δεδομένα. Η διαφορά σε σχέση με τη μέγιστη και τη μέση συγκέντρωση είναι ότι η στοχαστική συγκέντρωση έχει κάποιο έλεγχο στις αμοιβαίες πληροφορίες μέσω της παραμέτρου a.

Το αποτέλεσμα της προηγούμενης ανάλυσης είναι ότι οι τρέχουσες λύσεις συγκέντρωσης δεν έχουν τις ιδιότητες για να χειριστούν την εντροπία της εξόδου H(A) και η απόδοσή τους επηρεάζεται σε μεγάλο βαθμό από την κατανομή των δεδομένων. Αυτό το πρόβλημα αντιμετωπίζεται επανεξετάζοντας το πρόβλημα της δειγματοληψίας μέγιστης εντροπίας και σχεδιάζοντας μια νέα λειτουργία συγκέντρωσης βασισμένη στην εντροπία.

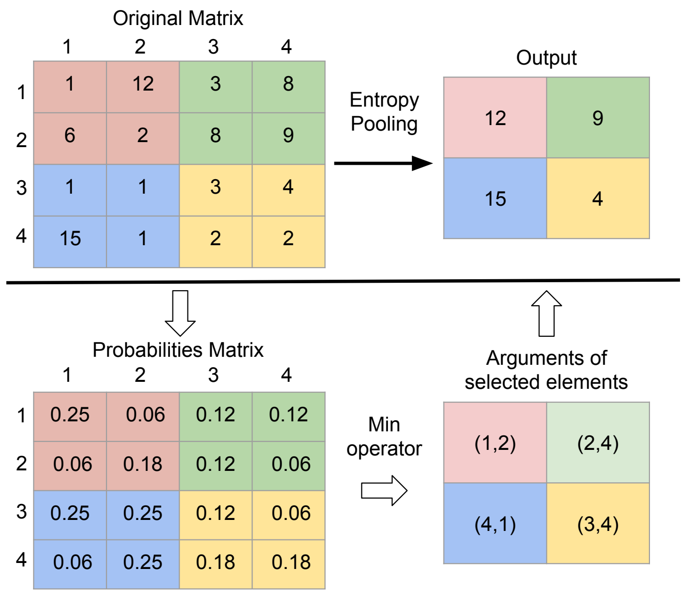


Figure B.5.2: Η διαδικασία enropy pooling. Ο συνάρτηση ελαχίστου επιλέγει τα πιο σπάνια χαρακτηριστικά.

Ο αλγόριθμος συγκέντρωσης εντροπίας υπολογίζει τις πιθανότητες p για κάθε χάρτη χαρακτηριστικών με μέγεθος N. Στη συνέχεια, ο χάρτης των πιθανοτήτων χωρίζεται σε περιοχές, σύμφωνα με το καθορισμένο μέγεθος του πυρήνα και τα βήματα, με τον ίδιο τρόπο όπως στις κλασικές πράξεις συγκέντρωσης. Για κάθε περιοχή επιλέγεται το στοιχείο με τη μικρότερη πιθανότητα. Η διαδικασία απεικονίζεται στο Σχ. B.5.2. Ένας επίσημος ορισμός της συγκέντρωσης εντροπίας δίνεται παρακάτω.

Definition B.5.2. Η πράξη entropy pooling είναι ένα κανάλι συγκέντρωσης που λειτουργεί σύμφωνα με την ντετερμινιστική συνάρτηση f, που περιγράφεται από τον ακόλουθο τύπο:

$$f_{entr}(X_r) = X_r[g(P_r)], \tag{B.25}$$

$$g(P_r) =_{1\leq i\leq r} p_i \tag{B.26}$$

, όπου X_r είναι ο χάρτης χαρακτηριστικών εισόδου, P_r ο κατασκευασμένος χάρτης των πιθανοτήτων των χαρακτηριστικών και r το μέγεθος μιας περιοχής. Η ικανότητα συγκέντρωσης εντροπίας δίνεται από:

$$I(X;A) = H(f_{entr}(X_r)) \tag{B.27}$$

B.5.2 Μία Μέθοδος Μεταβλητότητας της Διαδικασίας **Pooling** με Βάση την Αρχή **Information Bottleneck**

Η μέθοδος VIB-Pooling είναι ένα πρωτότυπο επίπεδο συγκέντρωσης για βαθιά νευρωνικά δίκτυα που βασίζεται σε θεμελιώδεις αρχές. Το VIB-Pooling είναι μια διαδικασία παραλλαγής που δεν έχει παραμέτρους εκμάθησης. Σύμφωνα με την αρχή Information Bottleneck, επιλέγει χαρακτηριστικά με βάση την αρχή της Μέγιστης Εντροπίας και προσθέτει έναν όρο κανονικοποίησης στην αντικειμενική συνάρτηση του νευρωνικού δικτύου. Η μεθοδολογία χρησιμοποιεί μια μεταβλητή προσέγγιση και προσαρμόζεται αυτόματα για να επιλέξει σχετικά χαρακτηριστικά μέσω της χρήσης ενός στοιχείου που είναι γνωστό ως Online Adaptation. Παρακάτω περιγράφεται η συνάρτηση κόστους που πληρεί τα κριτήρια της αρχής Information Bottleneck για ένα Γκαουσιανό κανάλι πολλάπλών μεταβλητών. Στη συνέχεια ακολουθεί μία συνοπτική περιγραφή ολόκληρης της αρχιτεκτονικής του προτεινόμενου μοντέλου.

Proposition B.5.3. ***VIB-Pooling cost*** Έστω $\boldsymbol{z} \sim p(\boldsymbol{z}) = N(\boldsymbol{z}|\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$ να είναι μια πολυδιάστατατη Γκαουσιανή τυχαία μεταβλητή με *i.i.d.* στοιχεία. Χρησιμοποιώντας το τέχνασμα επαναπαραμετροποίησης *(**?**)* έχουμε $p(z|x)ds = p(\epsilon)d\epsilon$ όπου $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{\epsilon})$ είναι μία ντετερμενιστική συνάρτηση $\boldsymbol{x}$ και η Γκαουσιανή τυχαία μεταβλητή $\boldsymbol{\epsilon} \sim N(\boldsymbol{\epsilon}|\boldsymbol{\mu}_\epsilon, \boldsymbol{\Sigma}_\epsilon)$. Τότε ισχύει

$$KL(p(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))) = log(\frac{|\boldsymbol{\Sigma}_z|}{|\boldsymbol{\Sigma}_\epsilon|}) + tr(\boldsymbol{\Sigma}_z^{-1}\boldsymbol{\Sigma}_\epsilon) + (\boldsymbol{\mu}_z - \boldsymbol{\mu}_\epsilon)^T \boldsymbol{\Sigma}_z^{-1}(\boldsymbol{\mu}_z - \boldsymbol{\mu}_\epsilon) - n \tag{B.28}$$

Η αρχιτεκτονική του VIB-Pooling απεικονίζεται στο σχήμα B.5.3. Πρώτον, η είσοδος της μονάδας αποτελείται από χαρακτηριστικά με διαστάσεις $batch\_size \times r \times c$. Στη συνέχεια, υπολογίζονται δύο πίνακες και καθένας από αυτούς έχει διαστάσεις $r \times c$. Ένας από τους πίνακες αντιπροσωπεύει τις διακυμάνσεις των χαρακτηριστικών σε σχέση με τον αριθμό των δειγμάτων που ισούται με $batch\_size$. Το άλλο περιλαμβάνει τις μέσες τιμές των χαρακτηριστικών. Χρησιμοποιώντας τις τιμές αυτών των δύο πινάκων ως μέσους και διακυμάνσεις, έχουμε έναν τυχαίο πίνακα με σχήμα $r \times c$, τα στοιχεία του οποίου αντιπροσωπεύουν κανονικές κατανομές των χαρακτηριστικών. Χρησιμοποιώντας τον πίνακα διακυμάνσεων και τη συνάρτηση επιλογής από την πρόταση 6.2.1, ο πίνακας βαθμολογίας ενημερώνεται σύμφωνα με τον αλγόριθμο προσαρμογής **??**. Στη συνέχεια, ο πίνακας βαθμολογίας χρησιμοποιείται για τον υπολογισμό της σταθμισμένης μέσης δεξαμενής δίνοντας έναν τυχαίο πίνακα μεταβλητών Γκάους με μέγεθος $r' \times c'$. Οι διαστάσεις r' και c' εξαρτώνται από το μέγεθος του πυρήνα της συγκέντρωσης. Στη συνέχεια, χρησιμοποιώντας το τέχνασμα επαναπαραμετροποίησης, η έξοδος του επιπέδου είναι ένας πίνακας $r' \times c'$ με στοιχεία τα χαρακτηριστικά που έχουν επαναδειγματοληψία. Η απόκλιση KL υπολογίζεται σύμφωνα με την πρόταση B.5.3 και χρησιμοποιείται ως όρος κανονικοποίησης της αντικειμενικής συνάρτησης.

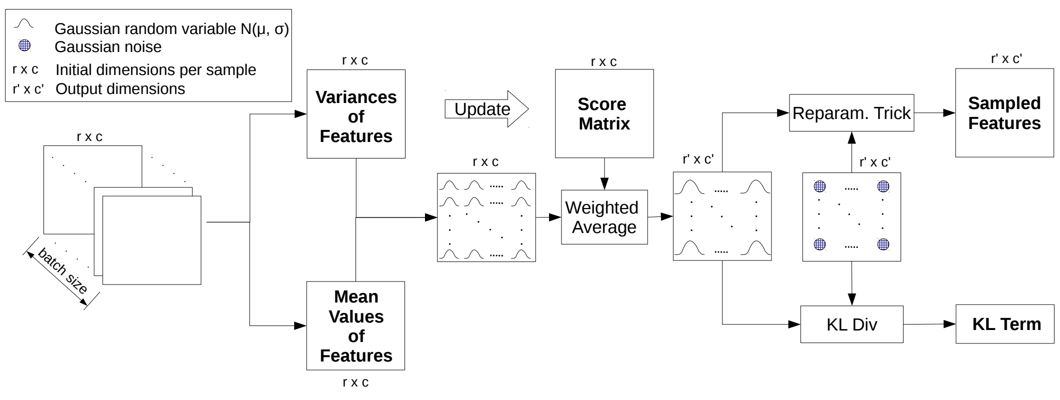


Figure B.5.3: VIB-Pooling.

Table B.5.1: Πειραματικά αποτελέσματα που δείχνουν την ακρίβεια του υπολειπόμενου νευρωνικού δικτύου με διαφορετικές λειτουργίες συγκέντρωσης. Τα πειράματα καλύπτουν τα τέσσερα διαφορετικά σενάρια: ¨ναιε αππροαςη (ΦΦΦ)¨, ¨κνοων νοισε (ΤΤΤ)¨, ¨νοισε αυγμεντατιον (ΤΤΦ)’ ανδ ¨ουτ οφ διστριβυτιον (ΦΦΤ)¨.

Pooling config.	FFF	TTT	FFT	TTF
AVG - AVG	0.988	0.967	0.683	0.961
ENTR - AVG	0.984	0.949	0.644	0.961
MAX - AVG	0.993	0.955	0.640	0.971
AVG - ENTR	**0.996**	0.966	**0.685**	0.970
ENTR - ENTR	0.983	0.956	0.659	0.975
MAX - ENTR	0.987	0.943	0.655	**0.977**
AVG - MAX	0.992	**0.968**	0.681	0.976
ENTR - MAX	0.989	0.962	0.656	0.955
MAX - MAX	0.992	0.966	0.654	0.976

B.5.3 Περίληψη Πειραματικών Αποτελεσμάτων

Ένας από τους στόχους αυτής της μελέτης είναι να δημιουργήσει αποτελεσματικά συστήματα βαθιάς μάθησης που να είναι ανθεκτικά σε θορύβους σε εργασίες ταξινόμησης ήχου. Η αναγνώριση ομιλητών και η αναγνώριση φωνητικών εντολών είναι οι εργασίες για την αξιολόγηση των προτεινόμενων μοντέλων. Το πρώτο αφορά την αναγνώριση πέντε διαφορετικών ομιλητών χρησιμοποιώντας το γνωστό σύνολο δεδομένων "Speaker Recognition Dataset Prominent Leaders Speeches". Το τελευταίο επιδιώκει να ανιχνεύσει δέκα φωνητικές εντολές από το δημοφιλές σύνολο δεδομένων της Google "Google Speech Commands". Δύο μοναδικές νευρωνικές αρχιτεκτονικές δημιουργούνται χρησιμοποιώντας αυτά τα σύνολα δεδομένων. Αυτά τα μοντέλα υπερτερούν των πανομοιότυπων μοντέλων τελευταίας τεχνολογίας όσον αφορά την απόδοση και είναι υπολογιστικά ελαφρύτερα λόγω λιγότερων παραμέτρων εκμάθησης. Επιπλέον, τα προτεινόμενα μοντέλα χρησιμοποιούν ομαδοποίηση χαρακτηριστικών με βάση την εντροπία, η οποία αυξάνει την ανθεκτικότητά τους στο θόρυβο.

Το νευρωνικό δίκτυο που βασίζεται στην υπολειπόμενη αρχιτεκτονική αξιολογείται στην εργασία

Table B.5.2: Αποτελέσματα που δείχνουν την ακρίβεια του δισδιάστατου συνελικτικού δικτύου με διαφορετικές λειτουργίες συγκέντρωσης. Το M σημαίνει max pooling και E για συγκέντρωση εντροπίας. Η πρώτη στήλη καλύπτει την αφελή προσέγγιση και οι υπόλοιπες περιλαμβάνουν θόρυβο στη δοκιμή με συντελεστή κλιμάκωσης.

Pool conf.	FFF	x0.25	x0.5	x1	x2
MMMM	**0.932**	0.856	0.782	0.706	0.620
MMEM	0.931	0.875	**0.811**	**0.720**	0.637
MMME	0.926	0.861	0.805	0.707	0.611
EMMM	0.923	**0.876**	0.805	0.677	0.512
MEMM	0.923	0.870	0.810	0.713	0.605
EMEM	0.923	0.836	0.775	0.686	0.556
MEEM	0.917	0.860	0.795	0.717	0.641
EMME	0.916	0.854	0.787	0.712	**0.644**
EEMM	0.916	0.826	0.748	0.645	0.540
EEME	0.915	0.830	0.755	0.624	0.581
EEEM	0.914	0.846	0.785	0.702	0.619
MEME	0.909	0.833	0.746	0.602	0.437
MMEE	0.907	0.843	0.773	0.696	0.634
EEEE	0.901	0.792	0.724	0.668	0.638
EMEE	0.898	0.841	0.776	0.708	0.628
MEEE	0.888	0.816	0.734	0.668	0.616

αναγνώρισης ηχείου χρησιμοποιώντας το σύνολο δεδομένων "Speaker Recognition Dataset Prominent Leaders Speeches". Το μοντέλο διαμορφώνεται μέσω μιας συστηματικής πειραματικής μελέτης που αξιολογεί όλους τους πιθανούς συνδυασμούς των επιπέδων συγκέντρωσης μέγιστου, μέσου όρου και εντροπίας. Η διαδικασία επαναλαμβάνεται και για τα τέσσερα σενάρια που περιγράφηκαν προηγουμένως. Προκειμένου να γίνει δίκαιη η σύγκριση των διαφορετικών παραλλαγών του μοντέλου, κάθε μία εκπαιδεύεται, αξιολογείται και δοκιμάζεται πολλές φορές. Στη συνέχεια, υπολογίζεται ο μέσος όρος και η τυπική απόκλιση των αποτελεσμάτων ακρίβειας και συγκρίνονται για να βρεθεί το πιο ισχυρό μοντέλο.

Ο πίνακας B.5.1 δείχνει τα αποτελέσματα όλων των διαφορετικών συνδυασμών των τριών πράξεων συγκέντρωσης για το υπολειπόμενο νευρωνικό δίκτυο. Σε αυτόν τον πίνακα το όνομα της διαμόρφωσης συγκέντρωσης έχει δύο μέρη. Το πρώτο μέρος αναφέρεται στη συγκέντρωση του υπολειπόμενου μπλοκ και το δεύτερο στην τελευταία συγκέντρωση ολόκληρου του δικτύου. Για παράδειγμα, εάν το υπολειπόμενο μπλοκ έχει το μέγιστο επίπεδο συγκέντρωσης και η τελευταία συγκέντρωση του δικτύου είναι η εντροπία, η διαμόρφωση συγκέντρωσης αναφέρεται ως MAX-ENTR. Οι πιο στιβαρές εκδόσεις της αρχιτεκτονικής είναι οι AVG-ENTR, AVG-MAX, AVG-ENTR και MAX-ENTR για τα σενάρια "naive approach (FFF)", "known noise (TTT)", "out of distribution (FFT)" και "noise augmentation (TTF)" αντίστοιχα.

Η δισδιάστατη συνεκτική αρχιτεκτονική αξιολογείται στο σύνολο δεδομένων Google Speech Commands. Αυτό το μοντέλο περιλαμβάνει τέσσερα στρώματα συγκέντρωσης. Τα πειράματα περιλαμβάνουν όλες τις παραλλαγές του μοντέλου συνδυάζοντας μέγιστο και εντροπικό στρώμα συγκέντρωσης. Για συντομία αναφερόμαστε στις αρχιτεκτονικές με τέσσερα γράμματα ανάλογα με τον τύπο των στρώσεων συγκέντρωσης. Για παράδειγμα, το MEME θα ήταν η αρχιτεκτονική όπου οι πρώτοι έως οι τελευταίοι τύποι συγκέντρωσης είναι max, entropy, max και entropy. Οι μετρήσεις που χρησιμοποιούνται είναι η ακρίβεια και το σφάλμα διασταυρούμενης εντροπίας. Τα πειραματικά περιβάλλοντα είναι η αφελής προσέγγιση και το σενάριο εκτός διανομής. Ο πίνακας B.5.2 παρουσιάζει τα αποτελέσματα με την ακρίβεια κάθε μοντέλου. Η πρώτη στήλη αντιπροσωπεύει την αφελή προσέγγιση όπου δεν υπάρχει θόρυβος. Τα δύο καλύτερα μοντέλα είναι τα MMMM και MMEM, με το πρώτο να έχει ελαφρώς καλύτερη απόδοση. Οι άλλες τέσσερις στήλες δείχνουν τα αποτελέσματα όταν υπάρχει θόρυβος στο περιβάλλον δοκιμών με διάφορους παράγοντες κλιμάκωσης. Για την περίπτωση όπου το πλάτος του θορύβου κλιμακώνεται κατά 0,25, το πιο στιβαρό μοντέλο είναι το EMMM. Τα MEMM και MMEM παρουσιάζουν πολύ παρόμοια απόδοση. Για τις περιπτώσεις με συντελεστές κλιμάκωσης 0,5 και 1, το καλύτερο μοντέλο είναι το MMEM ακολουθούμενο από το MEMM και στις δύο περιπτώσεις. Στην τελευταία περίπτωση με τον συντελεστή κλιμάκωσης 2, την καλύτερη απόδοση δείχνει το EMME και ακολουθούν τα MEEM, EEEE και MMEM. Συνολικά, το MMEM φαίνεται το πιο πολλά υποσχόμενο όταν το μοντέλο πρέπει να αναπτυχθεί σε περιβάλλον με απροσδόκητους θορύβους.

Στη συνέχεια ο στόχος των πειραμάτων είναι να αποδειχθεί ότι το VIB-Pooling βελτιώνει την απόδοση των τυπικών νευρωνικών δικτύων κατά την ταξινόμηση των δεδομένων ιατρικών χρονοσειρών. Το VIB-Pooling συγκρίνεται με τη μέγιστη και τη μέση συγκέντρωση, προκειμένου να επαληθευτεί ότι μπορεί να αντικαταστήσει με επιτυχία αυτά τα στρώματα. Το ResNet επιλέγεται ως γραμμή βάσης για όλες τις εργασίες χρονοσειρών και προσαρμόζεται ανάλογα για κάθε πρόβλημα και μέγεθος των δεδομένων. Οι λεπτομέρειες της υλοποίησης περιγράφονται παρακάτω.

Το ResNet18 περιλαμβάνει μία λειτουργία ομαδοποίησης μετά τα υπολειπόμενα μπλοκ και πριν από το τελευταίο γραμμικό στρώμα. Συνήθως πρόκειται για μια μέση συγκέντρωση. Για τους σκοπούς αυτών των πειραμάτων, το προεπιλεγμένο επίπεδο συγκέντρωσης αντικαθίσταται επίσης από τη μέγιστη συγκέντρωση και την VIB-Pooling. Η εκπαίδευση των μοντέλων διαρκεί έως και 70 εποχές με πρόωρη διακοπή και αποθήκευση του καλύτερου μοντέλου. Οι μετρήσεις που χρησιμοποιούνται είναι η ακρίβεια και η περιοχή κάτω από την καμπύλη ROC (AUC). Όλα τα πειράματα εκτελούνται 100 φορές και ο πίνακας B.5.3 παραθέτει τη μέση και τυπική απόκλιση των δύο μετρήσεων. Το VIB-Pooling ξεπερνά τους άλλους δύο διεκδικητές για την πλειονότητα των εργασιών. Επιτυγχάνει έως και 7,5% καλύτερη ακρίβεια από τη μέση συγκέντρωση με τη μεγαλύτερη διαφορά να λαμβάνει χώρα για το σύνολο δεδομένων DistalPhalanxOutlineAgeGroup.

B.6 Συμπεράσματα

Αυτή η διατριβή επικεντρώθηκε στις προσεγγίσεις μηχανικής μάθησης για προβλήματα χρονοσειρών. Οι εργασίες τεχνητής νοημοσύνης που καθόρισαν τους ερευνητικούς μας στόχους προέρχονται από τρεις διαφορετικούς τομείς που καλύπτουν την ενεργειακή κατανομή, την αναγνώριση ομιλίας και την ανάλυση των αρχείων υγείας. Αυτά τα πεδία διερευνήθηκαν σε βάθος, ειδικά για το πρόβλημα του NILM που είναι μια γνωστή NP-hard εργασία στην κατηγορία των προβλημάτων διαχωρισμού τυφλών πηγών. Τα πρωτότυπα συστήματα μηχανικής μάθησης και οι νευρωνικές αρχιτεκτονικές σχεδιάστηκαν με στόχο την επίλυση των στοχευμένων εργασιών AI. Τα προτεινόμενα πλαίσια και αλγόριθμοι, όπως το Signal2Vec, δεν περιορίζονται στις στοχευμένες εργασίες, αλλά μπορούν να χρησιμοποιηθούν σε οποιαδήποτε άλλα προβλήματα χρονοσειρών. Επιπλέον, η εργασία μας συμβάλλει στα θεωρητικά θεμέλια κατά το σχεδιασμό βαθιών νευρωνικών δικτύων χρησιμοποιώντας αρχές θεωρίας πληροφοριών και γεφυρώνοντας το χάσμα μεταξύ θεωρίας και πράξης στον τομέα της βαθιάς μάθησης.

Table B.5.3: Αποτελέσματα δοκιμών σε ιατρικές χρονοσειρές χρησιμοποιώντας ResNet18. Η προεπιλεγμένη αρχιτεκτονική ResNet18 περιλαμβάνει μια μέση συγκέντρωση. Η έκδοση ResNet15 VIB αντικαθιστά την προεπιλεγμένη λειτουργία συγκέντρωσης με το VIB-Pooling.

Model	ResNet18					
Metric	Acc			AUC		
Test Data	Avg	Max	VIB	Avg	Max	VIB
ECG200	0.767 ± 0.068	0.729 ± 0.101	**0.789** ± **0.047**	0.85 ± 0.081	0.759 ± 0.167	**0.867** ± **0.058**
ECGFiveDays	0.73 ± 0.152	0.702 ± 0.17	**0.735** ± **0.141**	0.863 ± 0.122	0.835 ± 0.155	**0.876** ± **0.101**
TwoLeadECG	0.564 ± 0.093	0.563 ± 0.069	**0.568** ± **0.078**	**0.644** ± **0.144**	0.629 ± 0.14	0.638 ± 0.111
ECG5000	0.931 ± 0.008	0.924 ± 0.059	**0.933** ± **0.007**	0.869 ± 0.017	0.866 ± 0.042	**0.874** ± **0.017**
CinC_ECG_torso	0.878 ± 0.062	0.886 ± 0.886	**0.891** ± **0.063**	0.926 ± 0.041	0.933 ± 0.932	**0.933** ± **0.04**
DistalPhalanxOutlineCorrect	0.575 ± 0.098	**0.612** ± **0.096**	0.602 ± 0.088	0.601 ± 0.072	**0.6** ± **0.094**	0.591 ± 0.062
DistalPhalanxTW	0.529 ± 0.226	0.465 ± 0.192	**0.561** ± **0.206**	0.668 ± 0.171	0.545 ± 0.132	**0.678** ± **0.174**
MedicalImages	0.605 ± 0.045	0.586 ± 0.045	**0.608** ± **0.042**	0.895 ± 0.027	0.878 ± 0.049	**0.898** ± **0.047**
ProximalPhalanxOutlineAgeGroup	**0.764** ± **0.139**	0.718 ± 0.166	0.759 ± 0.142	**0.858** ± **0.143**	0.798 ± 0.183	0.857 ± 0.139
PhalangesOutlinesCorrect	0.679 ± 0.062	**0.732** ± **0.058**	0.694 ± 0.066	0.72 ± 0.076	**0.786** ± **0.063**	0.727 ± 0.088
DistalPhalanxOutlineAgeGroup	0.66 ± 0.177	0.635 ± 0.164	**0.709** ± **0.144**	0.683 ± 0.188	0.642 ± 0.176	**0.764** ± **0.159**
NonInvasiveFatalECG_Thorax2	0.835 ± 0.148	0.797 ± 0.2	**0.852** ± **0.089**	0.98 ± 0.085	0.965 ± 0.118	**0.99** ± **0.049**
ProximalPhalanxOutlineCorrect	0.726 ± 0.045	0.737 ± 0.044	**0.737** ± **0.043**	0.832 ± 0.049	0.84 ± 0.037	**0.835** ± **0.04**
ProximalPhalanxTW	0.574 ± 0.2	0.464 ± 0.18	**0.604** ± **0.188**	0.702 ± 0.177	0.597 ± 0.148	**0.724** ± **0.171**