New Prototype Concepts in Classification Learning

Sascha Saralajew

New Prototype Concepts in Classification Learning

A dissertation submitted in partial fulfillment of the requirements for the degree of Doktor der Naturwissenschaften (Dr. rer. nat.) at the Faculty of Technology at Bielefeld University

by

Sascha Saralajew

in February, 2020.

Supervised by:
Prof. Dr. rer. nat. Barbara Hammer and Prof. Dr. rer. nat. habil. Thomas Villmann.

Reviewed by:
Prof. Dr. rer. nat. Barbara Hammer;
Prof. Dr. rer. nat. Thomas Martinetz;
Prof. Dr. rer. nat. habil. Thomas Villmann.

In cooperation with:
Dr. Ing. h.c. F. Porsche AG, Weissach, Germany;
University of Applied Sciences Mittweida, Mittweida, Germany.

Defended on November 18th, 2020.

It's not what it is... It's what it could be.

Inspired by the final credits of the movie:
Chris Akrigg – Brakeless pt.. 1.

Acknowledgments

It is difficult to keep the balance between work and private life, especially during a PhD when you are motivated, excited, frustrated, and disappointed—all at the same time—about your research. That is why it is indescribably important and not to honor with words, to have fixed points. Without a particular order, I would like to thank these fixed points:

Thomas Villmann for supervising my dissertation, for listening to all my questions and thoughts and for your constant support. I am also very grateful to you for bringing me into such a great research group—cheers, to all the members!

Barbara Hammer and Thomas Martinetz for reviewing this thesis—thanks a lot! Barbara, I am also very grateful to you for giving me the opportunity to do my PhD in Bielefeld and for always helping me when I needed help.

My colleagues at Porsche AG. Especially Sebastian and Torsten, who gave me the chance to work in this area and made all this possible in the first place.

All my colleagues from the amazing EFH Innovation Campus team: Simon^2, Gennady, Christian, Matthias^2, Ebu, Lars^2, Emilio, Rikardo, Stefan, Lukas, Mathis, and all the others. I really enjoyed the time, the discussions, the leisure activities, the disturbing NerfGun fights, the Business-Trips, the Patent Fridays, and so on. Thanks also to all the students with whom I was allowed to work on my projects. All your contributions helped me to achieve this goal.

The people I met during my time in California. It was such a great experience to work with you, discuss scientific topics and discover the country. Thanks to Nils (plus Adrienne), Joe, Philip, Hannes, Sai, Emre, Maike, Melanie, Kacem (still the best project manager), Mathias (unforgettable: Moab, baby!), and all the others.

My family and friends who make my life colorful: My Muggie's (for recovering with jam sessions), all my friends who share my two-wheeled passion (for the refreshing times in the mountains and to Ferdi for proofreading the whole thesis), to Villy (for showing me the taste of whisk(e)y, wonderful journeys, and becoming a friend), and *my beloved Katharina—no words that could describe my thankfulness to you.*

Abstract

Machine learning algorithms are becoming more and more important in everyday life. Applications in search engines, driver assistance systems, consumer electronics, and so on use them heavily and would not be as powerful without them. Neural Networks (NNs), for example, are state-of-the-art classification approaches and dominate the field. However, they are difficult to interpret and not fully understood. For instance, the existence of adversarial examples that are imperceptible to humans contradicts the general belief that convolutional NNs classify objects in images mainly by breaking them down into increasingly complex object shapes. In this thesis, we study prototype-based classification algorithms with the goal of improving the classification capabilities of such algorithms while simultaneously preserving robustness and interpretability properties. Moreover, we investigate how properties of prototype-based classification algorithms can be transferred to NNs in order to increase their interpretability. First, we derive the concept of set-prototypes and apply it in a Learning Vector Quantization (LVQ) framework—a well-understood classification algorithm. We examine the mathematical properties and show that the derived method is provably robust against adversarial attacks. Furthermore, the method consistently outperforms other LVQ approaches while still being interpretable. Second, we relax the class-specific prototype concept to that of components and apply it in LVQ- and NN-based classifiers. This framework provides promising interpretation techniques for NNs. For example, we use them to explain how an adversarial attack is fooling an NN. We evaluate the methods on both toy and real-world datasets, including Indian Pine, MNIST, CIFAR-10, GTSRB, and ImageNet.

Contents

Chapter 1
Introduction

It is indisputable that we are experiencing a digital revolution right now. We are working on the development of self-driving cars, talking about smart homes and cities, and the transformation of the conventional industry to Industry 4.0. A common buzzword found in all these topics is *Machine Learning* (ML), nowadays better known as artificial intelligence. Even though most people only associate the aforementioned topics with ML, it has been part of our daily life for years. For example, spam filters, online purchase recommendations, spell checkers, translators, movie animations, internet search engines, computer game agents, and so on have been taking advantage of this technology for a long time now.

But to start with, what is ML? It is the selection or definition of an algorithm based on *data* or available information to generate or imitate a desired behavior. As a simple and naive example, consider the common mathematical school task of fitting a polynomial function of a certain degree based on a set of support points. The support points are the dataset and the polynomial function of a certain degree is the ML model. By fitting the polynomial to the support points (usually by solving a system of linear equations), we *train* the model on the dataset. After the training, we can evaluate the polynomial for arbitrary arguments to obtain predictions. In school, such trained "ML models" are linked to questions about the deepest point of a suspension bridge.

So what is different in current ML applications? Instead of having a simple polynomial function, we have complex concatenations of several nonlinear operations with thousands or millions of trainable parameters. Additionally, fitting these models is—very often—based on big datasets and the common idea to describe a desired behavior by data. Based on this, we usually train the models via iterative optimization schemes. This relatively simple principle is very powerful and has led to models with human or *superhuman* performance. For instance, Silver et al. (2017) have created a model called AlphaGo Zero that defeats the best players in the board game Go. Moreover, this principle was used to train an algorithm that generated a modern art image that was sold at the famous auction house CHRISTIE'S for a groundbreaking price of $ 432 500.[1]

If we measure the difficulty of a particular task by how many people in the world can do it, we might wonder why we do not have self-driving cars yet, considering that most people—even if they cannot play Go or draw like an artist—can drive a car. Leaving aside the fact that this argumentation is not really scientific, it clearly reflects the discrepancy between humans and ML methods in terms of the ability to solve tasks.[2]

With the development and provision of ML, there comes responsibility and, more notably, liability for the providers—especially in the field of self-driving cars. Both AlphaGo Zero and a self-driving car have to interact with humans, but the latter could cause a hazard for other humans in case of false decisions. Therefore, it is natural to ask whether an ML method always works correctly. Unfortunately, this question cannot be easily answered in modern ML architectures as it requires a certain understanding and interpretation of how the model arrives at its decision.

Currently, the most frequently used ML architectures are (deep) *Neural Networks* (NNs)—for example, see I. Goodfellow, Bengio, and Courville (2016) for an introduction to NNs. Usually, these methods far surpass other methods like support vector machines, k-nearest neighbors classifiers, logistic regression approaches, and so on in terms of accuracy. However, NN architectures frequently act as black-boxes and, hence, are hard to interpret.

An example of this difficulty is that it is not fully understood how an NN models a decision or, more generally, what an NN has learned about the dataset. These problems have led to the very active field of research about *adversarial examples* (e. g., I. J. Goodfellow, Shlens, & Szegedy, 2015; Szegedy et al., 2014). Roughly speaking, adversarial examples are input samples that are manipulated in such a way that they cause a drastic change of the network prediction compared to the original input. Furthermore, this manipulation should be imperceptible to humans or the manipulated input should still be correctly predicted by a human expert. At the moment, it seems that all NNs can be fooled and every defense proposed so far can be broken (e. g., Carlini, 2019; Carlini & Wagner, 2017; Eykholt et al., 2018; D. Wang, Li, Wen, Nepal, & Xiang, 2019). Therefore, the question of how this is possible remains, even though there already have been attempts to provide formal guarantees for the robustness of NNs (e. g., Croce, Andriushchenko, & Hein, 2019;

[1]Christie's. (2018, December 12). Is artificial intelligence set to become art's next medium? Retrieved from `https://www.christies.com/features/A-collaboration-between-two-artists-one-human-one-a-machine-9332-1.aspx`

[2]This observation is sometimes called the difference between weak and strong, or narrow and full artificial intelligence.

Hein & Andriushchenko, 2017; Singla & Feizi, 2019; Wei & Ma, 2019; H. Zhang, Weng, Chen, Hsieh, & Daniel, 2018).

Another example for the demanded interpretability of NNs is the ongoing research about visualization techniques. In the last years, a lot of different visualization techniques have been proposed to explain the decision of NNs (e. g., Erhan, Bengio, Courville, & Vincent, 2009; Nguyen, Yosinski, & Clune, 2019; Zeiler & Fergus, 2014; Zhou, Khosla, Lapedriza, Oliva, & Torralba, 2016). Recently, however, the trustworthiness of some of these techniques for the interpretation of a model decision has been questioned (Adebayo et al., 2018). But much worse, Geirhos et al. (2019) showed that the widely accepted property that *Convolutional NNs* (CNNs) "[recognize] objects by learning increasingly complex representations of object shapes" (p. 1) might generally not be true and that image texture is much more important than expected.

All of these aforementioned aspects prevent companies from deploying NNs for high-stakes decisions as nobody can guarantee their correct operation in every situation. This could also be the reason why there are still no commercially available self-driving cars. Rudin (2019) summarized these difficulties of explaining black-box models and requested: "Stop explaining [black-box] machine learning models for [high-stakes] decisions and use interpretable models instead" (title of the publication).

1.1 Scope and goal

The objective of this thesis is to study new prototype concepts in the field of classification learning with the goal to design interpretable and high performing models. *Classification learning* is a subfield of ML and especially of supervised learning—for an introduction, we refer to the book *Deep Learning* by I. Goodfellow et al. (2016). Throughout the whole thesis, we define a *classification task* as the prediction of a class label $c \in \mathcal{C} = \{1, 2, \ldots, \#\mathcal{C}\}$ for a given input $\mathbf{x} \in \mathbb{R}^{n_x}$.[3] Moreover, we want to find a *classifier function* $\mathbf{f} : \mathbb{R}^{n_x} \to \mathbb{R}^{\#\mathcal{C}}$ such that the predicted class label $c^*(\mathbf{x})$ is equal to the desired class label of $\mathbf{x}$. In particular, the predicted class label is computed by applying the winner-takes-all rule to $\mathbf{f}$:[4]

$$c^*(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} f_c(\mathbf{x}) . \tag{1.1}$$

[3]Without loss of generality, we always define the tasks for vectorial inputs knowing that they can be extended to multidimensional inputs such as images.

[4]A letter written in boldface like $\mathbf{x}$ refers to a vector or vector function, and a letter written in normal text mode like x_i refers to the i-th element of the vector or vector function.

The classifier function $\mathbf{f}$ is parameterized by a parameter vector $\boldsymbol{\vartheta}$ of trainable parameters (also called weights) and, hence, can be written as $\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta})$.[5] Given a training dataset $\mathcal{T}$ of labeled inputs $(\mathbf{x}, c(\mathbf{x}))$, where $c(\mathbf{x}) \in \mathcal{C}$ is the correct label of $\mathbf{x}$, and a loss function (also denoted as cost function) $l(\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta}), c(\mathbf{x}))$, we adjust the classifier function $\mathbf{f}$ by an empirical risk minimization (also called averaged loss minimization) with respect to the trainable parameters:

$$E(\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta}), \mathcal{T}) = \frac{1}{\#\mathcal{T}} \sum_{(\mathbf{x},c(\mathbf{x}))\in\mathcal{T}} l(\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta}), c(\mathbf{x})) \to \min . \tag{1.2}$$

Usually, the loss function l mimics a differentiable approximation of the zero-one loss. According to Bottou and Bousquet (2008), the function E is denoted as the empirical risk or averaged loss and is iteratively optimized by a form of *Stochastic Gradient Descent* (SGD). Given a learning rate $\eta \in \mathbb{R}_{>0}$ and a training sample from $\mathcal{T}$, an update step in SGD is defined by

$$\boldsymbol{\vartheta} \leftarrow \boldsymbol{\vartheta} - \eta \nabla_{\boldsymbol{\vartheta}} l(\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta}), c(\mathbf{x})) . \tag{1.3}$$

Therefore, we train the classifier by optimizing the parameter vector $\boldsymbol{\vartheta}$. For a proper convergence, a learning rate scheduler, a mini-batch SGD (e. g., LeCun, Bottou, Orr, & Müller, 2012), or an advanced SGD version like Adam (introduced by Kingma & Ba, 2015) can be applied. The computation of the parameter updates by the gradient with respect to the trainable parameters of the loss function at the given input point is called back-propagation because we propagate the error—measured by the loss function between the predicted and the desired output—back to the trainable parameters of the classifier function.[6]

The human ability to solve problems by using knowledge of similar problems solved in the past is called case-based reasoning and is a widely used concept in ML algorithms (e. g., Aamodt & Plaza, 1994; Slade, 1991). A subfield of these algorithms is formed by prototype-based methods. In these methods, the knowledge of problems solved in the past is stored centralized and problem-specific in so-called prototypes. One of the most prominent prototype-based methods are *Learning Vector Quantization* (LVQ) approaches introduced by Kohonen (1990, 1995). These methods are

[5]Even when we speak of a vector of trainable parameters, we do not write it as one. We use more comprehensible structures such as sets that can be restructured into vectors.

[6]Usually, we compute a gradient by the relation $\nabla_{\mathbf{x}} f(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} f(\mathbf{x})$ and thus by calculating the partial derivatives. Note that this relationship only holds if f is differentiable in $\mathbf{x}$, or if all the partial derivatives exist and are continuous in $\mathbf{x}$. Unless otherwise specified, we always assume that the function is differentiable in all points that are important for the consideration. Consequently, we use the notations $\nabla_{\mathbf{x}}$ and $\frac{\partial}{\partial \mathbf{x}}$ interchangeably depending on the best readability and always call it gradient.

known to be interpretable, and according to Biehl, Hammer, and Villmann (2016), "the conceptual simplicity and interpretability facilitates efficient exchange with the domain experts and promotes [transdisciplinary] collaborations" (p. 107). This concept of interpretability is in accordance with the description of Rudin (2019) that *interpretability* is a domain-specific notation and, thus, there is no general-purpose definition. We adopt this concept and call a model interpretable if it is explicitly constrained in such a way that the learned classification function makes sense to domain experts.

Besides interpretability, there are various other quantitative measures for trained classifier functions to evaluate their performance depending on the objective. The most frequently used measure in classification learning is the *classification accuracy.* Given a test dataset of labeled inputs, this is the ratio of how many predictions are correct to all predictions made. Another commonly used set of measures is created around the topic of robustness against adversarial examples.

The thesis presents extensions of the prototype concept frequently used in classification learning. In particular, the thesis contributes

- a framework to extend prototypes to sets such as n_s-dimensional affine subspaces or orthotopes and
- a relaxation of the class-specific prototype principle.

These two extensions are exemplarily applied to LVQ methods and the resulting algorithms are compared to basic LVQ variants. Additionally, the second contribution is used to construct NN architectures. The goals of these algorithmic extensions are

- to improve the classification capabilities of LVQ methods while preserving or improving the robustness and interpretability properties compared to commonly used LVQ variants and
- to improve the interpretability properties of feedforward NNs (simply denoted as NNs) in classification tasks while preserving or improving the performances compared to architecturally equivalent ordinary NNs.

Therefore, we provide possible solutions for interpretable ML methods with high performance that could be applicable for high-stakes decisions. Even though the methods are mostly evaluated for image classification problems, they are data input generic and can be extended to other data types.[7]

[7]Unless otherwise specified, all evaluations are performed on a single NVIDIA Tesla V100 32 GB GPU using the KERAS framework (`https://www.keras.io`) with the TENSORFLOW back end (`https://www.tensorflow.org`).

1.2 Overview

This thesis consists of five chapters. After this introductory Chapter 1, we continue with Chapter 2 about the fundamentals of LVQ. Then, the following two main chapters are presented.

Chapter 3: This chapter studies the extension of the point-prototype (prototype vector) principle to set-prototypes and applies the derived point-set dissimilarity in an LVQ framework.

Chapter 4: The focus of this chapter is on the relaxation of the class-specific prototype principle, which results in the classification-by-components framework. The framework is applied in LVQ-like algorithms as well as in NNs.

These two main chapters can be read independently of each other. The final Chapter 5 presents a summary of this thesis and concluding remarks.

In all chapters, mathematical symbols are used consistently and are summarized in a list of mathematical symbols. However, sometimes it is inevitable that we redefine the meaning of a symbol for some paragraphs. In such cases, we make sure that the meaning is clear from the respective context. For instance, the symbol H generally denotes the Heaviside step function according to Equation (2.14) but describes a subgroup in Lemma 3.3. If a symbol is not in the list of mathematical symbols, then this symbol is only used in a small scope of the thesis with different meanings in different sections (e. g., the variables x and y). Additionally, acronyms are summarized in a list of acronyms.

We collected a list of publications that have been published in the context of this thesis. References to such contributions are marked by square brackets. For example, [2019c] links to Saralajew, Holdijk, Rees, Asan, and Villmann (2019) with the title "Classification-by-components: Probabilistic modeling of reasoning over a set of components." Hence, instead of making the reference by the author names and year, we use a shorthand notation for such references that might be substituted by "a contribution by colleagues and the author" during reading.

The next chapter gives a brief introduction to the fundamentals of LVQ algorithms used in this thesis. In the first section, we describe the two core concepts of LVQ algorithms: dissimilarity measures and prototypes. These two concepts are used to derive basic LVQ schemes in a subsequent section. The first realization is the heuristically motivated LVQ1 algorithm introduced by Kohonen. Even though we do not use this algorithm for further considerations in this work, it is an illustrative example to show how LVQ is motivated by intuitive principles. After that, we describe the *generalized LVQ* algorithm that is designed with the goal to have a differentiable loss function so that the algorithm can be trained by SGD. Generalized LVQ is the basis for several advanced LVQ versions—exemplarily, we introduce the *generalized matrix LVQ* algorithm. This LVQ type has many similarities with an LVQ concept described in Chapter 3. Furthermore, we have adapted ideas from LVQ into the method proposed in Chapter 4.

The main Chapter 3 presents all findings and results about the extension of the point-prototype concept to sets and the application in LVQ algorithms. Some of the results are generic and independent of the particular definition of the set-prototypes—for instance, Theorem 3.3 about the relation of point-set dissimilarities and Hausdorff distances. However, most discussions are regarding *generalized tangent LVQ*, a new LVQ variant in which the prototypes are defined as affine subspaces. For example, we show that generalized tangent LVQ is a constrained version of the local generalized matrix LVQ and that the resulting dissimilarity measure is a single-sided tangent distance. In the experimental evaluation, we show that the derived LVQ method usually outperforms other LVQ variants while being interpretable. Additionally, we prove that generalized tangent LVQ is a hypothesis margin maximizer and evaluate this experimentally by showing that the method is robust against adversarial attacks.

In the following main Chapter 4, we examine the relaxation of the class-specific prototype principle. Because the prototypes are no longer class-specific, we call them components and the resulting classification method *classification-by-components*. To motivate this framework, we start with an intuitive motivation based on Biederman's recognition-by-components theory. Inspired by this, we formulate a probabilistic reasoning model and extend it to several versions in order to handle different tasks. After that, we explain how this framework can be used to train an NN-based feature extractor jointly with the proposed classification network by SGD. In an extensive evaluation, we present how this classification principle works in an LVQ-like setting and in combination with an NN-based feature extractor. This evaluation shows that both frameworks LVQ and NN can benefit from the proposed classification scheme. Applied in an LVQ-like setting, the method boosts the accuracy results beyond the usually achievable accuracies with standard LVQ while preserving the interpretability. When trained with an NN-based feature extractor, the method can compete with the accuracies of modern NN architectures and preserves a certain degree of interpretability, as shown in some experiments.

The final Chapter 5 presents an overall summary and concluding remarks, such as open problems that require further investigation.

Chapter 2

Learning Vector Quantization

Learning vector quantization is a supervised classification algorithm introduced by Kohonen (1990, 1995) as the supervised counterpart of vector quantization. It can be considered as an NN and was originally motivated by a Hebbian learning rule. Presently, there exist various realizations of LVQ, each designed for different applications or objectives. In this chapter, we introduce the LVQ algorithms considered in this thesis. We start with an overview of the common concept. After that, we describe the basic LVQ1, the *Generalized LVQ* (GLVQ), and *Generalized Matrix LVQ* (GMLVQ). The reason why there are so many different realizations of LVQ is that the dissimilarity has to be selected with respect to the classification task. Consequently, the choice of an appropriate dissimilarity measure is considered a major topic of LVQ methods. We refer the interested reader to the articles of Biehl et al. (2016); Nova and Estévez (2014); and T. Villmann, Bohnsack, and Kaden (2017) for an overview.

2.1 General concept: Dissimilarities and prototypes

Even though there are several different realizations of LVQ algorithms, the common concept is always the same: the use of prototypes and the measuring of dissimilarities. Usually, the difference between all realizations is defined by the used dissimilarity. Therefore, we first define the concept of dissimilarities, which are related to similarities, followed by some example definitions. After that, we continue with the explanation of the prototype concept and the classification principle of the best matching prototype.

2.1.1 Dissimilarities

In everyday life, we use distance concepts to a great extent: For example, to describe how far two points are apart or to describe the time difference between two events. Generally, a mathematical distance (mathematical metric) is well-defined and an essential part of prototype-based classification. Furthermore, a distance function is not restricted to points or scalars. Instead, it can be defined for arbitrary objects.

Definition 2.1 (metric and metric space)**.** A *metric* d on a non-empty set $\mathcal{S}$ is a function

$$d : \mathcal{S} \times \mathcal{S} \longrightarrow \mathbb{R}$$

that satisfies the following conditions for all $x, y \in \mathcal{S}$:

- $d(x, y) \geq 0$ (nonnegativity);
- $d(x, y) = 0$ if and only if $x = y$ (identity of indiscernibles);
- $d(x, y) = d(y, x)$ (symmetry);
- $d(x, y) \leq d(x, z) + d(z, y)$ for all $z \in S$ (triangle inequality).

A *metric space* $(\mathcal{S}, d)$ is a set $\mathcal{S}$ with a metric d defined on $\mathcal{S}$. The function d is also denoted as *distance function* or simply *distance.*

The definition of a metric is the mathematical formalization of the intuitive human understanding of distances in the three-dimensional space. For instance, the distance between two points x and y is always positive and zero if and only if x equals y (nonnegativity and identity of indiscernibles). Furthermore, it does not matter whether the distance is measured from x to y or from y to x—the distance is the same (symmetry). The triangle inequality expresses the idea that the direct path must always be the shortest.

Definition 2.2 (translation-invariant metric)**.** Let $(\mathcal{S}, d)$ be a metric space equipped with an operation

$$+ : \mathcal{S} \times \mathcal{S} \longrightarrow \mathcal{S}$$

that forms a group $(\mathcal{S}, +)$ on $\mathcal{S}$. The metric d is called *translation-invariant* if

$$d(x, y) = d(x + z, y + z) \tag{2.1}$$

for all $x, y, z \in \mathcal{S}$.

If we continue the example from above and assume that the operation $+$ is a vector shift, then the translation invariance specifies distances that remain unchanged if the points x and y are shifted in the same direction.

According to Nebel, Kaden, Villmann, and Villmann (2017), a *dissimilarity* can be obtained by relaxing or skipping some of the fundamental axioms of a metric. However, the minimal required property is the minimum principle:

$$d(x, x) \leq d(x, y) \text{ and } d(x, x) \leq d(y, x)\,.$$

Such a dissimilarity measure is also called basic dissimilarity. In the following, we consider a dissimilarity measure as some kind of distance function with potentially relaxed metric axioms (unless otherwise stated).

Definition 2.3 (semimetric)**.** A *semimetric* is a function d like in Definition 2.1 that fulfills all the metric axioms except the identity of indiscernibles. Instead, the following relaxed version must hold:

$$x = y \Longrightarrow d\left(x, y\right) = 0.$$

This means that the elements are not always distinguishable by their dissimilarity value. Several elements can have a dissimilarity value of zero to each other without being identical.

Definition 2.4 (quasimetric)**.** A *quasimetric*[1] is a function d like in Definition 2.1 that fulfills all the metric axioms except the triangle inequality.

The Euclidean distance is named after the Greek mathematician Euclid. This distance measures the length of the line segment between two points in a real vector space. Thus, it corresponds to the natural definition of a distance in the human sense and is the most frequently used distance measure.

Definition 2.5 (Euclidean distance)**.** Let $\mathbf{x}$ and $\mathbf{y}$ be vectors of the n_x-dimensional real vector space $\mathbb{R}^{n_x}$. The *Euclidean distance* is defined by

$$d_E\left(\mathbf{x}, \mathbf{y}\right) = \sqrt{\sum_{i=1}^{n_x} \left(x_i - y_i\right)^2}$$

or in terms of vector operations by

$$d_E\left(\mathbf{x}, \mathbf{y}\right) = \sqrt{\left(\mathbf{x} - \mathbf{y}\right)^{\mathrm{T}} \left(\mathbf{x} - \mathbf{y}\right)}. \tag{2.2}$$

The Euclidean distance is a metric and, moreover, a translation-invariant metric. If the squared version $d_E^2\left(\mathbf{x}, \mathbf{y}\right)$ is considered, the Euclidean distance becomes a quasimetric (see Definition 2.4). Additionally, d_E is a special case of a Minkowski distance of order two.

A generalization of the Euclidean distance is the Mahalanobis distance. The goal of this distance is to model a more appropriate distance measure if the information about the underlying data distribution is available. For example, suppose we have collected two-dimensional feature vectors consisting of the body weight in kilograms and the stature in meters for a group of people. Furthermore, the averaged body

[1] Sometimes, this is also denoted as a usual dissimilarity.

weight is assumed to be $(83 \pm 10)\,\text{kg}$ and the stature $(1.7 \pm 0.1)\,\text{m}$.[2] Now, if we compare two people by computing the Euclidean distance between the two feature vectors, the comparison is dominated by the body weight as the weight shows much greater variations than the stature. To avoid such a behavior, the Mahalanobis distance normalizes the feature dimensions before calculating the Euclidean distance.

Definition 2.6 (Mahalanobis distance)**.** Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_x}$ be two random vectors drawn from a probability distribution with a covariance matrix $\mathbf{\Sigma}$ of full rank. The *Mahalanobis distance* is defined by

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^{\mathrm{T}} \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})}, \tag{2.3}$$

where $\mathbf{\Sigma}^{-1}$ is called precision matrix.

The full rank of $\mathbf{\Sigma}$ preserves that the covariance matrix is invertible and, hence, that the probability distribution is nondegenerate. Additionally, the precision matrix is positive definite and symmetric so that a Cholesky decomposition into the form $\mathbf{\Sigma}^{-1} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$ exists. If we rewrite Equation (2.3) by this decomposition, we get

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{\left(\mathbf{L}^{\mathrm{T}} (\mathbf{x} - \mathbf{y})\right)^{\mathrm{T}} \mathbf{L}^{\mathrm{T}} (\mathbf{x} - \mathbf{y})}. \tag{2.4}$$

This equation corresponds to the Euclidean distance between the mapped input vectors $\mathbf{L}^{\mathrm{T}}\mathbf{x}$ and $\mathbf{L}^{\mathrm{T}}\mathbf{y}$. In general, the Mahalanobis distance becomes the Euclidean distance if the covariance matrix is the identity matrix.

To generalize Equation (2.4), we drop the full rank assumption of $\mathbf{\Sigma}$ and define the so-called quadratic-dissimilarity.

Definition 2.7 (quadratic-dissimilarity)**.** Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_x}$ and $\mathbf{Q} \in \mathbb{R}^{m_x \times n_x}$. The *quadratic-dissimilarity* is defined by

$$d_Q(\mathbf{x}, \mathbf{y}) = \sqrt{\left(\mathbf{Q} (\mathbf{x} - \mathbf{y})\right)^{\mathrm{T}} \mathbf{Q} (\mathbf{x} - \mathbf{y})}, \tag{2.5}$$

where $\mathbf{Q}$ is a transformation matrix that performs a linear mapping.[3]

In general, the quadratic-dissimilarity is a metric (see Definition 2.1) if the rank of $\mathbf{Q}$ equals n_x. Otherwise, it is a semimetric (see Definition 2.3). The hyperparameter m_x determines the dimensionality of the vector space after the linear mapping. If m_x is less than or equal to n_x, then the linear mapping is performed onto a linear subspace

[2] A notation like $(12 \pm 11)\,\%$ stands for an arithmetic mean (sample mean) of $12\,\%$ with a corrected sample standard deviation of $11\,\%$.

[3] Note that we make a strict distinction between a transformation (matrix) and a projection (matrix) in the mathematical sense.

of $\mathbb{R}^{n_x}$ and to the higher-dimensional vector space $\mathbb{R}^{m_x}$ otherwise. Similar to the Mahalanobis distance, the quadratic-dissimilarity measures the Euclidean distance on the transformed input vectors and can be written as

$$d_Q(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^{\mathrm{T}} \mathbf{\Lambda} (\mathbf{x} - \mathbf{y})}, \tag{2.6}$$

where $\mathbf{\Lambda}$ equals $\mathbf{Q}^{\mathrm{T}}\mathbf{Q}$.

All LVQ methods use some kind of dissimilarity measure, as we will see in the following sections. However, the quadratic-dissimilarity is the most frequently used dissimilarity measure in LVQ methods (e. g., Bunte et al., 2012; Schneider et al., 2010) and ML algorithms in general (e. g., Globerson & Roweis, 2006; Weinberger & Saul, 2009; Xing, Ng, Jordan, & Russell, 2003).

2.1.2 Prototypes and the best matching prototype principle

Besides a formal definition, the concept behind a prototype tries to capture the intuitive human understanding of prototypical things. This is "the premise that the prototypes of class [c] should consist of points that are close to many training points of class [c] and are far from training points of other classes" (Bien & Tibshirani, 2011, p. 2404) and, moreover, the idea to describe as many objects of a class as possible by unique objects that best describe the whole class. The first part of this statement already uses the wording *close* and *far*, and thus prototypes are always related to a respective dissimilarity measure to describe how similar or dissimilar objects are.

Definition 2.8 (prototype)**.** Given a classification task, a *prototype*[4] (vector) $\mathbf{w}_k$ is an element of the data space $\mathbb{R}^{n_x}$ equipped with a fixed class label $c(\mathbf{w}_k) \in \mathcal{C}$. All prototypes are collected into a set

$$\mathcal{W} = \{\mathbf{w}_k \in \mathbb{R}^{n_x} \mid k = 1, 2, \ldots, \#\mathcal{W}\}, \tag{2.7}$$

where $\#\mathcal{W}$ is the overall number of prototypes. This set contains at least one prototype per class.

Depending on the ML approach, the definition could be equipped with more conditions—for example, the prototypes have to be elements of the training dataset, specific interpretability constraints, and so on.

To measure the fit of the prototypes $\mathbf{w}_k$ regarding a given data point $\mathbf{x}$, we use a dissimilarity measure d. Let $\mathbf{d}(\mathbf{x})$ be the *prototype response vector*, that is, the vector with the dissimilarities to each prototype with respect to $\mathbf{x}$:

$$\mathbf{d}(\mathbf{x}) = (d(\mathbf{x}, \mathbf{w}_1), d(\mathbf{x}, \mathbf{w}_2), \ldots, d(\mathbf{x}, \mathbf{w}_{\#\mathcal{W}}))^{\mathrm{T}}. \tag{2.8}$$

[4]Also denoted as reference vector, codebook vector, weight vector, or point-prototype.

Based on $\mathbf{d}(\mathbf{x})$, we derive the classifier function $\mathbf{f}(\mathbf{x})$ by calculating and negating the smallest dissimilarity value for each class:

$$\mathbf{f}(\mathbf{x}) = -\begin{pmatrix} \min\{d_k(\mathbf{x}) \mid c(\mathbf{w}_k) = 1\} \\ \min\{d_k(\mathbf{x}) \mid c(\mathbf{w}_k) = 2\} \\ \vdots \\ \min\{d_k(\mathbf{x}) \mid c(\mathbf{w}_k) = \#\mathcal{C}\} \end{pmatrix}, \tag{2.9}$$

where $f_c(\mathbf{x})$ is the negative value of the smallest dissimilarity of $\mathbf{x}$ to a prototype of class c. Note that we have to negate the dissimilarities to fit in the definition of the winner-takes-all rule of Equation (1.1). The computation of the smallest dissimilarity within a class is also called *competition* as the prototypes compete with each other to be the closest prototype.

The closest prototype $\mathbf{w}^*$ regarding the input $\mathbf{x}$ is determined by

$$\mathbf{w}^*(\mathbf{x}) = \underset{\mathbf{w}_k \in \mathcal{W}}{\arg\min}\, d(\mathbf{x}, \mathbf{w}_k) \tag{2.10}$$

and is called the *best matching prototype* with respect to $\mathbf{x}$. This prototype determines by its class label $c(\mathbf{w}^*)$ the predicted class $c^*(\mathbf{x})$ of $\mathbf{x}$. In the context of prototype-based learning, this strategy is denoted as the *Best Matching Prototype Principle* (BMPP) because only $\mathbf{w}^*$ is used for the class determination. Furthermore, the BMPP represents a winner-takes-all rule.

Based on a training dataset $\mathcal{T}$, the goal of prototype-based learning is to find a proper set of prototypes. Therefore, the trainable parameters $\boldsymbol{\vartheta}$ of the classifier function $\mathbf{f}$ are the prototypes in $\mathcal{W}$.[5] According to Bien and Tibshirani (2011), we search for a set of prototypes such that

- the prototypes of class c are the closest points for as many training data points of class c as possible,
- the prototypes of class c are the closest points for as few training data points of a class other than c as possible, and
- the number of prototypes $\#\mathcal{W}$ is as small as possible.

In LVQ, this is realized by an iterative adaptation scheme of the prototypes in $\mathcal{W}$ and a predefined number of prototypes $\#\mathcal{W}$. However, depending on the specific prototype-based method, the set of prototypes $\mathcal{W}$ could also be determined by a different mathematical scheme, for instance, a convex optimization problem (e. g., Bien & Tibshirani, 2011).

[5]As we will see in Section 2.2.3, there can be more trainable parameters.

2.2 Realizations

LVQ algorithms are considered as interpretable ML approches. This interpretability assumption is based on the idea of having an understanding of the used dissimilarity measure and that prototypes can be interpreted by a human expert. For example, if the learning task is handwritten digit classification, we interpret the learned model by visualizing the prototypes as images and trying to understand them. More specifically, we hope that they resemble class-specific objects such as the usual writing styles of a particular digit.

In Hammer, Nebel, Riedel, and Villmann (2014), it is discussed that newer LVQ methods can be difficult to interpret. Roughly described, this is the case if all prototype vectors lie outside the class distributions, that is, if they are not close to a data input. However, the authors show that this effect can be controlled by an appropriate regularization.

Another point related to interpretability is that the data space could only be a subset of $\mathbb{R}^{n_x}$—for instance, images are frequently defined in $[0,1]^{n_x}$. In this case, we have to prevent the violation of the respective data constraints by the prototype updates. This can be achieved by

- a proper coding of the prototypes—for example, the coding that Carlini and Wagner (2017) used for the attacks to avoid box-constraints—or
- a projected update rule, also known as projected gradient descent learning (e. g., Suárez, García, & Herrera, 2018).

If we do not restrict the prototypes to the respective space, they could again be difficult to interpret.

Regardless of this, some understanding can be obtained by analyzing the dissimilarity distribution over each class or the entire dataset. This can provide information about what is far and close in the classification model and how confident the model is with respect to a classification decision. Additionally, such an analysis may be useful to identify the problems described by Hammer et al. (2014).

The next sections describe several realizations of the LVQ method. First, we describe Kohonen's LVQ algorithms as heuristically motivated classification methods. After that, we describe the GLVQ algorithm, which is a version of Kohonen's LVQ algorithms with a differentiable loss function so that it can be trained by SGD. Finally, we present an extended version of GLVQ called GMLVQ, where the prototypes are learned in parallel with a quadratic-dissimilarity measure, see Equation (2.5). Although there may be difficulties in interpreting these methods, we will consider them interpretable since we know that this is possible to a certain extent.

Algorithm 1 LVQ1 with learning rate η and a maximum number of steps N.

```
1: procedure LVQ1($\mathcal{T}, d, \mathcal{W}, \eta, N$)
2:     $\mathcal{W} \leftarrow$ initialize the set of prototypes
3:     $i \leftarrow 0$
4:     while $i < N$ do
5:         $i \leftarrow i + 1$
6:         $\mathbf{x}, c(\mathbf{x}) \leftarrow$ randomly pick a training sample from $\mathcal{T}$
7:         $\mathbf{w}^* \leftarrow \mathbf{w}^*(\mathbf{x})$          ▷ evaluate the closest protoype, see Equation (2.10)
8:         if $c(\mathbf{w}^*) = c(\mathbf{x})$ then
9:             $s \leftarrow -1$                    ▷ prototype of correct class: attract
10:        else
11:            $s \leftarrow 1$                     ▷ prototype of incorrect class: repel
12:        $\triangle\mathbf{w}^* \leftarrow s(\mathbf{x} - \mathbf{w}^*)$          ▷ evaluate the vector shift
13:        $\mathbf{w}^* \leftarrow \mathbf{w}^* - \eta\triangle\mathbf{w}^*$        ▷ update of the prototype in $\mathcal{W}$
14:    return $\mathcal{W}$                        ▷ return the trained prototypes
```

2.2.1 Kohonen's learning vector quantization algorithms

The original LVQ models proposed by Kohonen (1990, 1995) are heuristic approaches motivated by Bayes decision theory and vector quantization. Heuristic means that we do not directly optimize a loss function like Equation (1.2). All basic LVQ approaches distribute the prototypes via iteratively applied small vector shifts proportional to $\mathbf{x} - \mathbf{w}$ to improve the classifier function, see Equation (2.9).[6] The precise realization of the prototype selection and shifting scheme determines the LVQ approach. In general, the shifts are motivated by a Hebbian learning principle as *attraction* and *repulsion* forces on the prototypes. Even if the principle of attraction and repulsion forces is no longer explicitly modeled in modern LVQ variants, it can still be identified and is a typical property of these methods.

The method LVQ1, see the pseudocode in Algorithm 1, is the simplest LVQ scheme and consists of the following steps: We define the set of prototypes $\mathcal{W}$ such that each class is represented by at least one prototype. After that, the prototypes are initialized with a suitable scheme—for instance, as random vectors, as randomly selected data samples from the respective class, as k-means over the data samples of the respective class, and so on. Then, we iterate over the following two steps:

1. Randomly pick a training sample $(\mathbf{x}, c(\mathbf{x}))$ from the training dataset $\mathcal{T}$ and determine the closest prototype $\mathbf{w}^*$ using Equation (2.10).

[6]We omit the prototype index k if the statements are independent of a specific prototype k.

2. If the class label $c(\mathbf{w}^*)$ of the closest prototype is equal to the class label $c(\mathbf{x})$ of the input, we push the prototype $\mathbf{w}^*$ a little bit towards $\mathbf{x}$ (attraction) and pull it slightly away (repulsion) otherwise.

The magnitude of the applied shift is controlled by a learning rate $\eta \in \mathbb{R}_{>0}$. After the model is trained, we use the BMPP to assign class labels to arbitrary data points.

Usually, LVQ1 uses the Euclidean distance d_E according to Equation (2.2). The gradient of d_E with respect to a prototype is

$$\nabla_{\mathbf{w}} d_E(\mathbf{x}, \mathbf{w}) = -\frac{1}{d_E(\mathbf{x}, \mathbf{w})}(\mathbf{x} - \mathbf{w}) \tag{2.11}$$

if $d_E(\mathbf{x}, \mathbf{w}) \neq 0$. By considering step 12 in the Algorithm 1, we can conclude that the differentiation of the Euclidean distance with respect to a prototype yields a scaled version of the vector shift. Together with the update rule of step 13, the prototype update of LVQ1 is similar to Equation (1.3), the update rule of SGD. Nevertheless, LVQ1 cannot be interpreted as a form of SGD since the variable s, which defines whether a prototype is attracted or repelled, is not differentiable with respect to the prototypes and, moreover, has only trivial gradients at points where the gradient exists. The gradient at a given point is called trivial if it is the zero vector.

The characteristic property of LVQ1 is the update of exactly one prototype per iteration. This idea was extended in LVQ2.1 and LVQ3 to update up to two prototypes per iteration. LVQ2.1 determines the two closest prototypes and attracts and repels them simultaneously if one of the prototypes belongs to the correct class and the other to the incorrect class. Like in LVQ1, the correct prototype is attracted and the incorrect one is repelled. Additionally, this update is only performed if the given training point falls within a certain midplane window between the two prototypes. If both prototypes are from the correct or incorrect class, no update is performed.

2.2.2 Generalized learning vector quantization

In 1996, Sato and Yamada proposed the GLVQ algorithm. The goal of GLVQ is to provide

- a generalization of Kohonen's LVQ algorithms,
- an LVQ algorithm that can be trained by an empirical risk minimization according to Equation (1.2) using SGD, and
- an LVQ learning rule that satisfies the convergence condition.[7]

[7]Given a training sample and the corresponding closest prototype of the correct and incorrect class, the convergence condition is satisfied if the attraction force on the prototype of the correct class is greater than the repulsion force on the prototype of the incorrect class.

Sato and Yamada have realized this by developing a differentiable loss function called GLVQ loss.

Given a classifier function $\mathbf{f}(\mathbf{x})$ as defined in Equation (2.9) and a training sample $(\mathbf{x}, c(\mathbf{x}))$, we calculate the smallest dissimilarity to a prototype of the correct class $c(\mathbf{x})$ of $\mathbf{x}$ by

$$d^{+}(\mathbf{x}) = -f_{c(\mathbf{x})}(\mathbf{x}).$$

Furthermore, we compute the smallest dissimilarity to a prototype of a class different than $c(\mathbf{x})$ of $\mathbf{x}$ by

$$d^{-}(\mathbf{x}) = \min\{-f_c(\mathbf{x}) \mid c \neq c(\mathbf{x})\}.$$

Using these two dissimilarities, we define the *relative distance difference*[8] as

$$\mu(\mathbf{x}) = \frac{d^{+}(\mathbf{x}) - d^{-}(\mathbf{x})}{d^{+}(\mathbf{x}) + d^{-}(\mathbf{x})} \in [-1, 1]. \tag{2.12}$$

The function returns negative values if and only if $\mathbf{x}$ is correctly classified, otherwise it returns positive values.[9] This is similar to the behavior of the variable s in Algorithm 1, which is minus one for correct classifications and one otherwise. But in contrast to s, the function μ is differentiable with respect to $\mathbf{w}$ and generally has nontrivial gradients.

The relative distance difference can be used to compute the empirical classification error by

$$\mathit{error}(\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta}), \mathcal{T}) = \frac{1}{\#\mathcal{T}} \sum_{(\mathbf{x}, c(\mathbf{x})) \in \mathcal{T}} H(\mu(\mathbf{x})) \tag{2.13}$$

with

$$H(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{2.14}$$

being the Heaviside step function that realizes the zero-one loss function. Since $H(x)$ is not differentiable, Sato and Yamada replaced the Heaviside step function with a monotonically increasing, differentiable squashing function $\phi : [-1, 1] \to \mathbb{R}$. Finally, this results in the *GLVQ loss function*:

$$l(\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta}), c(\mathbf{x})) = \phi(\mu(\mathbf{x})). \tag{2.15}$$

This loss function is used in the empirical risk E to approximate the empirical classification error, see Equation (1.2) and Equation (2.13), respectively. Common realizations of ϕ are the identity function (i. e., $\mathrm{id}(x) = x$) or a sigmoid function like the

[8]Even if the relative distance difference is calculated with dissimilarities, we call it relative *distance* difference to keep the original formulation of Sato and Yamada.

[9]Due to this property, the function μ is sometimes called the LVQ classifier function.

logistic function given by

$$\operatorname{sigmoid}_{\sigma}(x) = \frac{1}{1+\exp\left(-\frac{x}{\sigma}\right)}, \tag{2.16}$$

where the slope parameter σ has to be greater than zero. If σ becomes arbitrarily small, the function $\operatorname{sigmoid}_{\sigma}(x)$ converges to the Heaviside step function $H(x)$. We refer to T. Villmann, Ravichandran, Villmann, Nebel, and Kaden (2019) for an extensive study about other possible squashing functions.

To adapt the classifier function, we optimize the obtained empirical risk function by SGD. Accordingly, we randomly pick a training sample $(\mathbf{x}, c(\mathbf{x})) \in \mathcal{T}$ and perform a gradient descent step regarding $l(\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta}), c(\mathbf{x}))$. Due to $\mu(\mathbf{x})$, the loss function only depends on two prototypes, called $\mathbf{w}^{+}$ and $\mathbf{w}^{-}$. The prototype $\mathbf{w}^{+}$ is the closest prototype of the correct class associated with $d^{+}(\mathbf{x})$ and $\mathbf{w}^{-}$ is the closest prototype of an incorrect class associated with $d^{-}(\mathbf{x})$. Hence, the derivatives of the GLVQ loss with respect to the elements of $\boldsymbol{\vartheta}$ are nonzero only for the parameters of the two prototypes $\mathbf{w}^{\pm}$ so that we only have to calculate the gradient with respect to $\mathbf{w}^{\pm}$. The gradient of the loss function given $(\mathbf{x}, c(\mathbf{x}))$ with respect to $\mathbf{w}^{\pm}$ is[10]

$$\frac{\partial}{\partial \mathbf{w}^{\pm}} \phi(\mu(\mathbf{x})) = \pm 2\phi'(\mu(\mathbf{x})) \frac{d^{\mp}(\mathbf{x})}{\left(d^{+}(\mathbf{x}) + d^{-}(\mathbf{x})\right)^{2}} \frac{\partial d\left(\mathbf{x}, \mathbf{w}^{\pm}\right)}{\partial \mathbf{w}^{\pm}}. \tag{2.17}$$

Thus, the updates of the SGD are proportional to

$$\triangle \mathbf{w}^{\pm} = \pm \frac{\partial d\left(\mathbf{x}, \mathbf{w}^{\pm}\right)}{\partial \mathbf{w}^{\pm}}, \tag{2.18}$$

the so-called *signed gradients* of the respective dissimilarity measure. The terms

$$\xi^{\pm} = 2\phi'(\mu(\mathbf{x})) \frac{d^{\mp}(\mathbf{x})}{\left(d^{+}(\mathbf{x}) + d^{-}(\mathbf{x})\right)^{2}} \geq 0 \tag{2.19}$$

act as scaling factors. In summary, Equation (2.17) can be written as

$$\frac{\partial}{\partial \mathbf{w}^{\pm}} \phi(\mu(\mathbf{x})) = \xi^{\pm} \triangle \mathbf{w}^{\pm}$$

and the learning rule of the prototypes in a SGD according to Equation (1.3) is

$$\mathbf{w}^{\pm} \leftarrow \mathbf{w}^{\pm} - \eta \xi^{\pm} \triangle \mathbf{w}^{\pm}. \tag{2.20}$$

[10]The used notation must be read as follows: The expression $\mathbf{z}^{\pm} = \mathbf{x}^{\pm} \mp \frac{a^{\mp}}{a^{+}+a^{-}}$ combines two alternatives in one equation. These are $\mathbf{z}^{+} = \mathbf{x}^{+} - \frac{a^{-}}{a^{+}+a^{-}}$ and $\mathbf{z}^{-} = \mathbf{x}^{-} + \frac{a^{+}}{a^{+}+a^{-}}$. That is, each time we have an alternative due to $\pm$ or $\mp$, we consistently choose the expression of the upper or lower sign.

This update rule is similar to the one in Algorithm 1 with the difference that we update both prototypes at the same time—like in LVQ2.1. Accordingly, the prototype $\mathbf{w}^+$ is attracted towards the training sample and the prototype $\mathbf{w}^-$ is repelled. In contrast to Kohonen's LVQ algorithms, the derived scaling factors $\xi^\pm$ ensure that the attraction forces are stronger than the repulsion forces. Consequently, the convergence condition is fulfilled.

In the original GLVQ experiments performed by Sato and Yamada, the dissimilarity d was defined as the squared Euclidean distance d_E^2. In this case, the signed dissimilarity gradients, see Equation (2.18), are

$$\triangle \mathbf{w}^\pm = \mp 2 \left(\mathbf{x} - \mathbf{w}^\pm\right).$$

With this result in Equation (2.20), we get the GLVQ learning rule regarding d_E^2 as

$$\mathbf{w}^\pm \leftarrow \mathbf{w}^\pm \pm 2\eta\xi^\pm \left(\mathbf{x} - \mathbf{w}^\pm\right).$$

This update scheme represents the initially motivated LVQ attraction and repulsion forces on the prototypes $\mathbf{w}^\pm$.

2.2.3 Generalized matrix learning vector quantization

Schneider, Biehl, and Hammer (2009) proposed the GMLVQ classifier as an extension of the *generalized relevance LVQ* of Hammer and Villmann (2002). Both methods are versions of GLVQ with adaptable dissimilarity measures. The goal is to use dissimilarity measures that improve the class discrimination in the presence of different covariances or variances or both in the data. Considering the usually used Euclidean distance, we observe that implicitly equal importance of the input dimensions of $\mathbb{R}^{n_x}$ is assumed—as described in the example before the Definition 2.6 about the motivation of the Mahalanobis distance compared to the Euclidean distance. Obviously, this assumption might not be beneficial for class discrimination in general.

To overcome this problem, GMLVQ uses the squared quadratic-dissimilarity d_Q^2 according to Definition 2.7 and extends the learning rules of GLVQ by adaptation rules for $\mathbf{Q} \in \mathbb{R}^{m_x \times n_x}$. By adjusting $\mathbf{Q}$, the method tries to learn a proper transformation of the data points before the dissimilarity is measured by the Euclidean distance. This extends the parameter vector $\boldsymbol{\vartheta}$ of GLVQ by $\mathbf{Q}$. Apart from this change, we retain the general learning framework of GLVQ, which includes the GLVQ loss and the derived learning rules of the prototypes.

Equivalent to GLVQ, the GLVQ loss function regarding d_Q^2 has only nontrivial gradients for the prototypes $\mathbf{w}^\pm$ and, consequently, we only have to compute the gradients with respect to these prototypes. The learning rule for the prototypes in a basic SGD is identical to Equation (2.20) with the gradient

$$\nabla_{\mathbf{w}} d_Q^2(\mathbf{x}, \mathbf{w}) = -2\mathbf{Q}^{\mathrm{T}}\mathbf{Q}(\mathbf{x} - \mathbf{w}) \tag{2.21}$$

for $\frac{\partial d}{\partial \mathbf{w}^{\pm}}$ in the signed gradient $\triangle \mathbf{w}^{\pm}$, see Equation (2.18).

In the same way, we derive the update formula for the matrix $\mathbf{Q}$. The gradient of the GLVQ loss with respect to the matrix is

$$\nabla_{\mathbf{Q}} \phi(\mu(\mathbf{x})) = \xi^{+} \cdot \nabla_{\mathbf{Q}} d_Q^2(\mathbf{x}, \mathbf{w}^{+}) - \xi^{-} \cdot \nabla_{\mathbf{Q}} d_Q^2(\mathbf{x}, \mathbf{w}^{-}),$$

where $\xi^{\pm}$ are the same scaling factors as in GLVQ. Moreover, the gradient of d_Q^2 with respect to $\mathbf{Q}$ yields

$$\nabla_{\mathbf{Q}} d_Q^2(\mathbf{x}, \mathbf{w}) = 2\mathbf{Q}(\mathbf{x} - \mathbf{w})(\mathbf{x} - \mathbf{w})^{\mathrm{T}} \tag{2.22}$$

and a basic SGD rule according to Equation (1.3) to learn $\mathbf{Q}$ is

$$\mathbf{Q} \leftarrow \mathbf{Q} - \eta\left(\xi^{+} \cdot \nabla_{\mathbf{Q}} d_Q^2(\mathbf{x}, \mathbf{w}^{+}) - \xi^{-} \cdot \nabla_{\mathbf{Q}} d_Q^2(\mathbf{x}, \mathbf{w}^{-})\right). \tag{2.23}$$

Note that the matrix receives an update from $\mathbf{w}^{+}$ and $\mathbf{w}^{-}$ simultaneously. As a result, the matrix $\mathbf{Q}$ is adjusted to become more discriminative with respect to the class of $\mathbf{w}^{+}$ and the class of $\mathbf{w}^{-}$.

The matrix $\mathbf{Q}$ and the implied transformation have several interpretations. One is that we perform a linear mapping from $\mathbb{R}^{n_x}$ to $\mathbb{R}^{m_x}$ via

$$\hat{\mathbf{x}} = \mathbf{Q}\mathbf{x}$$

before we measure the dissimilarity by the Euclidean distance. Particularly, the following relation holds:

$$d_E(\hat{\mathbf{x}}, \hat{\mathbf{w}}) = d_Q(\mathbf{x}, \mathbf{w}).$$

Depending on the precise setting of the hyperparameter m_x, the dissimilarity performs a linear mapping into a higher-dimensional ($m_x > n_x$) or lower-dimensional ($m_x < n_x$) space compared to the input space $\mathbb{R}^{n_x}$ or maps into the input space. Since we learn this mapping while optimizing the GLVQ loss, the goal of this mapping is to improve the class separability and, therefore, implicitly the classification accuracy. The properties of such mappings, especially into the two- and three-dimensional space, have been studied extensively—we refer to the articles of Bunte, Hammer, Wismüller, and Biehl (2010); and Bunte et al. (2012).

Another interpretation is inspired by the connection of d_Q with the Mahalanobis distance and the alternative formulation of d_Q, see Equation (2.3) and Equation (2.6), respectively. If we constrain $\mathbf{Q}$ to a Frobenius norm of one (i. e., $\|\mathbf{Q}\|_F = 1$) during training, the matrix $\mathbf{\Lambda} = \mathbf{Q}^{\mathrm{T}}\mathbf{Q}$ is considered as a classification correlation matrix (e. g., T. Villmann, Bohnsack, & Kaden, 2017).[11] Now, the sum of the diagonal elements Λ_{ii} equals one and the values Λ_{ii} are interpreted as relevance factors reflecting the importance of each input dimension for class discrimination. The off-diagonal elements Λ_{ij} indicate the correlation between the different input dimensions to support the discrimination of the classes. This information can be used to perform a pruning (reduction) of input dimensions after the training or to gain an understanding of the learned classifier function.

A natural extension of the GMLVQ algorithm proposed above is to use for each prototype $\mathbf{w}_k$ an individual (local) transformation matrix $\mathbf{Q}_k$ in the dissimilarity d_Q. This leads to a prototype-index specific quadratic-dissimilarity and the derived method is called *local-GMLVQ*. We denote the corresponding matrix of $\mathbf{w}^{\pm}$ by $\mathbf{Q}^{\pm}$ and mark the dependency of d_Q from $\mathbf{Q}_k$ by respective subscripts or superscripts at Q—for instance, d_{Q_k} refers to the dissimilarity of $\mathbf{w}_k$ with $\mathbf{Q}_k$. The update formulas for the matrices $\mathbf{Q}^{\pm}$ are given by splitting the update rule of Equation (2.23) into the corresponding parts:

$$\mathbf{Q}^{\pm} \leftarrow \mathbf{Q}^{\pm} \mp \eta \xi^{\pm} \frac{\partial d^2_{Q^{\pm}}\left(\mathbf{x}, \mathbf{w}^{\pm}\right)}{\partial \mathbf{Q}^{\pm}}. \tag{2.24}$$

Thus, the trainable parameter vector $\boldsymbol{\vartheta}$ of GLVQ is now extended by all the prototype specific transformation matrices $\mathbf{Q}_k$. The previously discussed interpretation techniques of GMLVQ can be transferred to the local version.

[11]Besides better interpretability, the constraint $\|\mathbf{Q}\|_F = 1$ is often applied to avoid numerical instabilities and degenerations (e. g., Schneider et al., 2009). However, in order to guarantee this property, the constraint to a sufficiently large constant $\omega > 0$ is sufficient (i. e., $\|\mathbf{Q}\|_F = \omega$). It should also be noted that $\|\mathbf{Q}\|_F$ equals $\sqrt{\mathrm{trace}\left(\mathbf{\Lambda}\right)}$.

This chapter is mainly based on the following joint work:

[2016b]	Saralajew and Villmann (2016). Adaptive tangent distances in generalized learning vector quantization for transformation and distortion invariant classification learning.
[2016c]	Saralajew, Nebel, and Villmann (2016). Adaptive Hausdorff distances and tangent distance adaptation for transformation invariant classification learning.
[2017a]	Saralajew and Villmann (2017). Restricted tangent distances for local data dissimilarities.
[2017c]	Saralajew and Villmann (2017). Transfer learning in classification based on manifold models and its relation to tangent metric learning.
[2019b]	Saralajew, Holdijk, Rees, and Villmann (2019). Robustness of generalized learning vector quantization models against adversarial attacks.

Chapter 3

Generalized Tangent Learning Vector Quantization

Generally, classification learning from noisy or corrupted data is one of the major topics in ML as it leads to reduced classification performance if not considered in the model. In principle, LVQ is able to adequately process noisy data because the prototypes are weighted averages of the training samples (e. g., Biehl, Hammer, Schleif, Schneider, & Villmann, 2015). However, as an alternative to noise, the data may also show systematic variations such as naturally occurring transformations. These data variations also contribute to reduced classification performance if the model cannot process these variations properly.

The mathematical properties of systematic variations and how they are treated in ML differ significantly from data noise. If these variations are known beforehand, respective data preprocessing methods can be applied in advance to remove or decrease their impact on the classification model. For instance, transformation invariant feature extraction methods such as the scale-invariant feature transform of Lowe (2004) are well-known techniques for image processing. If such a preprocessing is not possible, the data variations might be directly approximated by continuous transformations of the data inside the ML model. For example, Simard, LeCun, and Denker (1993) defined a set of transformations (consisting of line thickness variations, rotations, shifts, etc.) for handwritten digits to improve the accuracy of a classifier. However, such a definition might not be easy in general—for instance: What is a suitable transformation in image space to model different facial expressions for face recognition? To overcome this difficulty, a common scheme is to define a parameterized transformation model. The parameters of the transformation model are estimated on a training dataset before or parallel to the training of the classification model. Today, this framework is one of the main components of modern NNs and is generally referred to as a trainable feature extractor.

Currently, LVQ approaches try to model systematic variations by distributing several prototypes per class or by learning a dissimilarity measure such that the dataset is adequately represented. In contrast, we propose to relax the concept of prototype vectors and introduce set-prototypes to model a high amount of data variations directly by a single prototype. Based on this idea, we show how to derive respective dissimilarity measures and how to train a resulting GLVQ network. The derived dissimilarity measures are closely related to the concept of tangent distances proposed by Simard et al. (1993) as they learn a set of tangent approximations around each prototype vector to improve the class discrimination. Therefore, we call the obtained GLVQ scheme *Generalized Tangent Learning Vector Quantization* (GTLVQ). According to the objectives of the thesis, we show that GTLVQ

- consistently trains to higher accuracies than standard LVQ methods,
- preserves the interpretability, and
- is a hypothesis margin maximizer and, therefore, very robust.

In particular, we present how the hypothesis margin maximization property is related to the smallest adversarial distance required to generate adversarial examples. Based on numerical evaluations, we show that this property can be used to train adversarially robust LVQ models that can compete with current state-of-the-art NNs in terms of adversarial robustness. Additionally, we show that GTLVQ

- is a constrained version of the local-GMLVQ approach described in Section 2.2.3 with limited rank (i. e., with $m_x < n_x$),
- is related to a Hausdorff distance regarding the used dissimilarity, and
- can be derived as a linear approximator of data manifolds.

In the next section, we describe how the GTLVQ method is motivated by previous work on tangent distances. Subsequently, we define the framework of set-prototypes and derive GTLVQ as a nontrivial realization of LVQ with set-prototypes. Thereby, the set-prototypes are n_s-dimensional affine subspaces. Moreover, we define a restricted version of GTLVQ in which the set-prototypes are orthotopes. This section is followed by a study about the relation of the derived dissimilarity measures to

Hausdorff distances. By taking advantage of this relation, we show that the derived dissimilarity measures satisfy certain metric axioms. Then we demonstrate that GMLVQ methods and GTLVQ are related and that local-GMLVQ is equivalent to GTLVQ if trained with a sufficiently large regularization parameter. After that, we present a numerical evaluation of GTLVQ regarding accuracy and interpretability on several toy and benchmark datasets.

A generally important finding about LVQ methods is that they maximize the hypothesis margin. This topic is covered by the subsequent section. Additionally, we show how the hypothesis margin is related to adversarial perturbations and present a numerical evaluation. Finally, we discuss related work and give a summary of this chapter. The created software and the scripts of the experiments are available at `https://github.com/saralajew/thesis_GTLVQ_experiments` as a Keras package. The robustness evaluations are made available at `https://github.com/LarsHoldijk/robust_LVQ_models` as pretrained Tensorflow graphs and as part of the Foolbox[1] model zoo.

Research chronology

The research about the application of tangent distances in LVQ was initiated by the statement of Crammer, Gilad-Bachrach, Navot, and Tishby (2003) to apply tangent distances in LVQ algorithms. After formulating the GTLVQ learning rules in [2016b], we studied the refinement regarding restricted tangent distances and formulated a first theorem about the relation to Hausdorff distances in [2016c]. Motivated by findings in the field of transfer learning, we focused in [2017c] on relations of GTLVQ to transfer learning and manifold learning. In [2017a] and [2017b], we presented the results about restricted tangent distances. This framework triggered the general considerations about set-prototypes and point-set dissimilarities and has led to the general formulation of the theorems presented in this thesis.

At the same time, we began to investigate the relationship of LVQ methods to NNs [2017d]. A first attempt to merge LVQ and NN methods was presented in [2018d]. Motivated by the results obtained with such models, we started to analyze LVQ methods regarding their adversarial robustness. In [2019b], we analyzed the different LVQ methods regarding adversarial robustness and explained why LVQ methods are provably robust classifiers. The results of this publication are extended and refined in this thesis.

[1] `https://foolbox.readthedocs.io`

Author contributions

David Nebel: He conducted the experiments for the methods other than GTLVQ on the SPIRAL dataset in [2016c]. Additionally, David contributed during the writing of the respective publication.

Lars Holdijk: The idea to study the robustness properties of LVQ methods was initiated by him after he participated in the conference competition Adversarial Vision Challenge 2018[2] of the conference Neural Information Processing Systems where he applied the technologies presented in [2018d]. He evaluated the adversarial robustness of all the LVQ methods and also trained and evaluated the baseline NNs. Additionally, he created the online available models. The analysis of the robustness results and their implications were discussed with Lars. Finally, he wrote a substantial part of the corresponding publication.

Maike Rees: Maike contributed substantially to writing, designing, and structuring the respective publication. Moreover, she implemented the early evaluation pipelines for the Adversarial Vision Challenge 2018 and contributed to the discussion of the results of the robustness evaluation in [2019b].

Thomas Villmann: Based on the first application of tangent distances in LVQ, Thomas helped to formalize the framework. Furthermore, he motivated further research about restricted tangent distances and their relation to Hausdorff distances and GMLVQ. Additionally, he initiated the considerations about the relationship of GTLVQ to transfer learning and discussions with him resulted in the ideas about the theory of Crammer et al. (2003). Finally, he contributed to all publications with writing, proofreading, and a lot of advice.

3.1 Motivation

Tangent distances were introduced by Simard et al. (1993) to have a more suitable distance measure than the Euclidean distance in the presence of data variations for distance-based classification algorithms. As an example, Simard et al. applied the tangent distance in a k-nearest neighbors algorithm. In their final discussion, they emphasized that "many distance-based classification schemes could be used in conjunction with [the] tangent distance, among them LVQ" (p. 58).

[2]Brendel, W. (2018, November 9). Results of the NIPS Adversarial Vision Challenge 2018. Retrieved from `https://medium.com/bethgelab/results-of-the-nips-adversarial-vision-challenge-2018-e1e21b690149`

Later, their work was refined by Hastie, Simard, and Säckinger (1995) and they stated that “the tangent centroid or subspace models can be used to seed LVQ algorithms (Kohonen 1989), but so far we have not much experience with them” (p. 1004). Moreover, in this work, they already introduced the tangent distance as an approach “for generalizing the concept of a mean or centroid for a set of images, taking into account the tangent families” (p. 1001) and, hence, they extended the point-prototype concept to affine subspaces.

In the important work of Crammer et al. (2003) about the margin maximization properties of LVQ, the conclusion is that an interesting extension of the margin maximization theory is to use a different distance measure than the Euclidean norm. However, they expect complicated gradient equations for the update rules but suspect that this modification could significantly improve the results. In particular, they emphasize that an “interesting distance measure is the [t]angent [d]istance” (p. 486).

Interestingly, the investigation of tangent distances was not realized in LVQ algorithms, although this has been proposed in several publications for decades. The GTLVQ approach presented here closes this gap and learns prototype models with prototypes that are sets. These set-prototypes are able to model local variations of the data points so that the resulting classifier becomes locally invariant. This principle is advantageous if a dataset is affected by local transformations—for instance, in handwritten digit recognition where the digits can be slightly rotated or shifted.

From another point of view, GTLVQ increases the capacity of an LVQ classifier to capture information about the dataset. In basic LVQ approaches, the information storage is restricted to points, the prototypes, and parameters of the dissimilarity measure in use. This limits the model capacity of basic LVQ methods, and the only solution for this problem is to increase the number of prototypes. In contrast, each set-prototype in GTLVQ is an affine subspace and therefore contains infinitely many points. These additional degrees of freedom increase the model capacity enormously and raise the flexibility of LVQ methods to solve appropriate problems.

3.2 Set-prototypes and respective learning vector quantization variants

In the following, we extend the concept of prototype vectors to set-prototypes, discuss respective dissimilarity measures, and derive tractable solutions for LVQ variants.

Definition 3.1 (set-prototype)**.** Given a classification task, a *set-prototype* $\mathfrak{w}_k$ is a non-empty subset of the data space $\mathbb{R}^{n_x}$ equipped with a fixed class label $c(\mathfrak{w}_k) \in \mathcal{C}$.

Similar to prototype vectors (point-prototypes)—see Definition 2.8—we collect all set-prototypes into a set $\mathcal{W}$. This set contains at least one set-prototype per class.

Now, the dissimilarity definition of Section 2.1.1 cannot be applied because it defines a dissimilarity as a function $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$. Therefore, we extend this definition to a dissimilarity function between points and sets.

Definition 3.2 (point-set dissimilarity)**.** Given a dissimilarity d on $\mathcal{S}$, the *point-set dissimilarity* d is defined as

$$\mathsf{d}\left(x, Y\right) = \inf\left\{d\left(x, y\right) \mid y \in Y\right\}, \tag{3.1}$$

where $x \in \mathcal{S}$ and Y is a non-empty subset of $\mathcal{S}$. In this context, the dissimilarity d is called *underlying dissimilarity.*

Usually, $\mathcal{S}$ is defined as $\mathbb{R}^{n_x}$, Y is a set-prototype $\mathfrak{w}$, and x is an input sample $\mathbf{x}$. Applying these two definitions to the classification principles introduced in Section 2.1.2, we can construct set-prototype-based classifiers. For instance, trivial realizations are

- LVQ methods with set-prototypes as singletons $\mathfrak{w}_k = \{\mathbf{w}_k\}$,
- LVQ methods with several prototypes per class in which all prototypes of a certain class c are collected into a set-prototype, and
- k-nearest neighbors classifiers where all the training points of a class are collected into a respective set-prototype.

These examples imply that the evaluation of the classifier becomes computationally expensive if the cardinality of the set-prototypes increases. Therefore, we are interested in nontrivial realizations where the set-prototypes contain infinitely many elements, but the evaluation of Equation (3.1) is tractable.

3.2.1 Generalized tangent learning vector quantization: Affine subspace prototypes

This section derives the basic GTLVQ scheme, which is based on prototypes as affine subspaces and d_E as underlying dissimilarity. For example, if we assume the data space $\mathbb{R}^{n_x}$ with $n_x \geq 2$, then

- a line is a one-dimensional affine subspace of $\mathbb{R}^{n_x}$ and
- a plane is a two-dimensional affine subspace of $\mathbb{R}^{n_x}$.

Definition 3.3 (affine subspace)**.** An *affine subspace* (linear manifold) is a set

$$\{\mathbf{t} + \mathbf{B}\boldsymbol{\theta} \mid \boldsymbol{\theta} \in \mathbb{R}^{n_s}\},$$

where $\boldsymbol{\theta}$ is a parameter vector, $\mathbf{B} \in \mathbb{R}^{n_x \times n_s}$ is a basis of an n_s-dimensional linear subspace, and $\mathbf{t} \in \mathbb{R}^{n_x}$ is a vector called *translation.* Two affine subspaces that share the same linear subspace are said to be *parallel.*

Model derivation

To derive the GTLVQ method, we define the set-prototypes as affine subspaces with predefined subspace dimension n_s—that is, $\mathfrak{w}_k = \{\mathbf{t}_k + \mathbf{B}_k\boldsymbol{\theta} \mid \boldsymbol{\theta} \in \mathbb{R}^{n_s}\}$—and derive the point-set dissimilarity by

$$\mathsf{d}(\mathbf{x}, \mathfrak{w}) = \min\{d(\mathbf{x}, \mathbf{t} + \mathbf{B}\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \mathbb{R}^{n_s}\}. \tag{3.2}$$

This dissimilarity measure is equivalent to a general single-sided tangent distance (see Schwenk & Milgram, 1995, for single-sided tangent distances). However, we call this dissimilarity simply *general tangent distance.* The dissimilarity becomes tractable if we have an analytical expression to calculate the minimum value. This is ensured if we use the Euclidean distance d_E as the underlying dissimilarity, resulting in the so-called *tangent distance*:

$$\mathsf{d}(\mathbf{x}, \mathfrak{w}) = \min\{d_E(\mathbf{x}, \mathbf{t} + \mathbf{B}\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \mathbb{R}^{n_s}\}. \tag{3.3}$$

The necessary condition for a minimum is

$$\nabla_{\boldsymbol{\theta}}\mathsf{d}(\mathbf{x}, \mathfrak{w}) = \mathbf{0}.$$

Solving this equation leads to the explicit formula

$$\boldsymbol{\theta}^* = \mathbf{B}^{\mathrm{T}}(\mathbf{x} - \mathbf{t}), \tag{3.4}$$

assuming that $\mathbf{B}$ is an orthonormal basis (i. e., $\mathbf{B}^{\mathrm{T}}\mathbf{B} = \mathbf{I}_{n_s}$).[3] Furthermore, it follows

$$\frac{\partial^2\mathsf{d}(\mathbf{x}, \mathfrak{w})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}} = \mathbf{I}_{n_s}$$

so that the solution $\boldsymbol{\theta}^*$ determines a minimum.[4] Therefore, we can substitute the solution $\boldsymbol{\theta}^*$ into the dissimilarity Equation (3.3) and obtain

$$\begin{aligned}\mathsf{d}(\mathbf{x}, \mathfrak{w}) &= d_E\left(\mathbf{x}, \mathbf{t} + \mathbf{B}\mathbf{B}^{\mathrm{T}}(\mathbf{x} - \mathbf{t})\right),\\ &= \sqrt{(\mathbf{P}(\mathbf{x} - \mathbf{t}))^{\mathrm{T}}(\mathbf{P}(\mathbf{x} - \mathbf{t}))},\end{aligned} \tag{3.5}$$

[3]The orthonormality assumption is not necessary to solve the minimization problem (e. g., Simard et al., 1993). However, it is beneficial to keep the equation simple.

[4]Note that if $\mathbf{x} \in \{\mathbf{t} + \mathbf{B}\boldsymbol{\theta} \mid \boldsymbol{\theta} \in \mathbb{R}^{n_s}\}$, the solution $\boldsymbol{\theta}^*$ is still valid even though the gradient $\nabla_{\boldsymbol{\theta}}\mathsf{d}(\mathbf{x}, \mathfrak{w})$ does not exist.

where $\mathbf{P} = \mathbf{I}_{n_x} - \mathbf{B}\mathbf{B}^{\mathrm{T}}$ can be identified as the orthogonal projector onto the complement of the linear subspace spanned by $\mathbf{B}$. Hence, the Equation (3.5) determines the dissimilarity as the shortest path regarding the Euclidean distance after projecting the vectors $\mathbf{x}$ and $\mathbf{t}$ to the complement of the linear subspace. This is, in fact, the smallest dissimilarity of $\mathbf{x}$ to the affine subspace defined by $\mathfrak{w}$.

Since the matrix $\mathbf{P} \in \mathbb{R}^{n_x \times n_x}$ is an orthogonal projector, it has some special properties:

- $\mathbf{P}$ is idempotent (i. e., $\mathbf{P}^2 = \mathbf{P}$);
- $\mathbf{P}$ is symmetric (i. e., $\mathbf{P}^{\mathrm{T}} = \mathbf{P}$).

Thus, we can simplify the Equation (3.5) further to

$$\mathsf{d}\left(\mathbf{x}, \mathfrak{w}\right) = \sqrt{\left(\mathbf{x} - \mathbf{t}\right)^{\mathrm{T}} \mathbf{P} \left(\mathbf{x} - \mathbf{t}\right)}.$$

This equation is used to determine the best matching set-prototype $\mathfrak{w}^*$ according to Equation (2.10). Additionally, according to the BMPP, we assign the class $c\left(\mathfrak{w}^*\right)$ to a given data point $\mathbf{x}$. Note that we transfer all the mathematical notations from prototype vectors to set-prototypes accordingly. For example, we use $\mathfrak{w}^{\pm}$ to denote the closest correct and incorrect set-prototype and $\mathbf{t}^{\pm}$ and $\mathbf{B}^{\pm}$ to denote the corresponding translations and bases.

Training

Similar to GMLVQ defined in Section 2.2.3, we train a GTLVQ model by the GLVQ loss function according to Equation (2.15) and by adapting all the learning rules accordingly. Therefore, the vector of trainable parameters $\boldsymbol{\vartheta}$ is defined by all parameters of all set-prototypes $\mathfrak{w}_k$—that is, by the parameters of each $\mathbf{t}_k$ and $\mathbf{B}_k$. Consequently, we learn all the affine subspaces. In the following, we derive the learning rules for the squared tangent distance. The use of the squared tangent distance simplifies the learning rules, and the learning rules for the non-squared version result directly from these equations.

Based on a training sample $(\mathbf{x}, c\left(\mathbf{x}\right))$, we determine the closest correct set-prototype $\mathfrak{w}^+$ and incorrect set-prototype $\mathfrak{w}^-$. Only for these set-prototypes, the gradients are nontrivial. Similar to Equation (2.20), the update rule for the translation is

$$\mathbf{t}^{\pm} \leftarrow \mathbf{t}^{\pm} \mp \eta \xi^{\pm} \frac{\partial \mathsf{d}^2\left(\mathbf{x}, \mathfrak{w}^{\pm}\right)}{\partial \mathbf{t}^{\pm}} \tag{3.6}$$

with the gradient

$$\nabla_{\mathbf{t}} \mathsf{d}^2\left(\mathbf{x}, \mathfrak{w}\right) = -2\mathbf{P}\left(\mathbf{x} - \mathbf{t}\right)$$

of the squared tangent distance with respect to $\mathbf{t}$. The variables $\xi^{\pm}$ are the gradient scaling factors according to Equation (2.19). Moreover, similar to Equation (2.24), the learning rule for the basis matrices is given by

$$\mathbf{B}^{\pm} \leftarrow \mathbf{B}^{\pm} \mp \eta \xi^{\pm} \frac{\partial \mathsf{d}^2\left(\mathbf{x}, \mathfrak{w}^{\pm}\right)}{\partial \mathbf{B}^{\pm}}. \tag{3.7}$$

The gradient of d^2 with respect to the basis matrix is

$$\nabla_{\mathbf{B}} \mathsf{d}^2\left(\mathbf{x}, \mathfrak{w}\right) = -2\left(\mathbf{x} - \mathbf{t}\right)\left(\mathbf{x} - \mathbf{t}\right)^{\mathrm{T}} \mathbf{B}.$$

After we have adjusted the bases, we must orthonormalize the two new bases $\mathbf{B}^{+}$ and $\mathbf{B}^{-}$—for instance, by applying the Gram–Schmidt process. This is necessary to guarantee the orthonormality assumption that was used for solving the tangent distance minimization problem of Equation (3.3). At the same time, this step prevents possible degeneration effects: Suppose that we initialized each $\mathbf{B}_k$ as an n_s-dimensional basis and that we only apply small updates. Then, the orthonormalization avoids that the basis vectors of $\mathbf{B}_k$ can become linearly dependent and thus do not form an n_s-dimensional basis. Additionally, this step provides a regularization for the matrices $\mathbf{B}_k$ by restricting the solution space.

The complete learning process of the bases is a kind of projected gradient descent. This means that we apply an update that may result in a violation of the solution assumption—that is, that the matrices are orthonormal bases. Therefore, we then apply an appropriate projection strategy to map the updated bases back to the solution space.

Considering the learning rules, see Equation (3.6) and Equation (3.7), we notice that the SGD updates the affine subspaces in such a way that the correct affine subspace becomes more similar to data points of the respective class and is thus attracted. At the same time, the updates try to make the incorrect class more dissimilar by pushing away the corresponding affine subspace. In summary, this leads to a discriminative learning of the affine subspaces with respect to the given classification task.

Initialization

To determine the number of prototypes per class, we apply strategies known from basic LVQ methods. Furthermore, the translations $\mathbf{t}_k$ can be treated as prototype vectors and thus can be initialized by LVQ strategies, see Section 2.2.1. We recommend to use a k-means initialization for each class in which the number of means corresponds to the number of prototypes in the respective class.

After that, we initialize each basis $\mathbf{B}_k$ using the following procedure:

1. Determine all the training samples of the correct class for which $\mathbf{t}_k$ is the closest prototype vector in terms of the underlying dissimilarity. Hence, we consider $\mathbf{t}_k$ as a prototype vector of an ordinary LVQ approach and determine all the training samples that belong to the receptive field of $\mathbf{t}_k$.
2. Compute the n_s eigenvectors that belong to the n_s largest eigenvalues of the estimated covariance matrix over these training samples.
3. Use these n_s eigenvectors as initialization for $\mathbf{B}_k$ and orthonormalize the resulting matrix if necessary.

In principle, the whole procedure is similar to a principal component analysis with the predefined mean vector $\mathbf{t}_k$ and the training samples from the receptive field. Thereby, the computationally expensive step is the calculation of the eigenvectors. This usually performed by a singular value decomposition. However, if the number of dimensions n_x becomes large, then this could be a bottleneck. In this case, we recommend to use the Oja–Sanger method,[5] which estimates the eigenvectors of the covariance matrix in an online process (Sanger, 1989). If we do not have access to the training dataset in advance, we initialize the bases by random orthonormal bases.

As opposed to GLVQ, we have to define the additional hyperparameter n_s, the subspace dimension, in advance. In general, this parameter should be much smaller than the data space dimension n_x (i. e., $n_s \ll n_x$) so it is unlikely that a training sample is simultaneously an element of two set-prototypes of different classes. In consequence, a division by zero in the GLVQ loss is avoided. To find a good setting for n_s, we recommend slowly increasing n_s starting from a very small value. For each setting of n_s, we train the model for only a few iterations or apply the initialization approach to get an estimate of the model performance. As a baseline, we start with a GLVQ method that uses the same underlying dissimilarity. After that, we use a GTLVQ model with $n_s = 1$. Then, we increase n_s as long as no significant performance gain can be observed. The resulting value is used to train the algorithm. In addition, techniques for estimating the intrinsic dimension of the dataset can be used to determine an estimate for n_s.

3.2.2 Restricted generalized tangent learning vector quantization: n_s-orthotope prototypes

An extension of the GTLVQ method and, moreover, another realization of set-prototypes is the application of a restriction on the affine subspaces. In the following, we will consider box-constraints on $\boldsymbol{\theta}$ so that the affine subspaces become n_s-dimensional orthotopes.

[5] Also denoted as generalized Hebbian algorithm or Sanger's rule.

Definition 3.4 (n_s-orthotope)**.** An n_s*-orthotope* (box or hyperrectangle) is a set

$$\{\mathbf{t} + \mathbf{B}\boldsymbol{\theta} \mid \boldsymbol{\theta} \in \mathcal{R} \subset \mathbb{R}^{n_s}\},$$

where $\boldsymbol{\theta}$ is a parameter vector, $\mathbf{B} \in \mathbb{R}^{n_x \times n_s}$ is a basis of an n_s-dimensional linear subspace, and $\mathbf{t} \in \mathbb{R}^{n_x}$ is a translation. The set $\mathcal{R} \subset \mathbb{R}^{n_s}$ is a centered and axis-aligned hyperrectangle:

$$\mathcal{R} = [-a_1, a_1] \times [-a_2, a_2] \times \cdots \times [-a_{n_s}, a_{n_s}].$$

In this context, the symbol $\times$ denotes the Cartesian product. The nonnegative parameters a_i (i. e., $a_i \geq 0$) define the edges of the centered and axis-aligned hyperrectangle. We combine all the parameters a_i to a vector $\mathbf{a}$.

Note that the assumption of a centered and axis-aligned hyperrectangle is not a limitation: Every n_s-orthotope that is constructed on a non-centered and axis-aligned hyperrectangle is equivalent to an n_s-orthotope constructed on a centered and axis-aligned hyperrectangle. For a proof of this claim and all the following theorems, we refer to [2017a].

Model derivation

To derive the restricted-GTLVQ method, we define the set-prototypes as n_s-orthotopes with predefined subspace dimension n_s according to Definition 3.4—that is, $\mathfrak{w}_k = \{\mathbf{t}_k + \mathbf{B}_k\boldsymbol{\theta} \mid \boldsymbol{\theta} \in \mathcal{R}_k \subset \mathbb{R}^{n_s}\}$. Similar to GTLVQ, we use the Euclidean distance as underlying dissimilarity to keep the point-set dissimilarity tractable. We call this point-set dissimilarity *restricted tangent distance* because the set-prototypes are restricted affine subspaces:

$$\mathsf{d}(\mathbf{x}, \mathfrak{w}) = \min\{d_E(\mathbf{x}, \mathbf{t} + \mathbf{B}\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \mathcal{R} \subset \mathbb{R}^{n_s}\}. \tag{3.8}$$

The following theorem provides the solution for this optimization problem.

Theorem 3.1. *Assume that* $\mathbf{B}$ *is an orthonormal basis. The set of optimal solutions of Equation (3.8) is a singleton with the element*

$$\boldsymbol{\theta}^* = H\left(\mathbf{a} - \left|\hat{\boldsymbol{\theta}}\right|\right) \circ \hat{\boldsymbol{\theta}} + \operatorname{sgn}\left(\hat{\boldsymbol{\theta}}\right) \circ \left(\mathbf{1} - H\left(\mathbf{a} - \left|\hat{\boldsymbol{\theta}}\right|\right)\right) \circ \mathbf{a}, \tag{3.9}$$

where $\hat{\boldsymbol{\theta}} = \mathbf{B}^{\mathrm{T}}(\mathbf{x} - \mathbf{t})$ *is the solution of the unrestricted tangent distance problem, see Equation (3.4) and Equation (3.3), respectively.*

Thereby, $H(\cdot)$ denotes the Heaviside step function defined in Equation (2.14), $|\cdot|$ denotes the absolute value operation, and $\operatorname{sgn}(\cdot)$ denotes the sign function—the functions are extended to vectorial inputs by applying them element-wise. The symbol $\circ$ denotes the Hadamard product (element-wise multiplication).

The solution $\boldsymbol{\theta}^*$ is element-wise equivalent to

$$\theta_i^* = \begin{cases} a_i & \text{if } \hat{\theta}_i \geq a_i, \\ \hat{\theta}_i & \text{if } \left|\hat{\theta}_i\right| < a_i, \\ -a_i & \text{otherwise.} \end{cases}$$

This shows how the solution is constructed: We compute the unrestricted solution and clip all the values outside of $\mathcal{R}$ to the boundary.

Proof sketch. The theorem is proven by the following two steps:

1. We prove that there exists a solution and that the solution is unique.
2. We prove that the vector $\boldsymbol{\theta}^*$ is optimal by verifying the Karush–Kuhn–Tucker and Slater condition.[6] □

With $\boldsymbol{\theta}^*$, we derived a tractable solution of the Equation (3.8) so that we can efficiently determine the BMPP. Similar to GTLVQ, we transfer all the mathematical notations from ordinary prototypes to the defined set-prototypes accordingly.

Training

To derive the learning equations, we need the gradients with respect to all the trainable parameters—that is, with respect to $\mathbf{t}$, $\mathbf{B}$, and $\mathbf{a}$. As opposed to GTLVQ, it is not obvious that the Equation (3.8) is differentiable because the solution $\boldsymbol{\theta}^*$ contains non-differentiable parts like the Heaviside step function. The following theorem ensures the differentiability on the required domain for SGD.

Theorem 3.2. *Let* $\mathbf{x} \in \mathbb{R}^{n_x}$ *be arbitrary but fixed. The restricted tangent distance according to Equation (3.8) is differentiable with respect to the variables* $\mathbf{t}$, $\mathbf{B}$, *and* $\mathbf{a}$ *on the domain*

$$\left\{(\mathbf{t}, \mathbf{B}, \mathbf{a}) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x \times n_s} \times \mathbb{R}^{n_s}_{\geq 0} \mid \mathbf{x} - \mathbf{t} \neq \mathbf{B}\boldsymbol{\theta}^* \text{ and } a_i > 0 \text{ for all } i\right\}. \tag{3.10}$$

Proof sketch. A function $f(\mathbf{x})$ defined on an open domain $D(f) \subseteq \mathbb{R}^{n_x}$ is differentiable if

- $f(\mathbf{x})$ is continuous on $D(f)$,
- the partial derivatives $\frac{\partial f(\mathbf{x})}{\partial x_i}$ exist on $D(f)$ for all i, and
- the partial derivatives $\frac{\partial f(\mathbf{x})}{\partial x_i}$ are continuous on $D(f)$ for all i.

[6]We refer to Boyd and Vandenberghe (2004) regarding the two conditions.

These three conditions are sufficient for the differentiability of f and we prove the theorem by validating them. □

Note that we ensure the open set assumption in Equation (3.10) by excluding the boundary points (i. e., by the condition $a_i > 0$ for all i). We have also excluded points where the derivative of the real square root does not exist:

$$\begin{aligned} d_E\left(\mathbf{x}, \mathbf{t} + \mathbf{B}\boldsymbol{\theta}^*\right) &= 0, \\ \mathbf{x} &= \mathbf{t} + \mathbf{B}\boldsymbol{\theta}^*, \\ \mathbf{x} - \mathbf{t} &= \mathbf{B}\boldsymbol{\theta}^*. \end{aligned}$$

Based on this theorem, the relation between the gradient and the partial derivatives holds so that we can derive the learning equations. The learning rule for the translations is equivalent to Equation (3.6) with the gradient

$$\nabla_{\mathbf{t}} \mathsf{d}^2\left(\mathbf{x}, \mathfrak{w}\right) = -2\left(\mathbf{x} - \mathbf{t}\right) + 2\mathbf{B}\boldsymbol{\theta}^*.$$

Note that this gradient becomes equal to the gradient of the tangent distance if $\boldsymbol{\theta}^*$ equals $\hat{\boldsymbol{\theta}}$. This has to be true because, in this case, the restricted tangent distance is equal to the tangent distance. A similar result holds for the gradient with respect to the basis—this gradient is

$$\nabla_{\mathbf{B}} \mathsf{d}^2\left(\mathbf{x}, \mathfrak{w}\right) = -2\left(\mathbf{x} - \mathbf{t}\right)\left(\boldsymbol{\theta}^*\right)^{\mathrm{T}}.$$

Using this gradient in the learning rule according to Equation (3.7), we obtain the learning rule for the basis matrices.

As opposed to GTLVQ, we learn the centered and axis-aligned hyperrectangle $\mathcal{R}$ represented by the vector $\mathbf{a}$. The respective SGD rule is given by

$$\mathbf{a}^{\pm} \leftarrow \mathbf{a}^{\pm} \mp \eta \xi^{\pm} \frac{\partial \mathsf{d}^2\left(\mathbf{x}, \mathfrak{w}^{\pm}\right)}{\partial \mathbf{a}^{\pm}}, \tag{3.11}$$

where the gradient is

$$\nabla_{\mathbf{a}} \mathsf{d}^2\left(\mathbf{x}, \mathfrak{w}\right) = -2\left|\boldsymbol{\theta}^* - \hat{\boldsymbol{\theta}}\right|.$$

Equivalent to GTLVQ, we perform a projected gradient descent learning and project the bases $\mathbf{B}^+$ and $\mathbf{B}^-$ back into the space of orthonormal basis matrices after each update. However, we cannot apply an arbitrary orthonormalization method, because a basis representation $\mathbf{B}$ of a linear subspace is not unique and an n_s-orthotope is not invariant to $\mathbf{B}$. The repeated application of an orthonormalization method could, therefore, result in different basis representations $\mathbf{B}$. For example, suppose that we permute the order of the basis vectors. On the one hand, the result is that the basis spans the same linear subspace as before.[7] On the other hand, we permute the orthotope because the dimensions of $\mathcal{R}$ are related to the basis vectors and we do not permute the intervals accordingly. Consequently, we have to use an orthonormalization method that returns the closest basis representation regarding the given matrix.

A possible solution for this problem is related to the so-called *orthogonal Procrustes problem* with which an orthogonal matrix $\mathbf{B}$ can be determined that is closest to a given matrix $\tilde{\mathbf{B}}$ in terms of the Frobenius norm (Schönemann, 1966). Assume that $\tilde{\mathbf{B}} \in \mathbb{R}^{n_x \times n_s}$ is the basis after an SGD update. Moreover, $\mathbf{U\Sigma V}^{\mathrm{T}}$ is the singular value decomposition of $\tilde{\mathbf{B}}$—that is, $\tilde{\mathbf{B}} = \mathbf{U\Sigma V}^{\mathrm{T}}$, where $\mathbf{U} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{\Sigma} \in \mathbb{R}_{\geq 0}^{n_x \times n_s}$, and $\mathbf{V} \in \mathbb{R}^{n_s \times n_s}$. Then, we search for a matrix $\mathbf{B} \in \mathbb{R}^{n_x \times n_s}$ such that

$$\mathbf{B} = \underset{\mathbf{\Omega} \in \mathbb{R}^{n_x \times n_s}}{\arg\min} \left\| \mathbf{\Omega} - \tilde{\mathbf{B}} \right\|_F^2 \text{ subject to } \mathbf{\Omega}^{\mathrm{T}}\mathbf{\Omega} = \mathbf{I}_{n_s}.$$

Using the relationship to the trace and the linearity property, the expression of the Frobenius norm can be rewritten to

$$\mathbf{B} = \underset{\mathbf{\Omega} \in \mathbb{R}^{n_x \times n_s}}{\arg\min} \left(\mathrm{tr}\left(\mathbf{\Omega}^{\mathrm{T}}\mathbf{\Omega}\right) - 2\mathrm{tr}\left(\tilde{\mathbf{B}}^{\mathrm{T}}\mathbf{\Omega}\right) + \mathrm{tr}\left(\tilde{\mathbf{B}}^{\mathrm{T}}\tilde{\mathbf{B}}\right)\right) \text{ subject to } \mathbf{\Omega}^{\mathrm{T}}\mathbf{\Omega} = \mathbf{I}_{n_s}.$$

Since $\mathbf{\Omega}$ is an orthonormal basis, this optimization is equivalent to

$$\mathbf{B} = \underset{\mathbf{\Omega} \in \mathbb{R}^{n_x \times n_s}}{\arg\max} \; \mathrm{tr}\left(\tilde{\mathbf{B}}^{\mathrm{T}}\mathbf{\Omega}\right) \text{ subject to } \mathbf{\Omega}^{\mathrm{T}}\mathbf{\Omega} = \mathbf{I}_{n_s}.$$

By substituting $\tilde{\mathbf{B}}$ with the singular value decomposition and by the property that the trace is invariant under cyclic permutations, we get

$$\mathbf{B} = \underset{\mathbf{\Omega} \in \mathbb{R}^{n_x \times n_s}}{\arg\max} \; \mathrm{tr}\left(\mathbf{\Sigma}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{\Omega}\mathbf{V}\right) \text{ subject to } \mathbf{\Omega}^{\mathrm{T}}\mathbf{\Omega} = \mathbf{I}_{n_s}.$$

Because $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices—characteristic of the singular value decomposition—and $\mathbf{\Omega}$ is an orthonormal basis, the product $\mathbf{U}^{\mathrm{T}}\mathbf{\Omega}\mathbf{V}$ is an orthonormal basis matrix of size $n_x \times n_s$. Since $\mathbf{\Sigma}$ is a nonnegative diagonal matrix of size $n_x \times n_s$,[8] the trace is maximum if

$$\begin{aligned} \mathbf{U}^{\mathrm{T}}\mathbf{\Omega}\mathbf{V} &= \mathbf{I}_{n_x \times n_s}, \\ \mathbf{\Omega} &= \mathbf{U}\mathbf{I}_{n_x \times n_s}\mathbf{V}^{\mathrm{T}}, \end{aligned}$$

[7]Thus, an affine subspace is invariant to the basis representation $\mathbf{B}$ of the linear subspace, so GTLVQ does not depend on a specific orthonormalization method.

[8]A rectangular diagonal matrix $\mathbf{\Sigma}$ is a matrix where all the entries Σ_{ij} with $i \neq j$ are zero.

where $\mathbf{I}_{n_x \times n_s}$ is a diagonal matrix of size $n_x \times n_s$ with ones on the main diagonal. Finally, the optimal orthonormal basis is

$$\mathbf{B} = \mathbf{U}\mathbf{I}_{n_x \times n_s}\mathbf{V}^{\mathrm{T}}.$$

If we take the compact singular value decomposition[9] of $\tilde{\mathbf{B}}$—that is, $\tilde{\mathbf{B}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$, where $\mathbf{U} \in \mathbb{R}^{n_x \times n_s}$, $\boldsymbol{\Sigma} \in \mathbb{R}_{\geq 0}^{n_s \times n_s}$, and $\mathbf{V} \in \mathbb{R}^{n_s \times n_s}$—the solution simplifies to

$$\mathbf{B} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$$

and the computation becomes more efficient. Similar to GTLVQ, the orthonormalization prevents possible degeneration effects and provides a regularization during training.

In addition to the orthonormalization, we have to ensure that each element a_i of the vector $\mathbf{a}$ is greater than zero to guarantee the differentiability. This can be realized by encoding the vector $\mathbf{a}$ into a vector $\tilde{\mathbf{a}} \in \mathbb{R}^{n_s}$ with the element-wise decoding rule $a_i = \exp(\tilde{a}_i)$. Then, the learning rule of Equation (3.11) is modified to a learning rule with respect to $\tilde{\mathbf{a}}$.

In general, the learning behavior of the restricted-GTLVQ is similar to that of the GTLVQ—it tries to make the orthotopes more discriminative for the classification task. This consists of

- the attraction of the orthotope of the correct class and an increase of the size in the direction of the data point and
- the repulsion of the orthotope of the incorrect class and a decrease of the size with respect to the data point.

Initialization

Restricted-GTLVQ can be initialized with the same methods that are used for GTLVQ. For instance, during the initialization phase, we set the elements of $\mathbf{a}$ to such large values that restricted-GTLVQ is like GTLVQ. Then, we initialize the translations and the bases with the proposed GTLVQ initialization methods. After that, we set the elements in $\mathbf{a}$ to such small values that restricted-GTLVQ is similar to GLVQ. In this setting, we start the training, and during this time, the orthotopes will grow to a size that is necessary to solve the classification task.

[9]Sometimes referred to as thin or economical singular value decomposition.

3.3 Relations to other concepts

The previously derived GTLVQ algorithms have some commonalities with other approaches or concepts. This section examines these relations. In the first subsection, we analyze the relation of point-set dissimilarities to Hausdorff distances. The presented theorem provides an important result about the tangent distance: The tangent distance can be interpreted as a Hausdorff distance and thus can be considered as a metric. Subsequently, we describe how the concept of (restricted) tangent distances is related to tangent space approximations of manifolds. This relation helps to understand the learning behavior and the way the set-prototypes are aligned during training. Finally, we investigate similarities between GMLVQ methods and GTLVQ. The result of this study is that the mathematical properties of GTLVQ prevent oversimplification and that GTLVQ performs a self-regularization. Furthermore, the results show that local-GMLVQ can become equivalent to GTLVQ if local-GMLVQ is trained with the commonly used regularization term. This strong relation provides new interpretation techniques for the learned transformation matrices in standard GMLVQ and local-GMLVQ.

3.3.1 Hausdorff distances

Felix Hausdorff was a German mathematician and is considered to be one of the founders of modern topology. He defined the so-called Hausdorff distance, which measures the dissimilarity between two subsets of a metric space $(\mathcal{S}, d)$.

Definition 3.5 (Hausdorff distance)**.** Let $(\mathcal{S}, d)$ be a metric space and $X, Y \subseteq \mathcal{S}$ be two non-empty subsets of $\mathcal{S}$. Their *Hausdorff distance* $d_H(X, Y)$ is defined by

$$d_H(X, Y) = \max\left\{\sup_{x\in X}\inf_{y\in Y} d(x, y), \sup_{y\in Y}\inf_{x\in X} d(x, y)\right\}. \tag{3.12}$$

The Hausdorff distance can be constructed by several extensions of the metric d. Suppose we have the metric space $(\mathbb{R}, d_E)$, a set $A = \{1, 2, 7, 9\}$, and another set $B = \{2, 7\}$. As usual in the Euclidean geometry, we define the distance between a point and a non-empty set by the point-set dissimilarity $\mathsf{d}(x, Y)$ (e. g., $\mathsf{d}(1, B) = 1$). The next extension is that we define a dissimilarity between X and Y by

$$d(X, Y) = \sup_{x\in X} \mathsf{d}(x, Y).$$

With the previously defined sets, $d(A, B) = 2$ and $d(B, A) = 0$. In general, $d(X, Y)$ is not symmetric and we define a symmetric version by taking the maximum value over both expressions. The reason to take the maximum and not the minimum value is that we want to measure the difference (dissimilarity) between the two sets. Finally, the Hausdorff distance between A and B is $d_H(A, B) = 2$.

Lemma 3.1. *Let $\mathcal{P}(\mathcal{S})$ be the power set of $\mathcal{S}$ without the empty set. The Hausdorff distance on $\mathcal{P}(\mathcal{S})$, defined by*

$$d_H : \mathcal{P}(\mathcal{S}) \times \mathcal{P}(\mathcal{S}) \longrightarrow \bar{\mathbb{R}},$$

is an extended semimetric. This is a semimetric according to Definition 2.3 with output values in the extended real numbers $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$.

Thus, Definition 3.5 defines an extended semimetric. Interestingly, by a small modification, d_H can become a metric.

Lemma 3.2. *Let $\mathcal{F}(\mathcal{S})$ be the set of all non-empty compact subsets of $\mathcal{S}$. The Hausdorff distance on $\mathcal{F}(\mathcal{S})$, defined by*

$$d_H : \mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S}) \longrightarrow \mathbb{R},$$

is a metric according to Definition 2.1 and d_H can be simplified to

$$d_H(X, Y) = \max\left\{\max_{x \in X} \min_{y \in Y} d(x, y), \max_{y \in Y} \min_{x \in X} d(x, y)\right\}. \tag{3.13}$$

We refer to Henrikson (1999) for a proof of the lemma. It is notable that these results are independent of the particular distance measure d.

In the following, we formulate a lemma about the relation between the general tangent distance according to Equation (3.2) and the Hausdorff distance. This lemma requires a translation-invariant metric and we then discuss whether the lemma would be generally provable without this property. Based on these results, we derive the theorem that describes the relationship between an arbitrary point-set dissimilarity according to Equation (3.1) and the Hausdorff distance.

Relation to the general tangent distance according to Equation (3.2)

By definition, the point-set dissimilarity is part of the Hausdorff distance and, hence, it is obvious to investigate the relationship. The following lemma is a special case of the theorem that we will introduce later and so the proof is provided by the theorem.

Lemma 3.3. *Let $(\mathcal{S}, d)$ be a metric space with d being a translation-invariant metric according to Definition 2.2, $(\mathcal{S}, +)$ be a group, and Y be a left coset—that is,*

$$Y = \{g + h \mid h \in H\},$$

where $g \in \mathcal{S}$ is a translation element and $H \subseteq \mathcal{S}$ is a subgroup. Then the following holds:

$$d_H(X, Y) = \mathsf{d}(x, Y), \tag{3.14}$$

where

$$X = \{x + h \mid h \in H\}$$

is a left coset that has the same subgroup as Y but the element x as translation.

This lemma states that we can always construct a left coset X such that the general tangent distance is equivalent to the Hausdorff distance.

In the definition of the tangent distance according to Equation (3.3), we used the Euclidean distance as underlying dissimilarity that is a translation-invariant metric. Moreover, an affine subspace is a left coset of the vector space. Consequently, the lemma applies to the definition of the tangent distance. The statement of the lemma is now that we can always construct an affine subspace with $\mathbf{x}$ as the translation such that it is parallel to the prototype $\mathfrak{w}$ so that the Hausdorff distance between these two parallel subspaces is equal to the tangent distance between $\mathbf{x}$ and $\mathfrak{w}$. Therefore, we know that we can formulate the tangent distance in the form of a Hausdorff distance so that the distance function is an extended semimetric. This result becomes important if a mathematical statement, method, or algorithm requires metric properties.

Is the lemma provable for non-translation-invariant metrics?

In general, it is not. Suppose that the lemma is true without the translation invariance assumption. We use the non-translation-invariant railway metric[10] to construct a counterexample where the dissimilarity values between the general tangent distance and the Hausdorff distance are different.

We define the *railway metric* over the normed vector space $\left(\mathbb{R}^2, \|\cdot\|_E\right)$ by

$$d_R(\mathbf{x}, \mathbf{y}) = \begin{cases} \|\mathbf{x} - \mathbf{y}\|_E & \text{if there exists a line } L \\ & \text{such that } \mathbf{x}, \mathbf{y}, \mathbf{o} \in L, \\ \|\mathbf{x} - \mathbf{o}\|_E + \|\mathbf{y} - \mathbf{o}\|_E & \text{otherwise,} \end{cases} \tag{3.15}$$

where $\|\cdot\|_E$ denotes the Euclidean norm, $\mathbf{o} \in \mathbb{R}^2$ is an arbitrary but fixed element—denoted as origin—and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ are arbitrary input elements. This metric is a common example for a non-translation-invariant metric. The general tangent distance according to Equation (3.2) with one-dimensional affine subspaces and d_R as underlying dissimilarity is

$$\mathsf{d}(\mathbf{x}, \mathfrak{w}) = \min\{d_R(\mathbf{x}, \mathbf{t} + \alpha\mathbf{r}) \mid \alpha \in \mathbb{R}\}. \tag{3.16}$$

[10] Also denoted as British rail metric, post office metric or SNCF metric.

In addition, for the counterexample, we define the variables to $\mathbf{t} = (2,0)^{\mathrm{T}}$, $\mathbf{r} = (0,1)^{\mathrm{T}}$, $\mathbf{x} = (1,0)^{\mathrm{T}}$, and the origin to $\mathbf{o} = (0,0)^{\mathrm{T}}$. Consequently, the railway metric according to Equation (3.15) simplifies to

$$d_R(\mathbf{x},\mathbf{y}) = \begin{cases} \|\mathbf{x}-\mathbf{y}\|_E & \text{if } \mathbf{x} \text{ and } \mathbf{y} \text{ are linear dependent,} \\ \|\mathbf{x}\|_E + \|\mathbf{y}\|_E & \text{otherwise.} \end{cases} \tag{3.17}$$

Compute the general tangent distance: We calculate the general tangent distance for the defined vectors. Hence, we search for the minimum value along the line $\mathfrak{w} = \mathbf{t} + \alpha\mathbf{r}$ with respect to the parameter α. The first case of Equation (3.17) applies to all values s and α for which

$$\begin{aligned} \mathbf{t} + \alpha\mathbf{r} &= s\mathbf{x}, \\ \begin{pmatrix} 2 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 1 \end{pmatrix} &= s \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \end{aligned}$$

This is true for $s = 2$ and $\alpha = 0$. However, if $\alpha = 0$, the dissimilarity value is calculated to

$$\begin{aligned} d_R(\mathbf{x},\mathbf{t}+\alpha\mathbf{r}) = &\ \|\mathbf{x}-\mathbf{t}\|_E\,, \\ = &\left\| - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\|_E, \\ = &\ 1. \end{aligned}$$

The second case of Equation (3.17) applies if $\alpha \neq 0$. Now, the railway metric equals

$$\begin{aligned} d_R(\mathbf{x},\mathbf{t}+\alpha\mathbf{r}) = &\ \|\mathbf{x}\|_E + \|\mathbf{t}+\alpha\mathbf{r}\|_E\,, \\ = &\left\| \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\|_E + \left\| \begin{pmatrix} 2 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\|_E, \\ = &\ 1 + \sqrt{4+\alpha^2}. \end{aligned}$$

If both results are combined, the general tangent distance, according to Equation (3.16), is computed to

$$\begin{aligned} \mathsf{d}(\mathbf{x},\mathfrak{w}) = &\ \min \begin{cases} 1 & \text{if } \alpha = 0, \\ 1 + \sqrt{4+\alpha^2} & \text{otherwise,} \end{cases} \\ = &\ 1. \end{aligned}$$

Compute the Hausdorff distance: We calculate the Hausdorff distance $d_H\left(Z, \mathfrak{w}\right)$, where

$$Z = \left\{\mathbf{z} \in \mathbb{R}^2 \mid \text{there exists a } \beta \in \mathbb{R} \text{ such that } \mathbf{z} = \mathbf{x} + \beta\mathbf{r}\right\}$$

is the left coset according to the lemma and, hence, the parallel line to $\mathfrak{w}$.

The first condition of the railway metric can be formulated by the parameters α and β. The vectors are linear dependent if

$$\begin{gathered} \mathbf{t} + \alpha\mathbf{r} = s\left(\mathbf{x} + \beta\mathbf{r}\right), \\ \begin{pmatrix} 2 \\ 0 \end{pmatrix} + \alpha\begin{pmatrix} 0 \\ 1 \end{pmatrix} = s\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta\begin{pmatrix} 0 \\ 1 \end{pmatrix}\right) \end{gathered}$$

is fulfilled. This implies $s = 2$ and $\alpha = 2\beta$. The Hausdorff distance is computed by evaluating both expressions of the maximum in Equation (3.12). The first expression can be simplified to

$$\sup_{\mathbf{z}\in Z}\inf_{\mathbf{w}\in\mathfrak{w}} d_R\left(\mathbf{z}, \mathbf{w}\right) = \sup_{\beta\in\mathbb{R}}\inf_{\alpha\in\mathbb{R}} \begin{cases} \sqrt{1 + \left(\alpha - \beta\right)^2} & \text{if } \alpha = 2\beta, \\ \sqrt{1+\beta^2} + \sqrt{4+\alpha^2} & \text{otherwise.} \end{cases} \tag{3.18}$$

Assume an arbitrary but fixed $\beta \in \mathbb{R}$. The infimum of the first case with $\alpha = 2\beta$ calculates to

$$\inf_{\mathbf{w}\in\mathfrak{w}} d_R\left(\mathbf{z}, \mathbf{w}\right) = \sqrt{1+\beta^2}.$$

Since β was arbitrary, the first expression of the Hausdorff distance under the condition $\alpha = 2\beta$ is infinite:

$$\begin{aligned} \sup_{\mathbf{z}\in Z}\inf_{\mathbf{w}\in\mathfrak{w}} d_R\left(\mathbf{z}, \mathbf{w}\right) &= \sup_{\beta\in\mathbb{R}} \sqrt{1+\beta^2}, \\ &= \infty. \end{aligned}$$

Note that we computed the Hausdorff distance on the extended real numbers.

The second case of Equation (3.18) applies if $\alpha \neq 2\beta$. Therefore, the infimum equals

$$\begin{aligned} \inf_{\mathbf{w}\in\mathfrak{w}} d_R\left(\mathbf{z}, \mathbf{w}\right) &= \inf_{\alpha\in\mathbb{R}} \left(\sqrt{1+\beta^2} + \sqrt{4+\alpha^2}\right), \\ &= \sqrt{1+\beta^2} + 2 \end{aligned}$$

with $\alpha = 0$, which implies $\beta \neq 0$. Finally, this case is also infinite:

$$\begin{aligned} \sup_{\mathbf{z}\in Z}\inf_{\mathbf{w}\in\mathfrak{w}} d_R\left(\mathbf{z}, \mathbf{w}\right) &= \sup_{\beta\in\mathbb{R},\ \beta\neq 0} \left(\sqrt{1+\beta^2} + 2\right), \\ &= \infty. \end{aligned}$$

Consequently, the first expression of the Hausdorff distance is infinite. Due to the outer maximum operation of the Hausdorff distance, see Equation (3.12), we can conclude that the Hausdorff distance with the underlying railway metric is infinite—that is, $d_H(Z, \mathfrak{w}) = \infty$.

In summary, we have constructed an example in which the general tangent distance provides a different dissimilarity value than the Hausdorff distance. Hence, we found a counterexample. Therefore, the lemma is generally not provable for non-translation-invariant metrics.

Generalization of the lemma

Lemma 3.3 is limited to cosets and, thus, to the general tangent distance, see Equation (3.2). Now, we generalize the lemma to the following theorem that is valid for an arbitrary set definition and, hence, also applies to the restricted-GTLVQ from Section 3.2.2.

Theorem 3.3. *Let $(\mathcal{S}, +)$ be a group and $(\mathcal{S}, d)$ be a metric space with d being a translation-invariant metric. Given an element x and a non-empty subset Y of $\mathcal{S}$, we assume that the resulting point-set dissimilarity problem has an optimal element y^* in the closure of Y so that*

$$d(x, y^*) = \mathsf{d}(x, Y) = \inf_{y \in Y} d(x, y).$$

Then the following holds:

$$d_H(X, Y) = \mathsf{d}(x, Y),$$

where

$$X = \{x - y^* + y' \mid y' \in Y\}.$$

Note that we use $-y$ to denote the inverse element to y and that we do *not* assume an Abelian group.[11] Additionally, in the following proof, the identity element is denoted by e.

Proof. According to Definition 3.5, the Hausdorff distance is given by

$$d_H(X, Y) = \max\left\{\sup_{x' \in X} \inf_{y \in Y} d(x', y), \sup_{y \in Y} \inf_{x' \in X} d(x', y)\right\}.$$

We prove the theorem by the following steps:

[11] An Abelian group is a group $(\mathcal{S}, +)$ with the additional property that the operation is commutative (i. e., $x + y = y + x$).

1. We show that $\mathsf{d}(x, Y)$ is an upper bound for the first expression of the maximum evaluation of $d_H(X, Y)$.
2. We prove that this bound is the least upper bound and, thus, the supremum.
3. For the second expression of the maximum evaluation of $d_H(X, Y)$, we show that $d(x, y^*)$ is also an upper bound.
4. Due to the outer maximum operation, this proves the theorem.

Step 1: Applying the definition of the set X from the theorem, we obtain

$$\sup_{x' \in X} \inf_{y \in Y} d(x', y) = \sup_{y' \in Y} \inf_{y \in Y} d(x - y^* + y', y).$$

Since the metric d is translation-invariant, we can reformulate the expression to

$$\sup_{x' \in X} \inf_{y \in Y} d(x', y) = \sup_{y' \in Y} \inf_{y \in Y} d(x, y - y' + y^*).$$

Now, we show that $\mathsf{d}(x, Y)$ is an upper bound. Let the element $y' \in Y$ of the supremum be arbitrary but fixed. Because we calculate the infimum, it holds that

$$\inf_{y \in Y} d(x, y - y' + y^*) \leq d(x, y' - y' + y^*) = d(x, y^*).$$

Since y' was arbitrarily chosen, it follows that the point-set dissimilarity is an upper bound:

$$\sup_{x' \in X} \inf_{y \in Y} d(x', y) \leq d(x, y^*).$$

Step 2: Assume that $d(x, y^*)$ is not the least upper bound and, therefore, not the supremum. This means that there is an element $y'' \in \mathcal{S}$ such that

$$\inf_{y \in Y} d(x, y - y'' + y^*) < d(x, y^*) \tag{3.19}$$

and *all* evaluations for $y' \in Y$ are less than or equal to

$$\inf_{y \in Y} d(x, y - y' + y^*) \leq \inf_{y \in Y} d(x, y - y'' + y^*). \tag{3.20}$$

Given an arbitrary y', we define the element in terms of y^* by

$$\begin{aligned} y' &= \underbrace{y' - y^*}_{\delta} + y^*, \\ &= \delta + y^*. \end{aligned} \tag{3.21}$$

Since y^* is in the closure of Y, we can conclude that

$$\inf_{y' \in Y} d\left(y', y^*\right) = 0.$$

Due to the metric properties, we can further conclude that if $y^* \in Y$, then this implies that $y' = y^*$. Moreover, if $y^* \notin Y$, then there is a sequence such that y' converges to y^* if $d\left(e, \delta\right) \to 0$. Furthermore, by substituting y' in Equation (3.20) with the expression from Equation (3.21), we obtain

$$\inf_{y \in Y} d\left(x, y - \delta\right) \leq \inf_{y \in Y} d\left(x, y - y'' + y^*\right).$$

From the previous discussion, we know that we can either select the element y' directly such that $d\left(e, \delta\right) = 0$, or that we can find an element y' that is arbitrarily close to y^* so that $d\left(e, \delta\right) \to 0$. Applying this result in the previous equation yields

$$\inf_{y \in Y} d\left(x, y\right) \leq \inf_{y \in Y} d\left(x, y - y'' + y^*\right).$$

Because $\inf_{y \in Y} d\left(x, y\right) = d\left(x, y^*\right)$, this contradicts the initial assumption in Equation (3.19) and it follows that $d\left(x, y^*\right)$ is the supremum.

Step 3: Similar to the first step, we use the translation invariance of the metric to obtain

$$\sup_{y \in Y} \inf_{x' \in X} d\left(x', y\right) = \sup_{y \in Y} \inf_{y' \in Y} d\left(x, y - y' + y^*\right).$$

Let the element $y \in Y$ of the supremum be arbitrary but fixed. Because we calculate the infimum, it holds that

$$\inf_{y' \in Y} d\left(x, y - y' + y^*\right) \leq d\left(x, y - y + y^*\right) = d\left(x, y^*\right).$$

Since y was arbitrarily chosen, it follows that the point-set dissimilarity is an upper bound:

$$\sup_{y \in Y} \inf_{x' \in X} d\left(x', y\right) \leq d\left(x, y^*\right).$$

Step 4: Since we compute the maximum over both expressions, we have proven that

$$d_H\left(X, Y\right) = \mathsf{d}\left(x, Y\right). \qquad \square$$

It should be noted that, in general, the equality—that is, $d_H\left(X, Y\right) = \mathsf{d}\left(x, Y\right)$—holds for the first argument of the maximum expression. Similar to the interpretation of the Lemma 3.3, the theorem provides an answer about the relation of the Hausdorff distance to an arbitrary point-set dissimilarity. Because we made no assumption

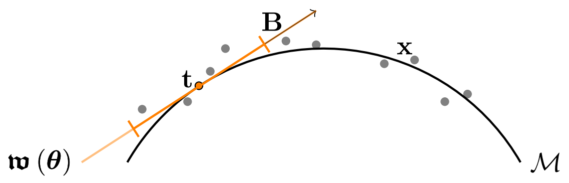


Figure 3.1: Illustration of a tangent space approximation by $\mathfrak{w}(\boldsymbol{\theta})$ of a manifold $\mathcal{M}$ at point $\mathbf{t}$ with tangent basis $\mathbf{B}$ in a two-dimensional embedding space.

about the subset Y and only assumed a group, this theorem applies to the restricted tangent distance according to Equation (3.8), and we can conclude that the distance evaluation can be interpreted as an evaluation of an extended semimetric. Additionally, because the set-prototypes of the restricted tangent distance are compact sets, the distance can be interpreted as a metric evaluation. Furthermore, since the evaluation of a k-nearest neighbors algorithm or a GLVQ method with several prototypes per class can be interpreted as a kind of set-prototype approach, the theorem also applies to these methods and answers the question how the BMPP is related to a metric function.

3.3.2 Tangent space approximations

Tangent distances are inspired by the idea that a certain (maybe technical) system generates data vectors (signals) $\mathbf{x} \in \mathbb{R}^{n_x}$ for a considered task. This system is determined by an intrinsic state that is controlled by a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{n_s}$. Moreover, the intrinsic state influences the signal generation and, therefore, the signal generation depends on $\boldsymbol{\theta}$. For instance, in the first publication about tangent distances, Simard et al. (1993) assumed that the variations in each class of handwritten digits are generated by seven dominating transformations: horizontal and vertical translations, rotation, scaling, two hyperbolic transformations (which can generate shearing and squeezing), and line thickening or thinning. Accordingly, they assumed that the intrinsic state of the system generating the variations for a digit of a given class is only seven-dimensional even though the embedded samples in the image space have a much higher dimension.[12] Therefore, by the application of the tangent distance, they tried to capture information about the data manifold instead of only exploring the high-dimensional embeddings as individual data points.

In Figure 3.1, we present an illustrative example of a tangent space approximation. Assume that the data points $\mathbf{x}$ are (maybe noisy) embedded samples $x \in \mathcal{M}$ of an

[12]This concept is equivalent to the intrinsic dimension.

(usually) unknown smooth n_s-dimensional manifold $\mathcal{M}$.[13] We linearly approximate the embedded manifold curve at $\mathbf{t} \in \mathbb{R}^{n_x}$ by the tangent basis $\mathbf{B} \in \mathbb{R}^{n_x \times n_s}$ resulting in a linear approximation. This approximation is represented by the points $\mathbf{t} + \mathbf{B}\boldsymbol{\theta}$, where $\boldsymbol{\theta} \in \mathbb{R}^{n_s}$ is a local coordinate vector. All these approximation points are collected into a set $\mathfrak{w}$ and we define by $\mathfrak{w}(\boldsymbol{\theta})$ the mapping from a local coordinate vector $\boldsymbol{\theta}$ to the corresponding embedding in $\mathbb{R}^{n_x}$—this set $\mathfrak{w}$ is an affine subspace.

Relation to GTLVQ

If we assume that the data samples of a class are embedded samples of a manifold, then the GTLVQ algorithm tries to approximate the class manifold by respective tangent spaces. Each tangent space is a set-prototype and, in particular, an affine subspace. By presenting samples from the class during training, the algorithm aligns the set-prototypes such that they capture the data structure and thus the unknown manifold structure. In addition, the basis vectors of $\mathbf{B}$ will capture the directions of variations within the class. Hence, the basis can be considered as tangent basis and the basis vectors as tangent vectors. Additionally, the vector $\boldsymbol{\theta}^*$ determines the local coordinates for a given data point.

The tangent distance computes the shortest distance from a given point to the affine subspace (the tangent space). Therefore, the distance estimates a kind of approximation error between the given data point and the best approximation. This best approximation $\mathbf{x}^*$ of a point $\mathbf{x}$ is determined by

$$\begin{aligned}\mathbf{x}^* &= \mathbf{t} + \mathbf{B}\boldsymbol{\theta}^*, \\ &= \mathbf{t} + \mathbf{B}\mathbf{B}^{\mathrm{T}}(\mathbf{x} - \mathbf{t}). \end{aligned} \tag{3.22}$$

Using this approximation, the tangent distance, see Equation (3.3), becomes equal to the Euclidean distance between the given point and the best approximation:

$$\mathsf{d}(\mathbf{x}, \mathfrak{w}) = d_E(\mathbf{x}, \mathbf{x}^*). \tag{3.23}$$

In Figure 3.1, this approximation is accurate as long as the data points are close to the set-prototype. Moreover, in this case, the tangent distance returns a much more accurate dissimilarity measure than the frequently used Euclidean distance. However, if points are not well approximated by $\mathfrak{w}$, then the tangent distance and the Euclidean distance often return similar values (e. g., the point $\mathbf{x}$ in the figure).

GTLVQ uses affine subspaces as set-prototypes. Thus, the method implicitly assumes that all tangent space approximations are globally valid. In theory, this could lead to difficulties in distinguishing different classes. For example, suppose

[13] A bold symbol like $\mathbf{t}$ refers to the embedded manifold point t of $\mathcal{M}$.

we classify handwritten digits and learned a tangent that perfectly approximates rotations of a six. If we rotate a six by 180 degrees, then the six is similar to a nine and, hence, the method would have problems to distinguish between sixes and nines. In practice, however, we learn the set-prototypes and, thus, the tangent space approximations discriminatively so that the method only learns tangents that support the class discrimination. Consequently, tangents that lead to confusion between classes are unlikely to be learned—this is a result of the learning rules because they pull and push the set-prototypes.

Relation to restricted-GTLVQ

Restricted-GTLVQ is a generalization of GTLVQ and, hence, almost all the statements made for GTLVQ remain true. The difference is in the definition of the set-prototypes and their relation to tangent space approximations. In Figure 3.1, this difference is highlighted by the boundaries (small right-angled strokes) on the set-prototype $\mathfrak{w}(\boldsymbol{\theta})$ so that the set-prototype becomes a line segment and thus an n_s-orthotope.

The approximation of the manifold can be derived in terms of a Taylor series centered at $\mathbf{t}$. Particularly, the tangent spaces are the first-order approximations of the manifold at point $\mathbf{t}$. As usual with a Taylor series approximation, the approximation is only accurate in a certain vicinity around the center point (i. e., around $\mathbf{t}$). Therefore, the approximation is less accurate for points that are far from the center point. As already mentioned, we implicitly assume through the affine subspaces in the standard GTLVQ that this approximation is globally valid, that is, for all $\boldsymbol{\theta} \in \mathbb{R}^{n_s}$. In contrast, restricted-GTLVQ applies a constraint on $\boldsymbol{\theta}$ and, consequently, learns indirectly that the approximation is only valid in a certain range around the center point—this range is described by the hyperrectangle $\mathcal{R}$.

The idea to interpret the tangent space approximation as a Taylor series was used to study similarities between transfer learning realized by LVQ methods and both GTLVQ and restricted-GTLVQ. A detailed description can be found in [2017c].

3.3.3 Generalized matrix learning vector quantization

Considering the tangent distance, see Equation (3.5), of GTLVQ and the quadratic-dissimilarity, see Equation (2.5), of GMLVQ, then it is obvious that the methods are similar to some extent—especially, in the local version of GMLVQ where each prototype $\mathbf{w}_k$ is equipped with an individual transformation matrix $\mathbf{Q}_k \in \mathbb{R}^{m_x \times n_x}$. Both methods transform the data point as well as the prototype or translation vector before calculating the dissimilarity by the Euclidean distance. However, local-GMLVQ applies an arbitrary transformation to an m_x-dimensional space, while GTLVQ applies a projection onto a linear subspace and, hence, stays in the input space $\mathbb{R}^{n_x}$.

The methods become even more similar if we use local-GMLVQ with a limited rank $m_x \leq n_x$ or, moreover, with a limited rank equivalent to the dimension of the complementary subspace—that is, m_x equals $n_x - n_s$. Then, the projector $\mathbf{P} \in \mathbb{R}^{n_x \times n_x}$ has the rank m_x and the transformation matrix $\mathbf{Q}$ also has the rank m_x, assuming the full rank of $\mathbf{Q}$—this is always assumed in the following discussion. Now, if we write the quadratic-dissimilarity as

$$d_Q(\mathbf{x}, \mathbf{w}) = \sqrt{(\mathbf{x} - \mathbf{w})^{\mathrm{T}} \boldsymbol{\Lambda} (\mathbf{x} - \mathbf{w})},$$

where $\boldsymbol{\Lambda} = \mathbf{Q}^{\mathrm{T}}\mathbf{Q} \in \mathbb{R}^{n_x \times n_x}$,[14] and the tangent distance as

$$\mathsf{d}(\mathbf{x}, \boldsymbol{\mathfrak{w}}) = \sqrt{(\mathbf{x} - \mathbf{t})^{\mathrm{T}} \mathbf{P} (\mathbf{x} - \mathbf{t})},$$

then the dissimilarity measures appear to be completely equivalent but differ in the properties of the matrix: $\boldsymbol{\Lambda}$ is an arbitrary transformation matrix, while $\mathbf{P}$ is an orthogonal projector.[15]

Even if GMLVQ could also learn a projector, it will generally not do so. Without regularization, GMLVQ will oversimplify to a few dimensions and thus map the data points onto a low-dimensional subspace. According to Schneider et al. (2010), oversimplification is “an overly pronounced elimination of dimensions in [the] feature space [that] can have negative effects on the performance and may lead to instabilities in the training” (p. 831). They also note that “the computation of the distance values is finally based on a strongly reduced number of features compared to the original input dimensionality of the data” (p. 833). This effect is sometimes called *collapsing dimensions* and was examined by Xing et al. (2003) and Globerson and Roweis (2006) for quadratic-dissimilarities and by Biehl et al. (2015) intensively for GMLVQ.

In Schneider et al. (2010), the authors proposed a regularization method for GMLVQ to control the oversimplification based on a measure of uniformity of the eigenvalue profile of $\boldsymbol{\Lambda}$ under the assumption that $m_x \leq n_x$.[16] Since $\boldsymbol{\Lambda}$ is the Gram matrix of $\mathbf{Q}$, the matrix is positive semidefinite and, therefore, we can find a decomposition of the form

$$\boldsymbol{\Lambda} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}}$$

[14]Note that rank $(\boldsymbol{\Lambda})$ equals rank $(\mathbf{Q})$.

[15]This strong relation is the reason why the learning rule for the translation of the affine subspace is equivalent to the learning equation for the prototype vector in GMLVQ.

[16]Similar statements can be derived for the case $m_x > n_x$.

by computing a singular value decomposition. Because we assume that the rank of $\mathbf{Q}$ equals m_x, the matrix $\mathbf{\Lambda}$ has exactly m_x nonzero singular values, and a compact singular value decomposition with $\mathbf{U} \in \mathbb{R}^{n_x \times m_x}$ and $\mathbf{\Sigma} \in \mathbb{R}_{\geq 0}^{m_x \times m_x}$ yields

$$\mathbf{\Lambda} = \underbrace{\left(\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\right)}_{\tilde{\mathbf{Q}}^{\mathrm{T}}} \left(\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\right)^{\mathrm{T}}.$$

Thus, we can find a decomposition of $\mathbf{\Lambda}$ as a function of the transformation matrix $\tilde{\mathbf{Q}}$. The matrix $\mathbf{U}$ is an orthonormal basis and it follows immediately that the column vectors of $\mathbf{U}$ are the eigenvectors to nonzero eigenvalues of $\mathbf{\Lambda}$. Additionally, the singular values in $\mathbf{\Sigma}$ are the nonzero eigenvalues.

If the transformation matrix $\mathbf{Q}^{\mathrm{T}}$ of GMLVQ is considered as a basis representation of the feature space, then $\tilde{\mathbf{Q}}^{\mathrm{T}}$ provides a basis representation where the basis vectors (column vectors) are orthogonal. In general, the basis $\tilde{\mathbf{Q}}^{\mathrm{T}}$ is not orthonormal because the basis vectors are eigenvectors scaled with respect to the square root of the corresponding eigenvalue. Exactly this scaling is responsible for the oversimplification effect as it scales the dimensions individually and, consequently, could lead to the effect that dimensions are overemphasized or almost squashed to zero vectors. Therefore, to control the oversimplification effect, we regularize the eigenvalue profile of nonzero eigenvalues of $\mathbf{\Lambda}$ such that it becomes more uniform.

A direct optimization of the eigenvalues is difficult, so we take advantage of the property that the product over the m_x nonzero eigenvalues λ_i is equal to the determinant of $\mathbf{Q}\mathbf{Q}^{\mathrm{T}}$:[17]

$$\det\left(\mathbf{Q}\mathbf{Q}^{\mathrm{T}}\right) = \prod_{i=1}^{m_x} \lambda_i.$$

If the trace of the matrix $\mathbf{\Lambda}$ is equal to m_x, then the sum over the nonzero eigenvalues λ_i is also equal to m_x:

$$\mathrm{tr}\left(\mathbf{\Lambda}\right) = \sum_{i=1}^{m_x} \lambda_i = m_x.$$

If we apply this trace condition as a constraint—which is assumed in the following—then the determinant of the matrix $\mathbf{Q}\mathbf{Q}^{\mathrm{T}}$ is maximized if and only if all $\lambda_i = 1$.[18]

[17]Eigenvalues are counted with respect to the algebraic multiplicity.

[18]As explained in Section 2.2.3 and, in particular, in Footnote 11, such a constraint is often used in GMLVQ to avoid the degeneration of single entries in the matrix $\mathbf{\Lambda}$. The constraint to $\mathrm{tr}\left(\mathbf{\Lambda}\right) = m_x$ is applied in the form of a projected gradient descent approach by applying the transformation

$$\mathbf{\Lambda} \longleftarrow m_x \frac{\mathbf{\Lambda}}{\mathrm{tr}\left(\mathbf{\Lambda}\right)}$$

after each matrix update.

Finally, a regularization term that avoids oversimplification is obtained by

$$-\alpha \ln\left(\det\left(\mathbf{Q}\mathbf{Q}^{\mathrm{T}}\right)\right),$$

where $\alpha > 0$ is the *regularization parameter* that controls the impact of the regularization term on the training. The gradient of the regularization term with respect to $\mathbf{Q}$ is $-2\alpha\left(\mathbf{Q}^{\dagger}\right)^{\mathrm{T}}$, where $\mathbf{Q}^{\dagger}$ is the Moore–Penrose pseudoinverse of $\mathbf{Q}$. Note that the regularization term is greater than or equal to zero.

In view of the previous discussion, this regularization term penalizes differences between the nonzero eigenvalues and is zero if and only if all nonzero eigenvalues are equal to one. If the term is zero, then the matrix $\mathbf{\Lambda}$ is an orthogonal projector because it is symmetric and idempotent. To show that $\mathbf{\Lambda}$ is idempotent, consider the following proof:

$$\begin{aligned}\mathbf{\Lambda}\mathbf{\Lambda} &= \tilde{\mathbf{Q}}^{\mathrm{T}}\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^{\mathrm{T}}\tilde{\mathbf{Q}},\\ &= \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{U}^{\mathrm{T}},\\ &= \mathbf{U}\mathbf{I}_{m_x}\mathbf{U}^{\mathrm{T}},\\ &= \mathbf{\Lambda}.\end{aligned}$$

Additionally, the transformation matrix $\mathbf{Q}^{\mathrm{T}}$ has to be an orthonormal basis because $\mathbf{\Lambda}$ equals $\mathbf{Q}^{\mathrm{T}}\mathbf{Q}$ and $\mathbf{U}\mathbf{U}^{\mathrm{T}}$. Finally, this implies that the regularization term is zero if and only if $\mathbf{Q}^{\mathrm{T}}$ is an orthonormal base matrix. Therefore, if we add the regularization term for each transformation matrix $\mathbf{Q}_k$ with an arbitrarily high regularization parameter α, then the local-GMLVQ algorithm becomes equivalent to the GTLVQ algorithm. The implicit affine subspace dimension n_s of the set-prototypes associated with local-GMLVQ is given by $n_x - m_x$ and the dimension of the complement by m_x.

In summary, we can say that GTLVQ is not affected by the oversimplification effect and is more efficient in terms of using parameters. GTLVQ models a set-prototype by $n_x\left(n_s + 1\right)$ parameters. However, local-GMLVQ requires $n_x\left(n_x - n_s + 1\right)$ parameters to implicitly learn a set-prototype of the same dimension. Thus, as long as we assume that the intrinsic dimension of the dataset is small and we model the data by affine subspaces with a dimension $n_s \ll n_x$, GTLVQ uses fewer parameters. Furthermore, the relation to GTLVQ makes clear that the transformation $\mathbf{Q}$ tends to map the dimensions that are discriminative for the classification task to $\mathbb{R}^{m_x}$—this is similar to the complement of an affine subspace learned by GTLVQ. Additionally, it maps the dimensions in which the dataset is invariant for the classification task to the kernel (null space) of $\mathbf{Q}$ with the dimension $n_x - m_x$—this is similar to an affine subspace learned by GTLVQ.

3.4 Accuracy and interpretability evaluations

This section presents a numerical evaluation of the derived GTLVQ and restricted-GTLVQ approach. We concentrate on the interpretability properties and the comparison of GTLVQ with other LVQ methods. In the first part of the experiments, we use two widely used toy datasets: CIRCLE and SPIRAL. These datasets are two-dimensional and, therefore, useful to visualize the basic behavior of GTLVQ and restricted-GTLVQ. Subsequently, we analyze GTLVQ on two real-world datasets: MNIST and INDIAN PINE.

In the following, all networks use the squared Euclidean distance as dissimilarity or underlying dissimilarity measure. Moreover, unless otherwise specified, they were trained with the identity activation function (i. e., $\phi(x) = x$) for the GLVQ loss according to Equation (2.15). The used optimizer was the Adam method of Kingma and Ba (2015) with the default settings from KERAS. During training, we monitored the validation loss and automatically adjusted the learning rate accordingly. If the validation loss has not decreased over five epochs, we have reduced the learning rate by a factor of 0.5. Furthermore, we trained without a box-constraint on the prototypes: If the dataset consists of images and the image space is defined over the unit interval, then we have not restricted the prototypes to this space during training.

3.4.1 Toy datasets

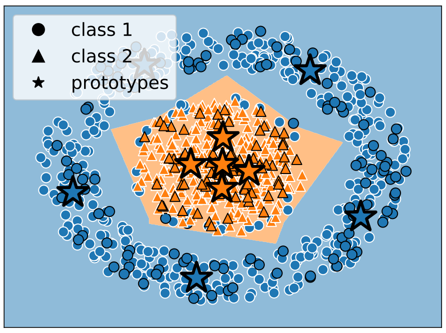
The analysis of toy datasets is a common approach to study basic properties of ML methods. They are helpful to give users a first impression of how the methods work and also to get a feeling for certain hyperparameters. For NNs, there are several online tutorials for this purpose—one widely used is TENSORFLOW PLAYGROUND.[19] This web interface provides access to several two-dimensional toy datasets and an interactive interface for the definition and training of NNs with the aim to give newcomers an impression on how NNs work. We took from this tutorial the CIRCLE and the SPIRAL dataset—two highly nonlinear datasets—to evaluate the following three LVQ methods.[20]

GLVQ: The GLVQ algorithm was trained without constraints or regularizations. We initialized the prototypes by computing class-wise a k-means, where the number of means was equal to the number of prototypes in the respective class.

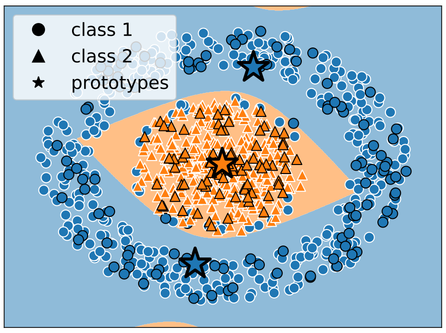
Local-GMLVQ: The transformation matrices $\mathbf{Q}_k$ are defined to be of size 2×2. At the beginning of the training, we initialized the prototypes by calculating

[19] `https://playground.tensorflow.org`

[20] We want to motivate the reader to visit TENSORFLOW PLAYGROUND to find NN architectures that perform on the same level as the LVQ methods on the two datasets used.



(a) GLVQ after training.



(b) Local-GMLVQ after training.

Figure 3.2: Visualization of the trained GLVQ and local-GMLVQ model on the CIRCLE dataset. The training samples (white border) and test samples (black border) of the classes are color-coded, and the whole input spaces are color-coded regarding the class assignments. Therefore, the visible boundaries represent the decision boundaries. The corresponding prototypes are drawn as big color-coded stars.

class-wise a k-means—similar to GLVQ—and initialized each matrix as $\frac{1}{2}\mathbf{I}_2$ (i. e., as a scaled identity matrix). During training, we normalized the trace of the matrices to one after each update step, as discussed in Section 2.2.3.

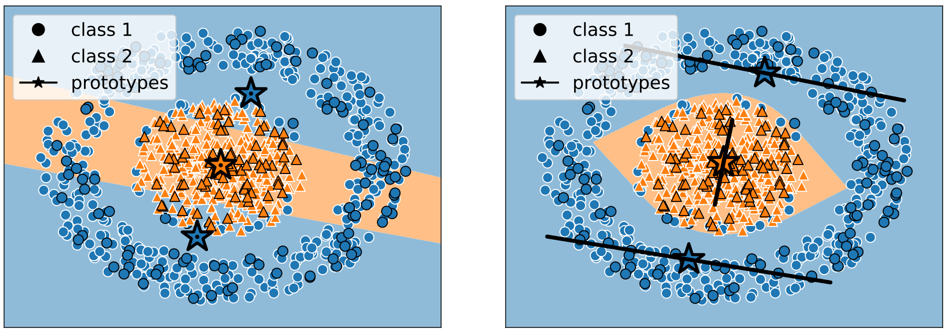
Restricted-GTLVQ: The set-prototypes are defined to be line segments. Hence, we used one-dimensional affine subspaces. The orthogonal normalization of the matrices and the initialization were performed as described in Section 3.2.2.

The Adam optimizer was used with an initial learning rate of 0.001 and a batch size of 24. We trained all networks until they converged, which always happened in less than 200 epochs. The datasets were randomly split into 80 % training and 20 % test data.[21] Moreover, we have not applied a normalization method to the datasets.

CIRCLE dataset

The CIRCLE dataset consists of an inner and an outer circle. Each circle corresponds to one class. Thus, the dataset is a binary classification problem. We generated 500 data points with a noise level of 10 for each class of the dataset—see Figure 3.2 for a visualization of the dataset. GLVQ is used with five prototypes per class and the

[21]Because we have no training-validation-test split, the validation and test data are identical. Therefore, the validation loss or accuracy is the same as the test loss or accuracy, respectively.



(a) Restricted-GTLVQ after initialization. (b) Restricted-GTLVQ after training.

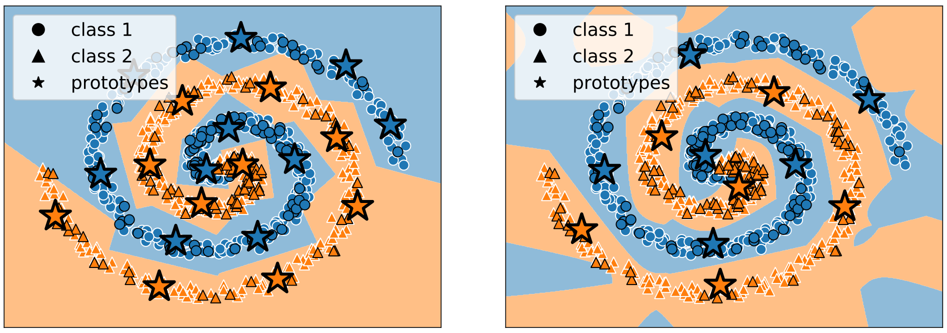
Figure 3.3: Visualization of the initialized and trained restricted-GTLVQ model on the CIRCLE dataset. The figure description is the same as in Figure 3.2 except for the prototypes: A set-prototype is represented by the translation point $\mathbf{t}_k$ (marked by the star) and the affine subspace (the black line segment). Note that after the initialization, the set-prototypes are close to prototype vectors, which is the reason for the tiny line segment (black dot) in the middle of each star.

other two LVQ methods with only one prototype for the inner circle and only two prototypes for the outer circle.

The training of the models is stable and the final test accuracy for each method is around 99 %. The training accuracy is significantly lower (around 96 %) for all methods because the dataset has an overlap of the two classes at the inner circle. However, almost none of the test points are in this overlap, so this difficulty is not reflected in the test accuracy. It is noteworthy that despite this overlap, the training results are not affected by this since the models do not show a tendency to overfit to these points.

Considering Figure 3.2, we can visually inspect the training results from GLVQ. GLVQ with five prototypes per class approximates the decision boundary with a polygon. This polygon is non-smooth but has no artifacts in the sense that regions outside the outer circle are labeled as the inner circle.

Local-GMLVQ learns a similar decision boundary as restricted-GTLVQ, see Figure 3.2 and Figure 3.3, respectively. Therefore, it seems that local-GMLVQ naturally converges to restricted-GTLVQ on this highly nonlinear dataset. However, the classified input space has unwanted artifacts—note the two spots of the inner circle class at the top and bottom of the image. The nonlinear decision boundary is the result of the local transformation matrices applied before the Euclidean distance measurement.



(a) GLVQ after training. (b) Local-GMLVQ after training.

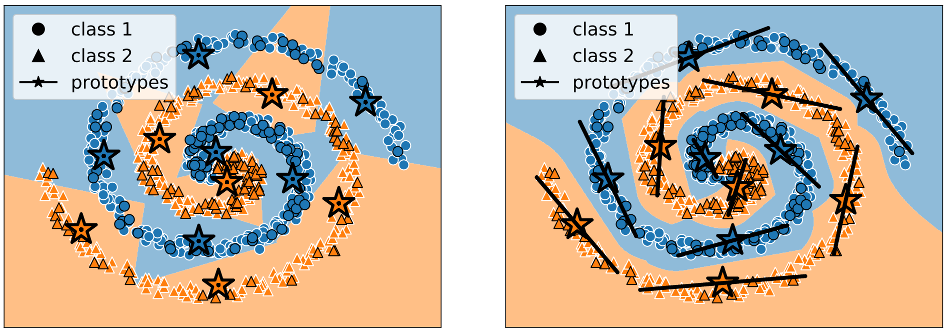
Figure 3.4: Visualization of the trained GLVQ and local-GMLVQ model on the Spiral dataset. The figure description is the same as in Figure 3.2.

In addition to the result of restricted-GTLVQ after training, Figure 3.3 shows the decision boundary after the initialization of restricted-GTLVQ. Because we initialized the parameter vectors $\mathbf{a}_k$ of the axis-aligned hyperrectangles $\mathcal{R}_k$ by 10^{-7} for each value and each set-prototype $\mathfrak{w}_k$, the one-dimensional orthotopes are close to prototype vectors. After the initialization, the orthotopes grow during training and classify the dataset by a fairly smooth decision boundary. Unlike local-GMLVQ, the classified input space has no artifacts. Due to the properties of restricted-GTLVQ, this is true for the entire input space: The decision boundary is determined by the points that have at least two closest set-prototype of different classes at the same time. Thus, if we consider a point on the decision boundary, the shortest distances from that point to at least two set-prototypes of different classes are equal. Consequently, the learned restricted-GTLVQ model cannot have artifacts like the learned local-GMLVQ model.

Spiral dataset

This dataset is motivated by the equation of the Archimedean spiral and consists of two spiral arms. The binary classification task is to classify the two arms and is, therefore, a more difficult task than the Circle dataset. Similar to the Circle dataset, we generated 500 data points for each class but with a noise level of 25—see Figure 3.4 for a visualization of the dataset. GLVQ is used with 10 prototypes per class and the other two LVQ methods with only 6 prototypes per class.

In contrast to the Circle dataset, the test accuracies of the algorithms differ. Restricted-GTLVQ achieves the highest accuracy of almost 99 %—the incorrectly



(a) Restricted-GTLVQ after initialization. (b) Restricted-GTLVQ after training.

Figure 3.5: Visualization of the initialized and trained restricted-GTLVQ model on the Spiral dataset. The figure description is the same as in Figure 3.3.

classified samples are located in the middle of the spiral arms. Local-GMLVQ has a test accuracy of 97 % and GLVQ of 96 %. The training of all methods was stable and converged consistently to similar results.

The classification results of GLVQ are comparable with the results on the Circle dataset. GLVQ tries to approximate the decision boundary by a polygon. However, due to the limited number of prototype vectors, the decision boundary is zigzagged, see Figure 3.4.

Again, local-GMLVQ and restricted-GTLVQ classify the dataset similarly—see Figure 3.4 and Figure 3.5, respectively. Especially in the middle of the spiral arms, the learned decision boundaries are very similar. In general, however, the classified input space of the local-GMLVQ model contains many artifacts and is littered with alternating class assignments—this is similar to the results of NNs on this dataset.[22]

Unlike the behavior of local-GMLVQ, restricted-GTLVQ classifies the Spiral dataset with a suitable approximation of the data manifold, see Figure 3.5. The line segments approximate the manifold structure of the data and take into account the vicinity in which the approximation is valid, see Section 3.3.2 for a theoretical discussion. At the beginning of the training, restricted-GTLVQ has the lowest accuracy between all methods. After the line segments have grown to an appropriate size during training, restricted-GTLVQ achieves the highest accuracy and provides the smoothest approximation of the decision boundary.

[22]We want to encourage the reader once again to visit Tensorflow Playground in order to try to train an NN to a similar result as restricted-GTLVQ.

Summary

The presented results for the two toy datasets show the basic classification principle of restricted-GTLVQ. As it was theoretically forecasted in Section 3.3.3, this could be similar to local-GMLVQ. However, restricted-GTLVQ is explicitly regularized and produces smoother decision boundaries compared to local-GMLVQ, which is also reflected in the visual results.

The training of restricted-GTLVQ was slower than all the other methods: GLVQ and local-GMLVQ converged within 50 to 100 epochs, whereas restricted-GTLVQ needed 200. This is because we initialized the method with 10^{-7} for the bounds of the centered and axis-aligned hyperrectangles. Normally, such a small initialization value is not necessary. If we increase the value to 0.1, restricted-GTLVQ has the same learning behavior and converges almost as fast as local-GMLVQ.

The application of the standard GTLVQ algorithm on these two-dimensional toy datasets is not appropriate as the required minimum subspace dimension is one. This means that the set-prototypes are lines and not line segments. These lines will likely cover regions of the other class, and therefore the GTLVQ algorithm is not applicable here. In [2016c] and [2017c], we solved this problem by using prior knowledge to mix set-prototypes and prototype vectors in a targeted manner. Even though this "trick" has solved the problem of GTLVQ on two-dimensional toy datasets, it seems to be an unrealistic approach for real-world tasks. For this reason, we have not used GTLVQ with such a parameterization in this evaluation. Nonetheless, the described and presented results about the alignment of the orthotopes also apply to the affine subspaces of GTLVQ, with the difference that they do not learn a vicinity in which the approximation is valid.

The application of the standard GMLVQ method is also not appropriate because the dataset is so nonlinear that the learned linear transformation cannot produce an improvement. More precisely, the algorithm learns an identity matrix and thus becomes equivalent to GLVQ. Consequently, we skipped the presentation of numerical GMLVQ results and used the local-GMLVQ algorithm instead.

3.4.2 Real-world datasets

We took two real-world datasets to evaluate the performance of GTLVQ: MNIST and Indian Pine. MNIST is an image dataset created by LeCun, Bottou, Bengio, and Haffner (1998),[23] and Indian Pine is a spectral dataset created by Baumgardner, Biehl, and Landgrebe (2015). In the first experiment on MNIST, we show how the subspace dimension of the affine subspace prototypes can be estimated. Then, we

[23]LeCun, Y., Cortes, C., & Burges, C. J. C. (n.d.). The MNIST database of handwritten digits. Retrieved from `http://yann.lecun.com/exdb/mnist/`

compare the performance of GTLVQ with other LVQ methods on this dataset. The purpose of this evaluation is to demonstrate that GTLVQ consistently outperforms the other LVQ variants while being interpretable. After that, we present how the proposed interpretation techniques from the image dataset MNIST are applied to the spectral dataset Indian Pine. During this experiment, we do not evaluate other methods since the objective of this experiment is to demonstrate the transferability of interpretation properties.

We have not applied a feature extraction method to the datasets and hence trained all LVQ algorithms on the raw data. Therefore, we do not compare the methods with NNs in terms of accuracy because they will consistently outperform LVQ if the NN architecture is appropriate. However, we refer to Section 3.5, where we evaluated an NN on MNIST regarding robustness against adversarial attacks, and to the evaluations in Chapter 4. For further evaluations, see [2016b] for results on a face recognition dataset and [2016c] for results on another spectral dataset.

MNIST dataset: Subspace dimension estimation

As already mentioned, for this experiment, we used the MNIST[24] dataset, which is one of the most studied datasets in ML. MNIST is a 10-class image classification task of handwritten digits, see Figure 3.7 for instance images. Each image has a size of 28×28 pixels, is given in grayscale, is centered, and is size normalized. Additionally, the images do not contain background noise—all digits are placed with white color on a clear black background. The official dataset consists of 60 k training images and 10 k test images. All images are 8 bit coded, so we first transformed each image value into the unit interval before we reshaped the images into vectors. Consequently, the input space of the data is defined by $[0,1]^{28 \cdot 28}$.

In Section 3.2.1, we proposed to estimate the subspace dimension of the affine subspace prototypes by incrementally increasing the subspace dimension while monitoring the training accuracy. Instead of training the model for each subspace dimension, we simply used the initialization procedure to get an accuracy estimate—this keeps the estimation of the subspace dimension fast. The used initialization procedure was the proposed strategy consisting of a k-means algorithm followed by a singular value decomposition.

We used GTLVQ with one prototype per class and repeated each initialization three times. The initialization was performed over 10 k random samples of the training dataset and we evaluated the initialized GTLVQ network over the entire training and test dataset. We stopped the experiment at a subspace dimension of 50.

[24]Modified National Institute of Standards and Technology.

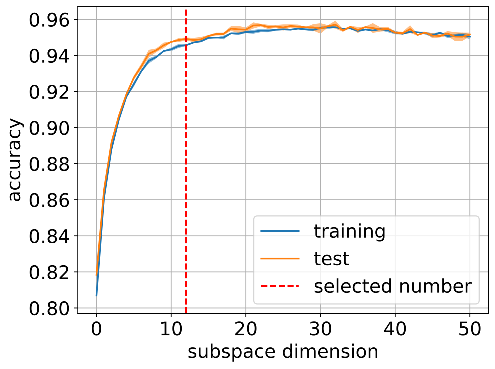


Figure 3.6: Estimation of the subspace dimension for GTLVQ by the proposed technique on the MNIST dataset. The plot shows the training and test accuracy over the subspace dimension. We calculated the means (solid lines) and the standard deviations (shaded regions) over three runs. The vertical line indicates the selected subspace dimension for the MNIST experiments. A subspace dimension of zero corresponds to prototype vectors—thus, the resulting model is a GLVQ network.

In Figure 3.6, we plotted the training and test accuracy after the initialization procedure over the subspace dimension. As can be seen, there is a saturation point around a subspace dimension of 30. However, we decided to choose a subspace dimension of 12 (12-dimensional affine subspaces, 12-dimensional tangent spaces) for the following experiments in order to keep the computational effort low and to be comparable with earlier experiments on tangent distances (e. g., Hastie et al., 1995). It is noteworthy that although we initialized the method with only 10 k samples from the training dataset, the method generalized well to the test dataset and achieved high test accuracies. This indicates that the proposed initialization strategy for GTLVQ is *extremely* efficient.

MNIST dataset: Accuracy and interpretability evaluations

The style of the handwritten digits of a class of the MNIST dataset varies greatly, making it a challenging dataset for LVQ algorithms. To compare GTLVQ with other LVQ methods, we use the following models.

GLVQ: A GLVQ network with one prototype per class. The network was trained without constraints or regularizations. We initialized the prototypes by computing class-wise a k-means, where the number of means was equal to the number of prototypes in the respective class.

Table 3.1: Accuracies in percentage of the evaluated LVQ methods on MNIST.

	model type	GLVQ	GMLVQ	GTLVQ
train accuracy	one prototype	73.8 ± 0.33	81.3 ± 0.09	89.7 ± 0.12
	1 M parameter	93.5 ± 0.12	89 ± 0.07	95.6 ± 0.09
test accuracy	one prototype	83.3 ± 0.02	88.3 ± 0.05	94.5 ± 0.05
	1 M parameter	95.9 ± 0.15	92.8 ± 0.08	97.4 ± 0.1

GLVQ-1M: Identical to the GLVQ model but with 128 prototypes per class.

GMLVQ: A GMLVQ network with one prototype per class. The transformation matrix $\mathbf{Q}$ is defined to be of size $n_x \times n_x$. At the beginning of the training, we initialized the prototypes by calculating class-wise a k-means—similar to GLVQ—and initialized each matrix as $\frac{1}{n_x}\mathbf{I}_{n_x}$ (i. e., as a scaled identity matrix). During training, we normalized the trace of the matrix to one after each update step, as discussed in Section 2.2.3.

GMLVQ-1M: Identical to the GMLVQ model but with 49 prototypes per class.

GTLVQ: A GTLVQ network with one set-prototype per class. The set-prototypes are defined to be 12-dimensional affine subspaces. We initialized and constrained the network using the methods proposed in Section 3.2.1.

GTLVQ-1M: Identical to the GTLVQ model but with 10 set-prototypes per class.

The multiple prototype settings were selected such that each method is as close as possible to the limit of 1 M trainable parameters—the choice of the 1 M parameter limit was arbitrary. In addition, this configuration followed the idea of increasing the number of prototypes in all models but making them comparable in terms of the number of trainable parameters—each model should have the same degrees of freedom to adapt to the dataset.

All models were trained for 150 epochs with a batch size of 128 and an initial learning rate of 0.001. During training, we applied basic data augmentations in the form of random shifts of up to ± 2 pixels and random rotations of up to ± 15 degrees. Each model was trained three times and we calculated the mean and standard deviation of the training and test accuracies.

In Table 3.1, we collected the accuracy results of the evaluation. With one set-prototype per class, GTLVQ clearly surpasses all other LVQ methods. If we increase the number of prototype vectors, the accuracy of GLVQ approaches the accuracy of GTLVQ and exceeds the results of GMLVQ. The accuracy gain of GTLVQ with

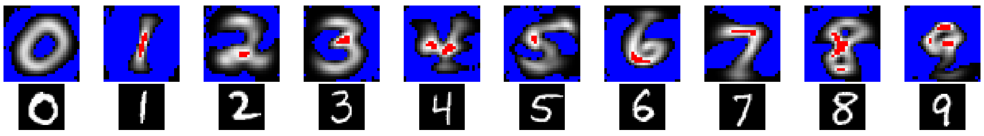


Figure 3.7: Class-specific prototypes learned by the GLVQ model on the MNIST dataset. The class is indicated by the MNIST digit below.

the increased number of set-prototypes is small (around 2 %). This can be explained by the already high accuracy of GTLVQ with only one set-prototype per class since this shows that the approximation of a class by a single tangent space is already sufficiently accurate. Therefore, the use of additional set-prototypes does not improve the generalizability of the model so much. If we add more prototypes in GLVQ, the prototype vectors will begin to approximate the data manifold and, thus, implicitly the set-prototypes used in GTLVQ. Consequently, the results of GLVQ with more prototypes per class will become similar to the results of GTLVQ.

Comparing the results of Table 3.1 with Figure 3.6, the training accuracy of GTLVQ appears to be less than the initialization accuracy. However, the initialization was performed without augmentation so that the accuracies are not comparable. Nevertheless, the comparison shows that the test accuracy after initialization and after training is almost the same (around 94.5 %), and therefore the model does not seem to have improved during training. But if we compare the validation loss after initialization and after training, we observe an improvement: The validation loss is -0.30 after the initialization and is -0.38 after the training (a similar result can be found for the GTLVQ-1M model). Therefore the relative distance difference has improved and with it the robustness and generalizability of the model—see also the results of Chapter 3.5.

To get an impression of the learned classifier function, we visualize the learned prototypes. However, they cannot be plotted directly as images because the prototypes are elements of $\mathbb{R}^{28\cdot 28}$. Therefore, we convert each prototype vector into an image using the following procedure:

1. We reshape the vector to the original image size of 28×28 pixels.

2. We replicate the matrix to a tensor of size $28 \times 28 \times 3$. Now, this is something like an RGB image but with values in $\mathbb{R}$.

3. We change the color of a pixel to blue if the intensity value is less than zero and to red if the intensity value is greater than one. This means that instead of simply clipping intensity values outside the intensity range (i. e., outside the

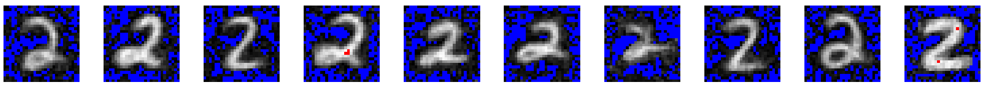


Figure 3.8: Randomly selected prototypes of the class 2 learned by the GMLVQ-1M model on the MNIST dataset.

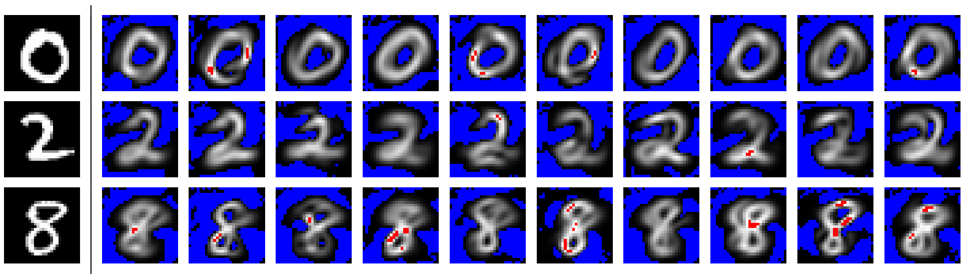


Figure 3.9: Randomly selected points from the learned affine subspaces of three different classes learned by the GTLVQ model on the MNIST dataset. The class of each set-prototype is indicated by the MNIST digit on the left.

interval $[0, 1]$), we color them to indicate the violation of the image space. The intensity values within the unit interval are not changed so that the image is displayed as a grayscale image at these pixel positions.

In Figure 3.7, we visualize the prototypes learned by the GLVQ model. The prototype vectors can be viewed as blurred images of real looking digits—compare the prototype images with the MNIST samples—and the violation of the image space restriction mostly occurs in the background. Especially, the fact that we can interpret the prototypes directly as digits suggests that the GLVQ model has learned what a digit looks like and that the classification decision is based on finding the most similar digit class. Additionally, we can discover biases in the MNIST dataset: MNIST is a collection of American handwritten digits, so the prototypes of the class 1 and the class 7 clearly show the American writing style.

The GMLVQ model learns prototypes similar to GLVQ: All prototypes look like blurred images of real digits. Unlike GLVQ, the GMLVQ prototypes do not consistently violate the image space restriction in the background. Similar to the images of the GMLVQ-1M model in Figure 3.8, the background shows a salt and pepper structure (i. e., some pixels violate the image space restriction and others do not). This behavior is caused by the learned transformation matrix, but it is not clear how this property is advantageously used to model the classification decision.

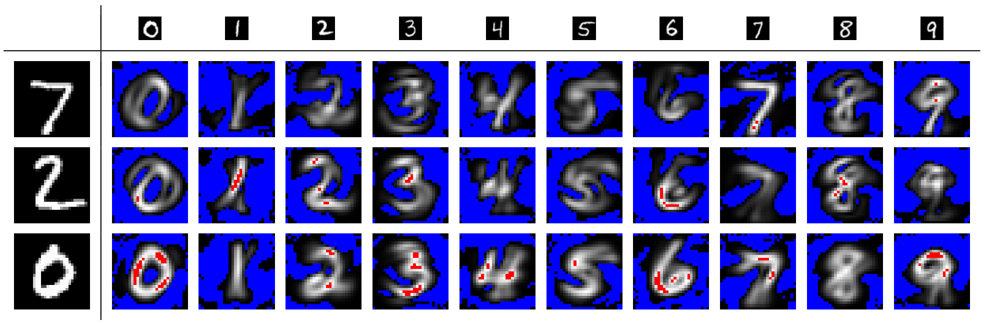
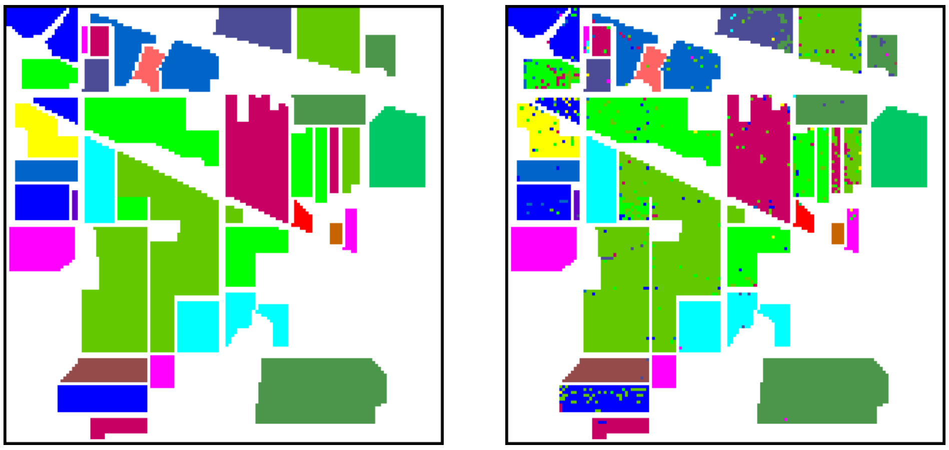


Figure 3.10: Best approximation for each set-prototype for three different input samples of the learned GTLVQ model on the MNIST dataset. The input samples are visualized in the left column and the class correspondences are indicated by the MNIST digits at the top.

Figure 3.8 shows some prototypes from the GMLVQ-1M model. The prototypes of the GLVQ-1M model look similar, and again the only difference is in the structure of the violations of the image space restriction. For both models, the prototype images resemble real digits of different writing styles. As opposed to the one-prototype models, the digits appear less blurry.

The GTLVQ models learn an infinite number of representations of the digits. Each point of a set-prototype is a possible realization of a prototype vector. So it is not enough to visualize a single prototype vector to illustrate what a set-prototype has learned from the dataset. However, to give an idea of what a set-prototype has learned about a class, we sample several points from each set-prototype and apply the visualization concept described for prototype vectors. We create a sample point according to $\mathbf{t} + \mathbf{B}\boldsymbol{\theta}$, where the parameters θ_i are drawn from a normal distribution with zero mean and unit standard deviation. Figure 3.9 shows sample points of the GTLVQ model for three different classes. Although the model learns only one set-prototype per class and the set-prototype model is linear, the variations in the image space appear nonlinear. For the class 0, the model learned to vary the circle style. The variations of the class 2 include different styles of the lower left circle, line end variations, and aspect ratios. Similar to the results of GLVQ and GMLVQ, the violation of the image space usually occurs in the background and all images appear blurred. Furthermore, some of the sampled images contain artifacts such as double lines, black spots, and so on and therefore resemble unreal digits.

In Figure 3.10, we present the best approximation of an input $\mathbf{x}$ for each set-prototype, which is the closest point to the given input—see Equation (3.22). For



(a) Ground truth of the dataset. (b) Predicted labels on the entire dataset.

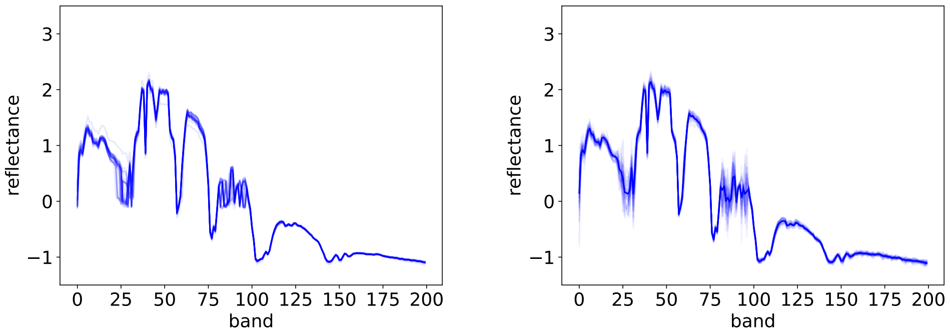
🟥 alfalfa	🟪 oats
🟩 corn-no-till	🟥 soybean-no-till
🟦 corn-min-till	🟩 soybean-min-till
🟨 corn-clean	🟦 soybean-clean
🟪 grass/pasture	🟫 wheat
🟦 grass/trees	🟩 woods
🟧 grass/pasture-mowed	🟦 buildings/grass/trees/drives
🟩 hay-windrowed	🟥 stone-steel towers

Figure 3.11: Ground truth and predicted results of the GTLVQ model on the Indian Pine dataset. The background is colored black.

the input of the seven, we see that the best approximation is realized by the set-prototype of the class `7`. Therefore, the prototype of this class has the shortest distance and the class `7` is the winning class. All the other approximations are not suitable. However, note how the diagonal line of the class `4` and `9` matches the diagonal line of the input. Similar interpretations are possible for the other inputs.

Indian Pine dataset: Transferability of interpretation properties

The Indian Pine dataset is a spectral dataset that was collected by an AVIRIS sensor over the area of northwestern Indiana. The 16 classes consist of agricultural crops, forests, and other natural vegetation that are not mutually exclusive, see Figure 3.11 for a visualization of the dataset. The original recording consists of 145×145 pixels (corresponding to an area of 2×2 miles), where each pixel is a 224 spectral reflectance band in the wavelength range of 0.4 to 2.5 nanometers. We used



(a) Input samples of the class. (b) Random samples of the set-prototype.

Figure 3.12: Visualization of the input samples and the learned set-prototype of the GTLVQ model on the INDIAN PINE dataset for the class `alfalfa`. We plotted all input samples of the class with a light blue color in the diagram so that a darker color indicates frequent spectra and a lighter color indicates rare spectra. The same visualization principle is applied to the set-prototypes using sampled random points. Additionally, the average value is plotted in dark blue.

the reduced dataset for the experiments in which the water-absorbing bands were removed. This dataset contains 200 instead of 224 reflectance bands. In general, the dataset is very unbalanced because the number of sample points varies between 20 (class `oats`) and 2 455 (class `soybean-min-till`). We generated the training and test dataset by a stratified random split into 80 % training and 20 % test samples.[25] Each sample was normalized to zero mean and unit standard deviation.

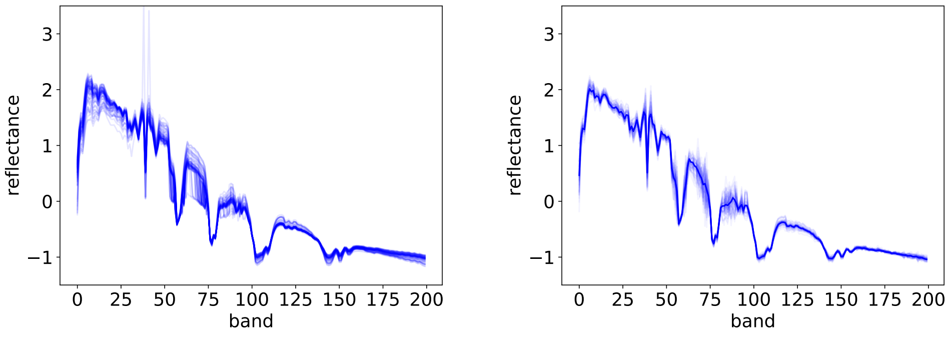
In this experiment, we used a GTLVQ model with a subspace dimension of 15. Moreover, we defined the number of prototypes per class as a function of the number of training samples in that class. In particular, the function was defined by

$$\text{ceiling}\left(\min\left\{\frac{\text{number\_of\_training\_samples\_in\_class\_}c}{100}, 5\right\}\right).$$

However, since the classes `corn-min-till` and `buildings/grass/trees/drives` have fewer variations, we excluded them and set the number of prototypes to four and two, respectively. The model was trained by optimizing the GLVQ loss with the squashing function $\phi(x) = \text{ReLU}(x + 0.3)$.[26] This loss function sets a relative

[25]This means that we split class-wise into training and test such that the class distribution is maintained.

[26]The *Rectified Linear Unit* (ReLU) activation is defined as $\text{ReLU}(x) = \max\{0, x\}$ and applies element-wise for vectors.



(a) Input samples of the class. (b) Random samples of the set-prototype.

Figure 3.13: Visualization of the input samples and the learned set-prototype of the GTLVQ model on the Indian Pine dataset for the class `stone-steel towers`. The figure description is the same as in Figure 3.12.

distance difference limit of −0.3 for correctly classified inputs. The training was performed with a batch size of 64, an initial learning rate of 0.001, and a training time of 20 epochs. In addition, the sampling of the batches during training was performed with respect to the class distribution, and GTLVQ was initialized and constrained by the proposed schemes, see Section 3.2.1.

The training and test accuracy of the trained GTLVQ model is $(97.96 \pm 0.25)\,\%$ and $(86.5 \pm 0.11)\,\%$, respectively. We illustrate the predicted output in Figure 3.11. Note that most confusions are between classes that are not mutually exclusive, such as `soybean-min-till` and `corn-min-till`.

In Figure 3.12 and Figure 3.13, we visualize the set-prototype and the corresponding input data for two classes. The visualization of a set-prototype is realized by the presentation of randomly sampled points in a diagram. In particular, we generated 50 random points by sampling θ_i from a normal distribution with a mean value of zero and a standard deviation of one third. For both classes, the set-prototypes show high variability in regions where the input data also show high variations, whereas they show low variability in regions with low variations. This suggests that the set-prototypes have learned variations that are useful to approximate the dataset.

Summary

In this section, we evaluated the performance of GTLVQ regarding interpretability and accuracy and showed how to estimate the subspace dimension required for

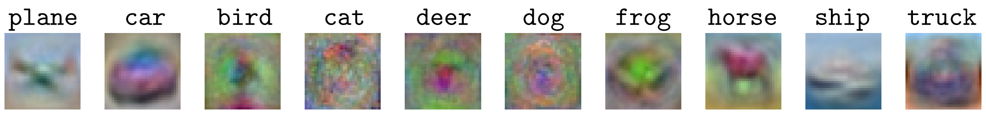


Figure 3.14: One random sample from each set-prototype of a GTLVQ model trained on CIFAR-10. The class is indicated by the name above.

GTLVQ. Additionally, we compared GTLVQ with GMLVQ and GLVQ as these methods are the most commonly used versions of LVQ. In general, GTLVQ outperforms both methods in terms of accuracy and maintains the interpretability at the same time.

We have not presented the evaluation of restricted-GTLVQ, because as long as the data dimension n_x is much greater than the subspace dimension n_s, restricted-GTLVQ has produced the same results as GTLVQ in all the experiments. However, due to the higher computational complexity, the training progress of an equivalent restricted-GTLVQ model is much slower than that of the corresponding GTLVQ model. If the data dimension is close to the subspace dimension, restricted-GTLVQ may have advantages over GTLVQ—see the evaluations on the toy datasets in Section 3.4.1, for instance.

Although GTLVQ can model variations within the data better than other LVQ variants, it is still not able to perfectly model general real-world variations of objects in images. For example, if we apply GTLVQ on the CIFAR-10 dataset, the test accuracy with one set-prototype per class and a subspace dimension of 12 is only around 52 %—see Section 4.5.6 for a description of the dataset. By examining the visualized sample points of the learned set-prototypes in Figure 3.14, the model shows that it is not able to model the variations appropriately.[27] These sample points are mostly like blurred color stains and not like images of real objects. However, some of the points highlight the general object shape of the class, such as the samples for the class `horse` and `car`, or they show the dominant colors, such as the sample for the class `ship`.

In summary, GTLVQ is inadequate if the dataset contains strong background noise or transformations that cannot be approximated by the set-prototypes and thus by affine subspaces. Therefore, such a case requires an appropriate feature extraction method that supports the class discrimination.

[27]The visualizations were created without clipping of the intensity values since the points were all elements of the image space.

3.5 Generalized tangent learning vector quantization as margin maximizer

This section discusses the margin maximization property of GTLVQ. In the first part, we discuss the theory of margin maximization from a theoretical point of view using the hypothesis margin. This theory is the basis for proving that LVQ methods are robust against adversarial attacks and that the hypothesis margin is a lower bound for adversarial perturbations. In the second part, we present an adversarial robustness evaluation of LVQ models on the MNIST dataset to support the theoretical results. By comparing the achieved robustness scores of GTLVQ with the hypothesis margin values, we show that the hypothesis margin is, indeed, a lower bound for adversarial perturbations. Additionally, we discuss several implications of these results, for instance, why GMLVQ is not robust against the evaluated attacks.

The presented robustness evaluation was partly published in [2019b]. In contrast to this version, the presented evaluation is performed with state-of-the-art attacks, and the robustness results are discussed from the perspective of the margin maximization theory of LVQ. However, this evaluation does not repeat the analysis of the NN architecture proposed by Madry, Makelov, Schmidt, Tsipras, and Vladu (2018)—this architecture is considered as one of the best in terms of robustness against adversarial attacks on the MNIST dataset. Additionally, we use a baseline NN consisting of more layers than in the previous evaluation in order to be comparable to the models used in Chapter 4. Note that the robustness of this model is less than the robustness of the shallow NN from [2019b]. Nevertheless, this is not too important since the overall results remain the same.

3.5.1 Theoretical analysis

In the following, we analyze the margin of LVQ and GTLVQ. First, we review the results of Crammer et al. (2003), who performed a margin analysis of LVQ. Second, we discuss how these results apply to GLVQ variants and, especially, to GTLVQ.

Margin analysis of LVQ algorithms

We summarize the results of the margin analysis by Crammer et al. (2003) analogously to the original publication, however, we adjust the mathematical notations accordingly and focus on the Euclidean distance. The mathematical proofs of this theory are not part of the original publication but are included in the supplementary material.[28] It should be noted that the formulations of the lemmas and the theorem

[28]The supplementary material is available at `https://www.cse.huji.ac.il/labs/learning/Papers/NNk_with_theory.ps.pdf`, for example.

differ slightly between the main body and the supplementary material of the publication. The lemmas and the theorem that we present here correspond to the versions of the supplementary material because the proofs make them comprehensibly correct. As this is a summary of the results of Crammer et al., we emphasize that the following descriptions are *highly related* to Crammer et al. (2003) and that all quotations also refer to this publication.

The margin of a classifier is roughly described as "margins measure the level of confidence a classifier has with respect to its decisions" (p. 479). The analysis of margins and their relation to the generalization error is a key concept of successful supervised ML algorithms (e. g., support vector machines). Thereby, the generalization error is the probability that an input is misclassified.

Support vector machines are analyzed by the so-called *sample margin*—sometimes just called margin (e. g., Schölkopf & Smola, 2002). That is the shortest dissimilarity (distance) of an input to the decision boundary induced by the classifier. This definition of margin is intuitive but not practical for LVQ algorithms. LVQ induces the decision boundaries implicitly by the prototypes and thus by the associated Voronoi tessellation. These induced decision boundaries are very sensitive to the position of the prototypes: A small displacement of the prototypes could lead to strong changes in the boundaries. Consequently, the use of the sample margin to analyze LVQ is inappropriate because it is numerically unstable and also difficult to calculate. Therefore, LVQ is analyzed by another margin definition, the so-called hypothesis margin.

Definition 3.6 (hypothesis margin)**.** Given a set of prototypes $\mathcal{W}$ and a dissimilarity d. The *hypothesis margin* of $\mathcal{W}$ with respect to a set $\mathcal{S}$ of inputs is the maximum radius r such that the following condition holds: If we define a ball[29] with radius r induced by d around each prototype, every change in the position of the prototypes within its ball does not change the class labels assigned to the inputs of $\mathcal{S}$. In symbols, we write $\text{margin}_h(\mathcal{S}, \mathcal{W})$.

This definition follows a description by Gilad-Bachrach[30] and is equivalent to the definition by Crammer et al. The advantage of this formulation, however, is that it does not require a countable set of prototypes, which is beneficial for applying the hypothesis margin to GTLVQ and restricted-GTLVQ. Also, note that this definition does *not* require labeled data points. Similarly to $\text{margin}_h(\mathcal{S}, \mathcal{W})$, we denote the sample margin by $\text{margin}_s(\mathcal{S}, \mathcal{W})$. The following lemma describes how the hypothesis margin can be calculated for the Euclidean distance and a single input.

[29] A ball is always considered *open*.

[30] Gilad-Bachrach, R. (2004, December 7). Two types of margins. Retrieved from `https://www.cse.huji.ac.il/labs/learning/code/feature_selection/tutorial/node3.html`

Lemma 3.4. *Let $d_E(\mathbf{x},\mathbf{w})$ be the Euclidean distance and $\mathbf{x} \in \mathbb{R}^{n_x}$ be an input. The hypothesis margin of $\mathcal{W}$ with respect to $\mathbf{x}$ is*

$$\operatorname{margin}_h(\{\mathbf{x}\},\mathcal{W}) = \frac{1}{2}\left(d_E(\mathbf{x},\mathbf{w}_j) - d_E(\mathbf{x},\mathbf{w}_i)\right),$$

where $\mathbf{w}_i$ is the closest prototype to $\mathbf{x}$ and $\mathbf{w}_j$ is the closest prototype to $\mathbf{x}$ with a different label than the label of $\mathbf{w}_i$.

This lemma provides a surprisingly simple equation for calculating the hypothesis margin regarding a single input. Additionally, we can prove the following lemma, which relates the hypothesis margin to the sample margin.

Lemma 3.5. *Let $d_E(\mathbf{x},\mathbf{w})$ be the Euclidean distance and $\mathcal{S}$ be a set of inputs. The hypothesis margin of $\mathcal{W}$ with respect to $\mathcal{S}$ is a lower bound of the sample margin of $\mathcal{W}$ with respect to $\mathcal{S}$:*

$$\operatorname{margin}_h(\mathcal{S},\mathcal{W}) \le \operatorname{margin}_s(\mathcal{S},\mathcal{W}).$$

This lemma states that if we find a prototype configuration with a large hypothesis margin, then the sample margin is large as well.

To formulate the theorem, we must define a signed version of the hypothesis margin that incorporates the class label of a training sample. We also use this definition to define the margin error: A training sample causes a margin error if it has a signed hypothesis margin that is less than a certain threshold.

Definition 3.7 (signed hypothesis margin and margin error)**.** Given a labeled input sample $(\mathbf{x}, c(\mathbf{x}))$ of a training dataset $\mathcal{T}$ with $\#\mathcal{T}$ training samples and a set of prototypes $\mathcal{W}$. The *signed hypothesis margin* of an input sample is

$$\operatorname{margin}_h^c(\mathbf{x}, c(\mathbf{x}), \mathcal{W}) = \begin{cases} \operatorname{margin}_h(\{\mathbf{x}\},\mathcal{W}) & \text{if } \mathbf{x} \text{ is correctly classified,} \\ -\operatorname{margin}_h(\{\mathbf{x}\},\mathcal{W}) & \text{otherwise.} \end{cases}$$

The *margin error* of $\mathcal{W}$ with respect to $\mathcal{T}$ and margin threshold $t > 0$ is

$$\operatorname{error}_h^t(\mathcal{T},\mathcal{W}) = \frac{\#\left\{(\mathbf{x}, c(\mathbf{x})) \in \mathcal{T} \mid \operatorname{margin}_h^c(\mathbf{x}, c(\mathbf{x}), \mathcal{W}) < t\right\}}{\#\mathcal{T}},$$

where # denotes the cardinality.

Note that the signed hypothesis margin is lower bounded by the absolute distance difference $\triangle d(\mathbf{x})$:

$$\begin{aligned} \triangle d(\mathbf{x}) &= d^-(\mathbf{x}) - d^+(\mathbf{x}), \\ &\le 2 \cdot \operatorname{margin}_h^c(\mathbf{x}, c(\mathbf{x}), \mathcal{W}), \end{aligned} \tag{3.24}$$

whereby the equality holds if $\mathbf{w}^{+}(\mathbf{x}) = \mathbf{w}_i$ or $\mathbf{w}^{+}(\mathbf{x}) = \mathbf{w}_j$ in accordance to the notation of Lemma 3.4. Also note the similarity between the absolute distance difference and the relative distance difference according to Equation (2.12).

Theorem 3.4. *Assume the following setting:*

- *Let $d_E(\mathbf{x}, \mathbf{w})$ be the Euclidean distance and $\mathcal{W}$ be a set of prototypes with p prototypes per class.*
- *Let $\mathcal{T}$ be a (training) dataset of labeled inputs drawn from some underlying distribution $\mathcal{D}$.*
- *For all the (training) samples $(\mathbf{x}, c(\mathbf{x})) \in \mathcal{T}$ holds $\|\mathbf{x}\|_E \leq R$ for some chosen constant $R \in \mathbb{R}_{>0}$.*
- *The selected margin threshold t is an element of the interval $(0, 0.5)$.*
- *Let $\mathrm{error}_{\mathcal{D}}(\mathcal{W})$ be the generalization error of the LVQ algorithm with respect to the distribution $\mathcal{D}$:*

$$\mathrm{error}_{\mathcal{D}}(\mathcal{W}) = P_{(\mathbf{x}, c(\mathbf{x})) \sim \mathcal{D}}\left(c(\mathbf{x}) \neq c(\mathbf{w}^{*})\right).$$

For every $\epsilon > 0$, with probability $1 - \epsilon$ over the choices of the training data

$$\mathrm{error}_{\mathcal{D}}(\mathcal{W}) \leq \mathrm{error}_h^t(\mathcal{T}, \mathcal{W}) + \underbrace{\sqrt{\frac{8}{\#\mathcal{T}}\left(n_{vc} \cdot \log^2 \frac{32 \cdot \#\mathcal{T}}{t^2} + \log \frac{4}{\epsilon}\right)}}_{\textit{upper bound of the generalization gap}},$$

where n_{vc} is the VC dimension (Vapnik–Chervonenkis dimension)

$$n_{vc} = 2 \cdot \min\left\{n_x + 1, \frac{64R^2}{t^2}\right\} \cdot p^{\#\mathcal{C}} \cdot \log\left(e \cdot p^2\right)$$

and e is Euler's number.

This theorem gives a bound for the generalization error by a sum of the margin error and an upper bound of the generalization gap. It can be used to characterize some important facts about LVQ methods: First of all, the bound does not directly depend on the input dimension n_x, but the VC dimension n_{vc} grows with the number p of prototypes per class. Second, the bound decreases if we reduce the margin error. This can be achieved by increasing the classification accuracy or by decreasing the margin threshold t. However, lowering the margin threshold can increase the VC dimension and, thus, the upper bound of the generalization gap and, eventually, the bound. Additionally, an increase in accuracy is usually achieved by using more prototypes, which also increases the upper bound of the generalization gap. Overall, this indicates that there is a nontrivial optimal number of prototypes and a nontrivial optimal margin threshold.

One implication of the nontrivial optimal number of prototypes is that an LVQ method could not only be faster than a k-nearest neighbors algorithm—in terms of computational speed due to dissimilarity calculations—it could also be more accurate as well. Once we have chosen the number of prototypes, we want to find a margin threshold t such that the margin error is small for a large t. However, these are contradictory goals because a large margin threshold causes a higher margin error. To overcome this, we define a corresponding loss function depending on the absolute distance difference $\triangle d(\mathbf{x})$ and indirectly optimize the margin threshold to a compromise. Since the absolute distance difference lower bounds the signed hypothesis margin and is equivalent for positive margins, see Equation (3.24), maximizing this expression within a proper loss function balances the two contradictory objectives and maximizes the hypothesis margin. This means that we generally weigh large margins of correctly classified samples against the margin error.

In summary, an LVQ algorithm is a margin maximizer if the loss function maximizes a function of the absolute distance difference $\triangle d(\mathbf{x})$. The choice of the loss function controls the trade-off between a large margin and a small margin error. Remarkably, the proofs of the lemmas are norm independent and only require the triangle inequality and absolute homogeneity of the norm. *Thus, the lemmas can be extended to homogeneous, translation-invariant semimetrics.*

Margin analysis of GLVQ algorithms

Now, we adapt the result from LVQ to GLVQ variants. As already mentioned, the theory of Crammer et al. states that LVQ is a margin maximizer if we maximize a loss function that depends on the absolute distance difference $\triangle d(\mathbf{x})$. This, of course, corresponds to minimizing a loss function based on $-\triangle d(\mathbf{x})$, and this expression corresponds to the numerator of the relative distance difference $\mu(\mathbf{x})$ according to Equation (2.12). Thus, the GLVQ loss function is based on a kind of margin value. Normally, however, GLVQ uses the *squared* Euclidean distance so that the norm assumption to proof the Lemma 3.4 is violated and the lemma cannot be applied directly. Therefore, the question arises how this GLVQ loss optimization is related to the margin maximization theory of LVQ.

If we consider the relative distance difference $\mu(\mathbf{x})$ and assume that we use the squared Euclidean distance, then we can rewrite $\mu(\mathbf{x})$ to

$$\begin{aligned}\mu(\mathbf{x}) &= \frac{d^{+}(\mathbf{x}) - d^{-}(\mathbf{x})}{d^{+}(\mathbf{x}) + d^{-}(\mathbf{x})}, \\ &= \frac{d_E^2(\mathbf{x}, \mathbf{w}^{+}) - d_E^2(\mathbf{x}, \mathbf{w}^{-})}{d_E^2(\mathbf{x}, \mathbf{w}^{+}) + d_E^2(\mathbf{x}, \mathbf{w}^{-})}, \\ &= -\triangle d(\mathbf{x}) \frac{d_E(\mathbf{x}, \mathbf{w}^{+}) + d_E(\mathbf{x}, \mathbf{w}^{-})}{d_E^2(\mathbf{x}, \mathbf{w}^{+}) + d_E^2(\mathbf{x}, \mathbf{w}^{-})}. \end{aligned} \tag{3.25}$$

Consequently, the GLVQ loss is based on the absolute distance difference $\triangle d(\mathbf{x})$ and, hence, the GLVQ method is a margin maximizer. The fraction is an input sample dependent scaling factor that controls the trade-off between a large margin value and a small margin error. If we use GLVQ with the Euclidean distance, then the relative distance difference $\mu(\mathbf{x})$ is equal to

$$\mu(\mathbf{x}) = -\triangle d(\mathbf{x}) \frac{1}{d_E(\mathbf{x}, \mathbf{w}^{+}) + d_E(\mathbf{x}, \mathbf{w}^{-})},$$

and GLVQ is again a margin maximizer. The difference between the two versions is that they train for different trade-offs. Overall, we can say that the GLVQ loss produces a margin maximization if the dissimilarity is induced by a norm. However, the identity of indiscernibles of a dissimilarity measure or the positive definiteness of a norm need not be valid. Therefore, for instance, GMLVQ is a margin maximizer with respect to d_Q.

How does this theory of margin maximization apply to GTLVQ? In GTLVQ, we are dealing with set-prototypes $\mathfrak{w}$ and a point-set dissimilarity d. To apply Definition 3.6 of the hypothesis margin, we consider all points from the set-prototypes as individual prototype vectors. Together with the underlying dissimilarity d_E of d, the hypothesis margin definition is applicable, although a GTLVQ network in this sense consists of an infinite number of prototypes.

The proofs for the lemmas presented by Crammer et al. (2003) are based on a countable set of prototypes. However, they can easily be extended to uncountable sets by replacing all index assignments of prototypes with a general set notation. Therefore, the two lemmas apply to GTLVQ, and we can state that GTLVQ is a margin maximizer with respect to the Euclidean distance. The same result holds for the restricted version of GTLVQ.

3.5.2 Experimental evaluation

As discussed in the introduction of this thesis, the robustness of NNs against adversarial attacks has become one of the most discussed topics in ML. By making almost imperceptible changes to the input of a classifier, attackers can force a misclassification of the input or even change the prediction to an arbitrary class. In this section, we show that GLVQ algorithms and, particularly, GTLVQ are very robust methods against adversarial attacks. Moreover, they are provably robust and the hypothesis margin provides a lower bound for the adversarial perturbation with respect to the Euclidean norm. The basis of this study is the relation between adversarial perturbations and the hypothesis margin.

The adversarial attacks and robustness measures used here are similar to those used by Schott, Rauber, Bethge, and Brendel (2019) with a few minor modifications in order to evaluate LVQ methods. The evaluation was performed with the MNIST dataset since it is one of the most frequently used datasets for robust model evaluations in the literature. Although it is considered by many to be a solved "toy dataset" that achieves almost perfect classification accuracy with state-of-the-art NNs, the defense against adversarial attacks on MNIST is anything but trivial (Schott et al., 2019). To demonstrate this, we evaluate an NN and, further, use it for comparisons with the LVQ models.

The structure of this section is as follows: First, we introduce a commonly used definition of an adversarial attack. Second, we define the robustness measures that are used to quantify the robustness. After that, we describe the evaluation setup and models before presenting and discussing the results.

Adversarial attacks

Given a classification task and a labeled input sample $(\mathbf{x}, c(\mathbf{x}))$, an *adversarial example* $\tilde{\mathbf{x}}$ of the sample $\mathbf{x}$ is defined as the smallest required perturbation of $\mathbf{x}$ by $\boldsymbol{\epsilon}$ to find a point on the decision boundary or in the classification region of a different class than $c(\mathbf{x})$:

$$\min_{\boldsymbol{\epsilon}} \|\boldsymbol{\epsilon}\| \text{ such that } c^*(\tilde{\mathbf{x}}) \neq c(\mathbf{x}) \text{ and } \tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon} \in \mathcal{X}. \tag{3.26}$$

It should be noted that the magnitude of the perturbation is measured by a norm $\|\cdot\|$ and that the adversarial example has to be an element of the input space $\mathcal{X}$. For MNIST and, hence, for the evaluation performed here, this is the space $\mathcal{X} = [0,1]^{28\cdot 28}$. Thus, the adversarial examples have to be images from the image space of MNIST.

The (optimization) algorithm to find such a perturbation is called *adversarial attack* and is denoted by $\mathfrak{a}$. Adversarial attacks can be grouped into two different approaches, white-box and black-box, distinguished by the amount of knowledge about the model available to the attacker. White-box or gradient-based attacks are based on exploiting the interior gradients of the NNs, while black-box attacks rely only on the output of the model, either the logits, the probabilities, or just the predicted discrete class labels.

Each attack is designed to optimize the adversarial example regarding a given norm, usually an L^p-norm (p-norm). For a $p \geq 1$, the L^p-norm is defined by

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n_x} |x_i|^p \right)^{\frac{1}{p}},$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$. If $p \to \infty$, the L^p-norm is denoted as maximum norm and is calculated by $\|\mathbf{x}\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_{n_x}|\}$. For the cases $0 < p < 1$, the function $\|\mathbf{x}\|_p$ does not define a norm but a quasinorm. If $p = 0$, we define the function $\|\mathbf{x}\|_0$ as the number of elements x_i that are nonzero. This definition of $\|\mathbf{x}\|_0$ by the nonzero counting function is frequently used in scientific computing.

As the LVQ methods were not designed with a specific thread model in mind, the robustness was evaluated over three different L^p-norms using the following black-box and white-box attacks:

- DeepFool by Moosavi-Dezfooli, Fawzi, and Frossard (2016);
- Carlini & Wagner (C&W) by Carlini and Wagner (2017);
- Pointwise by Schott et al. (2019);
- Fast Gradient Sign Method (FGSM) by I. J. Goodfellow et al. (2015);
- Boundary by Brendel, Rauber, and Bethge (2018);
- Projected Gradient Descent (PGD) by Madry et al. (2018);
- Salt & Pepper noise attack (S&P) by Rauber, Brendel, and Bethge (2017).

The difference between these attacks lies in the precise definition of the optimization procedure. Each attack optimizes regarding a certain L^p-norm and each attack handles the box-constraint of $\mathcal{X} = [0,1]^{28 \cdot 28}$ differently. The different norms that are used by these attacks are $p \in \{0, 2, \infty\}$, see Table 3.2 for the L^p-norm definition of each attack. Note that some of the attacks are defined for several norms. We call an attack $\mathfrak{a}$ an L^p-attack if it optimizes with respect to the L^p-norm. All the attacks are implemented in FOOLBOX (Rauber et al., 2017), which was used for the evaluation with the default settings for each attack.

Robustness measures

To evaluate the robustness, we tried to compute an adversary for *each* sample of the MNIST test dataset (10 k images in total) with *each* attack $\mathfrak{a}$—hence, the evaluation dataset $\mathcal{T}$ was the entire test dataset of MNIST. Then, we calculated the adversarial distance $\delta_{\mathfrak{a}}(\mathbf{x}, c(\mathbf{x}))$ proposed by Schott et al. (2019).

Definition 3.8 (adversarial distance)**.** Given a sample $(\mathbf{x}, c(\mathbf{x}))$, an attack $\mathfrak{a}$, and a classification model $\mathbf{f}$, the *adversarial distance* $\delta_{\mathfrak{a}}(\mathbf{x}, c(\mathbf{x}))$ is defined as: zero if the sample $\mathbf{x}$ is misclassified by the model; $\|\boldsymbol{\epsilon}\|$ if the attack $\mathfrak{a}$ found an adversarial example $\tilde{\mathbf{x}}$; infinite if no adversarial example was found by the attack $\mathfrak{a}$. In symbols:

$$\delta_{\mathfrak{a}}(\mathbf{x}, c(\mathbf{x})) = \begin{cases} 0 & \text{if } c^*(\mathbf{x}) \neq c(\mathbf{x}), \\ \|\boldsymbol{\epsilon}\| & \text{if } \mathfrak{a} \text{ found an adversary } \tilde{\mathbf{x}}, \\ \infty & \text{if } \mathfrak{a} \text{ found no adversary to } \mathbf{x}. \end{cases}$$

Based on this definition, we computed four robustness evaluation measures proposed by Schott et al. (2019) to summarize the robustness of a model.

Median adversarial distance: For each attack $\mathfrak{a}$, the *median adversarial distance* score is defined as

$$\text{median-}\delta_{\mathfrak{a}}(\mathcal{T}) = \text{median}\left\{\delta_{\mathfrak{a}}(\mathbf{x}, c(\mathbf{x})) \mid (\mathbf{x}, c(\mathbf{x})) \in \mathcal{T}\right\},$$

describing an averaged adversarial distance over $\mathcal{T}$ robust to outliers. Note that the median adversarial distance can be infinite.

Worst-case median adversarial distance: The *worst-case median adversarial distance* score with respect to an L^p-norm is

$$\text{median-}\delta_p^*(\mathcal{T}) = \text{median}\left\{\delta_p^*(\mathbf{x}, c(\mathbf{x})) \mid (\mathbf{x}, c(\mathbf{x})) \in \mathcal{T}\right\},$$

where $\delta_p^*(\mathbf{x}, c(\mathbf{x}))$ is defined as the *worst-case adversarial distance* of the sample $(\mathbf{x}, c(\mathbf{x}))$:

$$\delta_p^*(\mathbf{x}, c(\mathbf{x})) = \min\left\{\delta_{\mathfrak{a}}(\mathbf{x}, c(\mathbf{x})) \mid \mathfrak{a} \text{ is an } L^p\text{-attack}\right\}.$$

This score is a worst-case evaluation of the median adversarial distance, assuming that each sample is disturbed by the respective worst-case attack (the attack with the smallest perturbation).

Threshold accuracy: The *threshold accuracy* of a model over $\mathcal{T}$ with respect to an L^p-attack $\mathfrak{a}$ is defined as the relative proportion of adversarial examples found with an adversarial distance greater than an L^p-norm specific threshold t_p:

$$\text{acc-}\mathfrak{a}(\mathcal{T}) = \frac{\#\left\{(\mathbf{x}, c(\mathbf{x})) \in \mathcal{T} \mid \delta_{\mathfrak{a}}(\mathbf{x}, c(\mathbf{x})) > t_p\right\}}{\#\mathcal{T}}.$$

This measure represents the remaining accuracy of the model if only adversaries below the specified threshold are considered to be valid. The idea behind this score is to indicate how good a model would perform if we could reject adversarial examples up to a certain threshold.

Worst-case threshold accuracy: The *worst-case threshold accuracy* of a model over $\mathcal{T}$ with respect to L^p-attacks is defined as the relative proportion of adversarial examples found with a worst-case adversarial distance greater than an L^p-norm specific threshold t_p:

$$\text{acc-}\mathfrak{a}_p^*(\mathcal{T}) = \frac{\#\left\{(\mathbf{x}, c(\mathbf{x})) \in \mathcal{T} \mid \delta_p^*(\mathbf{x}, c(\mathbf{x})) > t_p\right\}}{\#\mathcal{T}}.$$

Similar to Schott et al. (2019), we used the following thresholds for the evaluation: $t_0 = 12$, $t_2 = 1.5$, and $t_\infty = 0.3$.

Evaluation setup and models

We evaluated three LVQ methods: GLVQ, GMLVQ, and GTLVQ. The training, as well as the model setups, are equivalent to Section 3.4.2. Since we trained each algorithm with one prototype per class and with so many prototypes per class that the model had roughly 1 M parameters, we have six LVQ models in total. Additionally, we trained a six-layer CNN with the following architecture:[31]

1. Convolution: 32 filters, kernel size 3×3, stride 1×1, bias, no padding, and ReLU activation;
2. Convolution: 64 filters, kernel size 3×3, stride 1×1, bias, no padding, and ReLU activation;
3. Max pooling: pool size and stride 2×2;
4. Convolution: 64 filters, kernel size 3×3, stride 1×1, bias, no padding, and ReLU activation;
5. Convolution: 128 filters, kernel size 3×3, stride 1×1, bias, no padding, and ReLU activation;
6. Max pooling: pool size and stride 2×2;
7. Fully connected: 512 units, bias, and ReLU activation;
8. Dropout: dropout rate 0.5;
9. Fully connected: 10 units, bias, and softmax activation.

[31] To determine the network depth, we only count layers with trainable parameters.

Furthermore, we applied a batch normalization layer with the Keras default setting after the first max pooling layer. The network was trained with the cross-entropy loss using the same augmentation and training pipeline as for the GLVQ algorithms. Unlike the GLVQ algorithms, we used an initial learning rate of 0.003. The initialization of the layers was performed by the default routines of Keras. In the following, this CNN architecture is denoted as CNN-0—see also Section 4.5, where we used this architecture fur further evaluations.

Results

In Table 3.2, we present the results of the robustness evaluation of the models. The robustness measures show outstanding robustness scores against adversarial attacks for GLVQ and GTLVQ models. GLVQ-1M, GTLVQ, and GTLVQ-1M significantly exceed the CNN-0 model for all attacks. As expected, however, they have a considerably lower baseline accuracy (accuracy on the original test dataset).

The reason for the high robustness of LVQ models is that the norm of the adversarial perturbation, see Equation (3.26), is lower bounded by the sample margin and, moreover, the sample margin is lower bounded by the hypothesis margin, see Lemma 3.5. Overall, the following relation holds:

$$\|\boldsymbol{\epsilon}\| = \|\tilde{\mathbf{x}} - \mathbf{x}\| \geq \operatorname{margin}_s\left(\{\mathbf{x}\}, \mathcal{W}\right) \geq \operatorname{margin}_h\left(\{\mathbf{x}\}, \mathcal{W}\right). \tag{3.27}$$

In addition, we know that GLVQ and GTLVQ are hypothesis margin maximizers. The margin maximization is performed with respect to the Euclidean distance and, thus, with respect to the L^2-norm. Therefore, these methods have to be robust against L^2-attacks because they were optimized for a large hypothesis margin. Consequently, we can say that *LVQ methods are provably robust against adversarial attacks.*

The result of Lemma 3.4 and the inequality of Equation (3.27) allows us to calculate a lower bound for adversarial perturbations based on the optimized norm of the LVQ method. For example, consider the results in Table 3.3, where we calculated the hypothesis margins for the GTLVQ model, and compare them to the results of the strongest L^2-attack, the C&W attack.[32] As can be seen, the attack is unable to find an adversarial example with a perturbation less than the hypothesis margin—this is a numerical representation of the theoretical result. Furthermore, if we compute the hypothesis margin for each sample and set the hypothesis margin to zero if an image is misclassified, then the median over these values provides a statistical value similar to the (worst-case) median adversarial distance. For instance, the median

[32]Note that according to Equation (3.25) and Lemma 3.4, we have to calculate the hypothesis margin with the non-squared tangent distances even if we have trained with squared versions.

Table 3.2: Results of the robustness evaluation. The attacks are grouped according to the norm that they optimize. Moreover, the boxes indicate whether the attack is a white-box or black-box attack. For each model, we report the baseline accuracy in percentage, the median adversarial distance median-$\delta_{\mathfrak{a}}$ (left value) and the threshold accuracy acc-$\mathfrak{a}$ (right value) in percentage for each attack, and the worst-case analysis of L^p-attacks by presenting the median-δ_p^* value (left value) and the acc-$\mathfrak{a}_p^*$ score (right value) in percentage. Higher scores mean better robustness. For each attack, the best median-$\delta_{\mathfrak{a}}$ is printed in bold.

			CNN-0		GLVQ				GMLVQ				GTLVQ			
#prototypes					1		128		1		49		1		10	
baseline accuracy				99.6		83.3		95.9		88.3		92.8		94.5		97.4
L^2	DeepFool	☐	1.01	9.6	1.64	53.2	2.33	72.5	0.49	2.8	0.62	4.6	2.32	72.6	**2.54**	80.9
	C&W	☐	0.86	5.2	1.46	48.8	2.05	68.1	0.46	2.3	0.54	2.3	2.06	67.8	**2.24**	76.6
	Pointwise	■	2.62	88.9	4.45	78.5	5.35	92.5	1.64	54.1	2.44	78.2	5.5	91.7	**5.62**	95.3
	Boundary	■	1.12	19.2	2.08	60.9	**3.5**	80.7	0.56	6.8	0.75	7.3	2.75	78.2	3.12	86.3
	worst-case		0.84	1.5	1.46	48.8	2.05	68.1	0.46	2.3	0.54	2.3	2.06	67.8	**2.24**	76.6
L^∞	FGSM	☐	0.19	21	0.17	11	0.29	42.9	0.04	0	0.05	0	0.22	17.5	0.25	26
	DeepFool	☐	0.1	0.1	0.13	6.4	0.22	22.4	0.04	0	0.04	0.1	0.19	8.8	0.22	18.2
	PGD	☐	0.09	6	0.12	3.8	**0.2**	10.7	0.04	0	0.05	0	0.17	3.5	**0.2**	6.1
	worst-case		0.09	0	0.12	3.8	**0.2**	10.7	0.04	0	0.04	0	0.17	3.5	**0.2**	6.1
L^0	Pointwise	■	5	3.3	22	63.9	32	79.1	3	6.1	6	17.9	34	80.1	**35**	85.4
	S&P	■	29	78	126	77	**188**	92.1	8	36.9	17	60.8	155	91.1	179	95.2
	worst-case		5	3.3	22	63.9	32	79.1	3	6.1	6	17.9	34	80.1	**35**	85.4

Table 3.3: Comparison of the hypothesis margin to the adversarial perturbation. We used the first nine test images from the MNIST dataset. The model was GTLVQ with one prototype per class and the attack was the C&W attack.

sample index	1	2	3	4	5	6	7	8	9
$\mathrm{margin}_h(\{\mathbf{x}\}, \mathcal{W})$	1.86	0.74	1.69	1.61	1.05	1.95	1.01	0.54	0.31
$\lVert\boldsymbol{\epsilon}\rVert_2$	3.1	1.72	2.83	2.76	1.95	2.87	1.96	1.35	0.71

over these hypothesis margin values of the previously considered GTLVQ model is 1.07. This value is indeed a lower bound for all the median adversarial distances based on L^2-attacks in the experiment, see Table 3.2.

In Figure 3.15, we present an adversarial example for each attack and each method. The images show a large semantic difference between the adversarial examples for GLVQ-1M, GTLVQ-1M, and the other models. For GLVQ-1M and GTLVQ-1M, a large part of the adversarial examples looks like an interpolation between the original digit and another digit—for example, the interpolation between a one and a four in the first sample. Such a sample corresponds to a sample that has a high hypothesis margin with respect to the classification model. Therefore, we can say that an input sample that has a large hypothesis margin cannot easily be transformed into an adversarial example with a perturbation that is imperceptible to humans.

Considering the results in Table 3.2 and Figure 3.15, then the question arises why GMLVQ is not robust against the adversarial attacks presented. Obviously, GMLVQ has the lowest robustness values across all attacks and methods. Taking into account the strong relationship between GTLVQ and GMLVQ, see Section 3.3.3, it is a remarkable and surprising result because we know from the theoretical analysis that GMLVQ is a hypothesis margin maximizer. However, the margin maximization is performed regarding the corresponding norm (seminorm) to $d_Q(\mathbf{x}, \mathbf{y})$, which is

$$d_Q(\mathbf{x}, \mathbf{0}) = \lVert\mathbf{x}\rVert_Q = \lVert\mathbf{Qx}\rVert_2 .$$

Unfortunately, this norm is not optimized by the adversarial attacks and, therefore, GMLVQ does not seem to be robust in the evaluation. A better statement than *GMLVQ is not robust against adversarial attacks* is that *GMLVQ is robust against adversarial attacks but with respect to another norm.* If we would generate attacks regarding $\lVert\mathbf{Qx}\rVert_2$, then GMLVQ would be robust. In contrast, GTLVQ and GLVQ would generate non-robust scores in this scenario.

But why are GLVQ and GTLVQ robust against adversarial attacks that are different from L^2-attacks—see the results in Table 3.2—considering that they are only

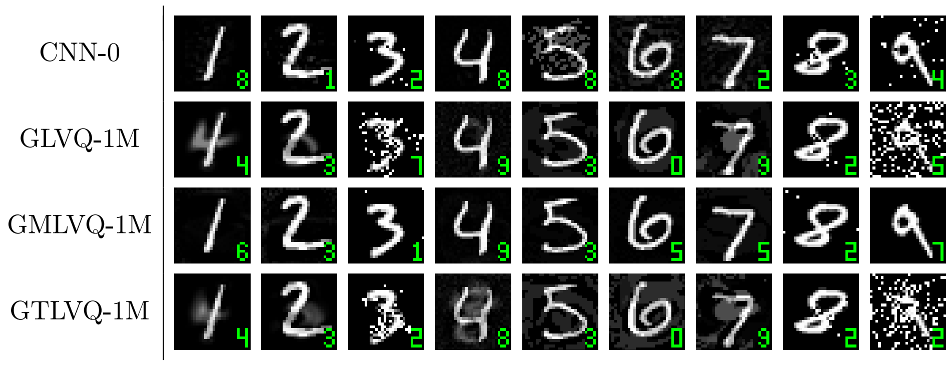


Figure 3.15: For each model, adversarial examples are generated for the following classes by the following attacks (from left to right): class `1` – DeepFool L^2-attack; class `2` – C&W L^2-attack; class `3` – Pointwise L^2-attack; class `4` – Boundary L^2-attack; class `5` – FGSM L^∞-attack; class `6` – DeepFool L^∞-attack; class `7` – PGD L^∞-attack; class `8` – Pointwise L^0-attack; class `9` – S&P L^0-attack. For each class, we used the first image in the test dataset that was correctly classified by all models. We show the prediction of the model after the adversarial attack by the green number in the lower right corner of each image.

provably robust for L^2-attacks? First, we know that all norms in a finite-dimensional vector space are equivalent. Second, the equivalence between L^p-norms implies that the relationship between the L^2-norm and other L^p-norms is good, in the sense that the optimization of the L^2-norm also preserves acceptable results for the other L^p-norms. To demonstrate this relation: For all L^p-norms with $0 < q < p$ in $\mathbb{R}^{n_x}$, it holds that

$$\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_q \leq n_x^{\frac{1}{q}-\frac{1}{p}} \|\mathbf{x}\|_p \, . \tag{3.28}$$

Therefore, a high hypothesis margin regarding the L^2-norm ensures a high hypothesis margin for all L^p-norms with $p < 2$ and, moreover, a high adversarial robustness for L^p-attacks with $p < 2$. Furthermore, the inequality in Equation (3.28) implies that a hypothesis margin maximization with respect to the L^∞-norm might be desirable to train a robust model.

In addition to explaining the robustness behavior of GMLVQ based on hypothesis margin theory, we present another description with the theory of oversimplification. In Section 3.3.3, we discussed that GMLVQ tends to oversimplify (to collapse data dimensions) without regularization. This is a good property to achieve higher accuracies (better generalization) as it helps the model to focus on the relevant features to solve the classification task. For instance, this is why the GMLVQ model outperforms the GLVQ model, and the CNN-0 model outperforms all other models in terms of accuracy. However, the GMLVQ and CNN-0 model are good examples to show that a model that generalizes well is not necessarily robust.

Oversimplification may induce heavy distortions in the mapping between the input and the transformation space, potentially creating dimensions in which a small perturbation in the input space can be mapped to a large perturbation in the transformation space. These dimensions are later used to place adversarial attacks efficiently. To improve the robustness of GMLVQ, penalizing the collapse of dimensions may be a successful approach, see the discussed GMLVQ regularization in Section 3.3.3. This idea is supported by the fact that GTLVQ is just a constrained version of GMLVQ. In general, the relation between the methods provides evidence that a proper regularization or constraint or both can force a model to be more robust.

Another result of this robustness investigation is that an increase in the number of prototypes improves both generalizability and robustness. For all three LVQ methods the robustness improves if the number of prototypes per class increases. Additionally, increasing the number of prototypes leads to a better ability to generalize (higher test accuracy). This observation provides empirical evidence supporting the results of Stutz, Hein, and Schiele (2019): Generalization and adversarial robustness are not necessarily contradictory objectives, which is discussed recently.

Mathematically, this result is also supported by Theorem 3.4. One implication of this theorem is that the selection of the margin threshold is not trivial. If we increase the margin threshold to obtain a more robust model, we may increase the margin error and thus also increase the generalization error. Therefore, if we want a model that is robust and accurate, then we have to search for a good trade-off, or if we want a robust model with a large margin threshold, then we may have to increase the model capacity (by increasing the number of prototypes) to mitigate a decrease of the accuracy due to the large threshold. In summary, the theorem does not state that generalization and adversarial robustness are necessarily contradictory goals.

What does the present evaluation say about the robustness behavior of NNs? In [2017d] and [2018e], we investigated the relationship between GMLVQ and NNs. Conceptually, the two methods are not that different and closer to each other than one might expect—see also the results of the robustness evaluation in Table 3.2. Therefore, some of the conclusions we have drawn for GMLVQ might also apply to NNs. For example, as mentioned above, GMLVQ is robust against adversarial attacks calculated with respect to the seminorm $\|\mathbf{Qx}\|_2$. Accordingly, NNs could also be robust against adversarial attacks calculated with a different norm or measure than those usually used in the evaluation.

In general, the discussion about adversarial robustness of NNs is driven by the goal of developing methods that classify like humans. Thus, we try to use robustness measures that reflect human perceptual abilities. However, the methods we evaluate are optimized to work as good as possible on one dataset and not in the diverse human world. Consequently, it might be wrong to expect a method to become robust in the human sense of perception after training in a fairly limited world—the training dataset—without constraints or regularizations.

3.6 Related work

The tangent distance concept was first defined by Simard et al. (1993) and used for a k-nearest neighbors classifier. There they used predefined tangents and thus more or less predefined affine subspaces. Moreover, the respective tangent distance was a double-sided version (i. e., they measured the distance between two affine subspaces). With their discussion on the application of constraints (restrictions) to the tangent distance, they motivated the investigation of restricted-GTLVQ.

The potential drawback that the tangents have to be predefined was improved by Hastie et al. (1995). By proposing the tangent-centroid and tangent-subspace algorithm, they presented two methods to estimate the affine subspaces for a k-means clustering. However, the resulting iterative estimation of the tangents is not class discriminative. The proposed initialization scheme for the GTLVQ algorithm is motivated by the tangent-subspace algorithm. Furthermore, the idea to study set-prototypes is motivated by this work because the authors motivated the algorithms by the goal to leverage the point or centroid principle of prototypes.

In the work of Schwenk and Milgram (1995), the concept of a single-sided and double-sided tangent distance was clearly defined for the first time. Additionally, they proposed a classification algorithm similar to the LVQ1 algorithm but based on tangent distances. Therefore, they invented the first algorithm that estimates the tangents discriminatively regarding the classes. The presented point-set dissimilarity is motivated by a single-sided tangent distance.

We proposed to estimate the subspace dimension by a stepwise increase of the dimension followed by the initialization procedure to evaluate the model performance. This estimation is inspired by the training algorithm of the tangent-distance-neuron described by Sona, Sperduti, and Starita (2000). In addition, their algorithm also estimates the tangents discriminatively regarding the classes but starting from an NN perspective.

The idea of estimating subspaces or applying the tangent distance concept is used in various ML frameworks. Keysers et al. investigated the derivation of tangent distances from a probabilistic point of view (Keysers, 2000; Keysers, Macherey, Ney,

& Dahmen, 2004). Moreover, Haasdonk and Keysers (2002) applied the tangent distance concepts in support vector machines. In the work of Bengio and Monperrus (2005), the authors considered the task of estimating the tangent space for each point in the input space by an NN. In general, subspace estimation techniques are studied by several authors (e. g., Chi, 2013; Fukui & Maki, 2015; Mi, Huang, Wang, & Zhu, 2013; X. Wang & Tang, 2004; Zhu, Fukui, & Xue, 2017). However, their methods are often reduced to the orthogonal projector equation and therefore turn out to be similar to the tangent distance. Recently, L. Zhang, Edraki, and Qi (2018) presented a capsule NN that uses subspace approximations, and Devos and Grossglauser (2019) investigated the benefits of subspace approximations for few-shot learning.

In the work of Hammer, Strickert, and Villmann (2005), the authors derived a generalization bound for generalized relevance LVQ based on Gaussian complexity. This bound is similar to that of Crammer et al. (2003) and all conclusions drawn in Section 3.5.1 are also reflected in their bound. For instance, the selection of the margin threshold (in their publication denoted by ρ) is not trivial or that there is a nontrivial optimal number of prototypes. In particular, all variables with a similar meaning between the two bounds are in similar dependencies. However, the theory of Hammer et al. is limited to diagonal matrices of size $n_x \times n_x$ in the quadratic-dissimilarity d_Q but applies directly to the squared version.

One motivation for the derivation of another bound was that Hammer et al. assumed that the margin maximization theory of Crammer et al. is only valid for the Euclidean norm. However, as explained in Section 3.5.1, the lemmas apply to an arbitrary seminorm. That this was assumed is not surprising since the publication of Crammer et al. is partly difficult to read. On page 481, for example, they denote the Euclidean norm by $\|\cdot\|_2$ and on page 483 by $\|\cdot\|$. Additionally, the proofs are not part of the main publication and not easily accessible, and the versions of the lemmas differ between the available versions with and without proofs.

3.7 Summary and discussion

In this chapter, we formalized the concept of set-prototypes and derived two nontrivial realizations where the set-prototypes contain infinitely many points: GTLVQ and restricted-GTLVQ. The methods learn the set-prototypes regarding the classification task by SGD and are an extension of the GLVQ framework. The naming *generalized tangent learning vector quantization* refers to this relationship. Moreover, the extensions are called generalized *tangent* LVQ because the methods can be constructed from the idea of approximating the dataset manifolds by tangent spaces. The difference between the two versions is that restricted-GTLVQ explicitly models that the approximation by a tangent space is only valid in a certain vicinity around the center point. On the contrary, GTLVQ assumes the global validity.

From another perspective, GTLVQ uses set-prototypes that are affine subspaces and restricted-GTLVQ uses set-prototypes that are orthotopes. Looking at the amount of information that can be captured by a prototype, GLVQ is limited to vectors and, therefore, to a single point of the data space. In contrast, GTLVQ and restricted-GTLVQ can capture information about infinitely many points by the set-prototypes.

We evaluated both methods on several datasets and showed that the additional degrees of freedom introduced by the set-prototypes are beneficial to achieve high classification accuracies. More specifically, GTLVQ and restricted-GTLVQ consistently outperform other state-of-the-art LVQ methods in all experiments and are interpretable at the same time.

To underline the robustness of the derived algorithms, we presented a theoretical robustness analysis and a numerical evaluation of the robustness against adversarial attacks. The theoretical analysis is based on the result that LVQ algorithms are hypothesis margin maximizers. In the first part, we summarized these results and discussed how they apply to GLVQ and GMLVQ. As far as we know, this is the first time that it has been shown how the theory of Crammer et al. (2003) can be applied to GLVQ and GMLVQ.

A consequence of this finding is an explanation for the observed effects of T. Villmann et al. (2019) in experiments with different squashing functions ϕ in model training. In this study, it was found that the Swish function, see Equation (4.21) for a definition, surpasses all other squashing functions tested in terms of accuracy and training speed. The fast convergence is the result of the good numerical properties of the derivative of the Swish function, and the high accuracy results from the property that the model is optimized for weak classification decisions when using Swish activation (depending on the exact parameterization). Swish has a global minimum in the negative range before converging to zero, so the SGD optimizes the model to a relative distance difference μ equal to this local minimum. If the minimum is close to zero, the SGD optimizes for a small hypothesis margin and thus for a less robust prototype configuration. Furthermore, if the hypothesis margin is small, we can choose a small margin threshold to reduce the margin error. According to Theorem 3.4, this implies a smaller generalization error and, therefore, a higher accuracy.

The original formulation of GLVQ assumes that the squashing function ϕ is monotonically increasing. However, the Swish function does not fulfill this property but represents a margin maximizer according to the results presented. What we influence by using different activation functions is the compromise between a small margin error and a large margin threshold.

In the second part, we discussed how the margin maximizer theory applies to GTLVQ and restricted-GTLVQ. The results show that GTLVQ and restricted-GTLVQ preserve the robustness properties of GLVQ. To confirm this numerically, we performed a robustness evaluation of the methods against adversarial attacks. The result of this evaluation is that GTLVQ is a very robust method on the MNIST dataset and far surpasses NNs. The reason for this good result is that the hypothesis margin is a lower bound of the adversarial perturbation. So if we optimize for a large hypothesis margin, we optimize for an adversarially robust method. A first attempt to transfer the margin maximizer principle to NNs to design adversarially robust methods was made by Elsayed, Krishnan, Mobahi, Regan, and Bengio (2018) by a first-order approximation of the sample margin. In summary, the overall robustness of LVQ models is impressive. Due to the margin maximizer theory, LVQ methods are *provably* robust against adversarial attacks, which can be a valid reason to deploy them instead of NNs in safety-critical applications.

The GTLVQ algorithm is highly similar to the local-GMLVQ algorithm, and we have shown that local-GMLVQ converges to GTLVQ if trained with the frequently used regularization term and a sufficiently high regularization parameter. This relationship is helpful to get a deeper understanding of both methods. For instance, it implies that GTLVQ is not affected by collapsing dimensions and thus by oversimplification. In terms of computational complexity, the GTLVQ method is more expensive during training because we have to orthonormalize the bases after each update step. However, during the inference phase, local-GMLVQ and GTLVQ have the same computational complexity. In contrast, restricted-GTLVQ is always slightly more complex since the equations require more floating-point operations. In terms of memory complexity, the GTLVQ versions are more efficient than the local-GMLVQ. This is because local-GMLVQ explicitly learns the transformation spaces, while the GTLVQ versions explicitly learn the null spaces and implicitly the transformation spaces. Consequently, as long as the dimension of the transformation spaces is higher than the dimension of the null spaces—which is usually given—the GTLVQ and restricted-GTLVQ algorithms have a lower memory complexity.

Regardless of the derived LVQ algorithms, we have proven that the point-set dissimilarity is related to a Hausdorff distance. If metric properties of the point-set dissimilarity are required, this result states that we can perform the computation in terms of a Hausdorff distance so that the metric properties are valid.

The presented algorithms are a promising extension of GLVQ methods and motivate further research. For example, on the following topics.

Set-prototype definitions: It is natural to look for other definitions of set-prototypes than the two presented: affine subspaces and orthotopes. We believe that there are more definitions with tractable solutions and that such concepts can further improve the performance of LVQ algorithms. One possible realization could be the definition of n_s-balls by restricting the parameter vector $\boldsymbol{\theta}$ of an n_s-dimensional affine subspace to a Euclidean norm less than or equal to a certain radius R (i. e., $\|\boldsymbol{\theta}\|_E \leq R$). We claim that the solution for the respective point-set dissimilarity with the Euclidean distance as the underlying dissimilarity is given by

$$\begin{aligned} \mathsf{d}(\mathbf{x}, \mathfrak{w}) &= \min \{d_E(\mathbf{x}, \mathbf{t} + \mathbf{B}\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \mathbb{R}^{n_s}, \|\boldsymbol{\theta}\|_E \leq R\}, \\ &= \sqrt{d_E^2(\mathbf{x}, \mathbf{x}^*) + (\mathrm{ReLU}(d_E(\mathbf{t}, \mathbf{x}^*) - R))^2}, \end{aligned}$$

where $\mathbf{x}^*$ is the best-approximating element in the affine subspace, see Equation (3.22). Another idea is to use a sphere instead of a ball.

Underlying dissimilarity: In all constructed point-set dissimilarities, we used the Euclidean distance as the underlying dissimilarity to derive a tractable solution. Thus, an interesting research topic is to study the tractability of other dissimilarity measures such as divergences, other L^p-norms, and so on. One special case of the L^p-norm could be important if someone is interested in an adversarially robust model for a wide range of L^p-norms, the L^∞-norm—see Equation (3.28) and the following discussion. This norm is related to the Chebyshev distance and could lead to a tractable solution.

Properties of restricted-GTLVQ: In all the experiments we conducted with real-world datasets, restricted-GTLVQ had no significant advantages over GTLVQ that justified the additional computational effort. Both methods worked almost identically and the only benefit of the restricted version was observed in toy datasets. The idea of restricting the affine subspaces was already studied by Simard et al. (1993), and they concluded: “One may worry that the tangent planes $[\mathfrak{w}_i(\boldsymbol{\theta}_i)]$ and $[\mathfrak{w}_j(\boldsymbol{\theta}_j)]$ may be parallel and be very close at a very distant region (a bad side effect of the linear approximation). This effect can be limited by imposing a constraint of the form $[\|\boldsymbol{\theta}_i\|_E < R_i]$ and $[\|\boldsymbol{\theta}_j\|_E < R_j]$. This constraint was implemented but did not yield better results. The reason is that tangent planes are mostly orthogonal in high-dimensional [spaces] and the norms of $[\|\boldsymbol{\theta}_i\|_E]$ and $[\|\boldsymbol{\theta}_j\|_E]$ are already small.” This constraint is comparable to the constraint in the restricted-GTLVQ (i. e., $|\theta_i| \leq a_i$). According to these early results on tangent distances, we believe that this conclusion is the reason why restricted-GTLVQ does not perform better than GTLVQ when low-dimensional affine subspaces are used in high-dimensional spaces. However, we will continue the research on restricted-GTLVQ to understand the observed effects and to investigate whether the restriction parameters a_i can be related to confidence values.

Robustification approaches: The GTLVQ versions are provably robust algorithms against adversarial L^2-attacks. GMLVQ is robust against $\|\mathbf{Qx}\|_2$-attacks and does not seem to be robust against commonly used L^2-attacks. The relationship between local-GMLVQ and GTLVQ is strong and local-GMLVQ can converge to GTLVQ. Furthermore, both methods can be interpreted as an NN. In the presented robustness evaluation, GMLVQ performs more like an NN and has low robustness scores, while GTLVQ is highly robust. An interesting implication is now the following: Currently, there is a lively discussion about how to train an NN to enhance its robust against adversarial attacks. With local-GMLVQ, we have an algorithm that behaves similar to NNs and is not robust against state-of-the-art attacks.[33] In contrast, GTLVQ can be considered as an equivalent algorithm that is highly robust and the result of a successful robustification method. Thus, GTLVQ can be considered as a desirable solution in terms of robustness, and the success of a proposed robustification method can be measured by applying the method to local-GMLVQ and comparing it with GTLVQ. This evaluation is fast and provides a deeper understanding of a robustification approach due to the inherent interpretability properties of LVQ methods. One of the reasons why GTLVQ is highly robust is the orthonormalization of the transformation matrices. This concept was recently studied by Cisse, Bojanowski, Grave, Dauphin, and Usunier (2017) for NNs by orthonormalizing each weight matrix. Beyond that, the applicability of the GMLVQ regularizer (see Section 3.3.3) in NNs should be investigated in the context of adversarial robustness of NNs.

[33] A local-GMLVQ network with one prototype vector per class and a transformation dimension $m_x = 28^2 - 12$, which corresponds to the subspace dimension of the GTLVQ model used in the robustness evaluation, achieves a test accuracy of 97 %. The robustness scores with respect to the C&W attack are 0.47 for the median adversarial distance and 0.6 % for the threshold accuracy. Therefore, the local-GMLVQ model is one of the worst models regarding the C&W attack in the evaluation. Moreover, the local-GMLVQ model has $28^2 \cdot \left(28^2 - 12 + 1\right) = 606\,032$ parameters per prototype, including the transformation matrix, and the GTLVQ model has $28^2 \cdot (12 + 1) = 10\,192$ parameters per set-prototype.

This chapter is mainly based on the following joint work:

[2019c] Saralajew, Holdijk, Rees, Asan, and Villmann (2019). Classification-by-components: Probabilistic modeling of reasoning over a set of components.

Chapter 4
Classification-by-Components Networks

Class-specific prototypes are the core concept of prototype-based ML methods. Unfortunately, this concept becomes computationally and memory intensive if the classification task consists of a large number of classes or a large input dimension n_x or both. This restricts the applicability of LVQ algorithms—for example, of the previously proposed GTLVQ. For instance, if we apply a GLVQ network to datasets from the ImageNet project of Deng et al. (2009) without applying appropriate data preprocessing techniques, the complexity can be enormous. The usual size of the images used as network input on ImageNet is $224 \times 224 \times 3$. Moreover, the ILSVRC-2012[1] dataset from the ImageNet project is a 1 000-class problem. If we train with at least one prototype per class defined in the input space, we get a total of approximately 150 M parameters without considering possible parameters of adaptive dissimilarity measures. In comparison, a relatively small *Residual neural Network* (ResNet) like the ResNet-50 proposed by He, Zhang, Ren, and Sun (2016) has about 26 M parameters and converges to unattainably high accuracies for LVQ algorithms.[2] Despite the high accuracies, this NN type is also difficult to interpret, see Chapter 1, and questions such as which regions in the image may provide evidence in favor of or against the current classification decision are not easy to answer.

In this chapter, we relax the concept of class-specific prototypes by drawing inspiration from Biederman's (1987) *Recognition-By-Components* (RBC) theory from cognitive psychology. Based on that idea, we derive a probabilistic classification principle called *Classification-By-Components networks* (CBCs), which is restricted to follow an intuitive reasoning-based decision process. There, the prototypes are not equipped with a class label and, hence, are called *components* instead of prototypes. The network is trained to learn and detect generic components that characterize objects. In parallel, a class-wise reasoning strategy based on these components is learned to solve the classification problem. In contrast to other approaches on reasoning (as discussed later), we propose three different types of reasoning: *positive*, *negative*, and

[1]ImageNet Large Scale Visual Recognition Challenge 2012.

[2]Pretrained ResNet-50 model from the Keras Applications library.

indefinite. These three types together form a probability space to provide a probabilistic classifier. The decomposition of objects into generic components combined with the probabilistic reasoning ensures a clear interpretation of the classification decision process.

The proposed CBC architecture can be interpreted as a commonly used NN architecture with a special constraint on the last two layers defined by the probability model of a CBC. This special constraint

- preserves the interpretability,
- gives a concrete meaning to the last two layers of an NN,
- ensures the applicability in almost all NNs designed for classification tasks, and
- preserves a probabilistic output.

Thereby, the probabilistic output in CBCs is *not* obtained by applying an artificial squashing operation, as it is often performed in NNs with the final softmax operation. Recently published results suspect that the final softmax operation in NNs causes some of the intriguing properties associated with adversarial robustness (e. g., Hein, Andriushchenko, & Bitterwolf, 2019; Nar, Ocal, Sastry, & Ramchandran, 2019).

In the next section, we describe in detail how the CBC architecture is inspired by Biederman's RBC theory and which intuitive parts we try to model in a mathematical framework. We then construct the basic classification paradigm called *reasoning* and show how this can lead to different classifier types. Although the method is closely related to principles of NNs, in this section, we keep the perspective of prototype-based ML methods. For example, we show how learning rules are derived in a simple SGD. In the following section, we present how the CBC architecture can be trained in conjunction with a feature extractor based on a CNN. The presented evaluations are performed to show that the CBC architectures have properties comparable to LVQ methods. In particular, these evaluations present new approaches to interpret NNs and show that CBCs achieve state-of-the-art accuracies. Finally, we discuss related work and provide a summary of this chapter. The software and example scripts for the most important experiments are available at `https://github.com/saralajew/cbc_networks` as a Keras package.

Research chronology

The research into the direction of CBCs began with a study about the relations between LVQ methods and NNs, as first discussed in [2017d]. Later we refined the findings and presented first results where we transferred ideas from NNs to LVQ methods, see [2018a], [2018b], [2018c], and [2019a]. Some of these ideas are based on incorporating cross-entropy learning in LVQ, for instance, see [2018b]. Additionally, in [2018e], we investigated the close relationship between *robust soft learning vector quantization* proposed by Seo and Obermayer (2003) and NNs with final softmax activation and cross-entropy loss.

A first attempt to define a general framework for hybrid LVQ and NN architectures was presented in [2018d]. This approach is mainly based on the idea to integrate a robust soft learning vector quantization layer into NNs. Although the method has worked well in some experiments, applying it to arbitrary datasets and tasks is a challenge. Sometimes the method trained to interpretable results and sometimes not. Later we identified two reasons for this: First, the training of prototypes by the cross-entropy loss does not force the model to learn data-point-like prototypes because, even with very different prototypes, an output probability of one for the correct class can be achieved, see also Section 4.2.4.[3] Second, the output does not depend on the dissimilarities of the intermediate LVQ layers. By applying NN layers before and after each LVQ layer, the NN layers learn to transmit the information only through the vector outputs and completely ignore the dissimilarity coding. In general, this is possible because, in modern NN architectures, the intermediate layers are not limited to transfer information through high activations, so the intermediate layers only learn a transformation of the input data. This backdoor is used by the NN layers to encode the information unexpectedly so that the proposed model generally does not work well.

Several remarks about LVQ and NNs, including ideas we used later to design the CBC architecture, are discussed in [2018e]. For example, the formulation of sliding operations with prototypes, the training with a GLVQ-loss-like function, and the restriction of models to preserve interpretability.

Author contributions

Lars Holdijk: The idea to classify with prototypes in a kind of positive and negative argumentation was first mentioned by him after he had realized that LVQ cannot gain information in favor of a class from a mismatch of a prototype. Moreover, he was deeply involved in the entire development process of the CBC architecture through numerous discussions. The extension of the CBC architecture with several versions of a component was proposed by Lars, see Section 4.2.3. Additionally, he conducted the evaluations on ImageNet, performed the robustness and rejection experiments on MNIST, and contributed to the writing of the publication.

[3]This is also possible by training with the GLVQ loss (Hammer et al., 2014).

Maike Rees: She was involved in several discussions about the relationship between CBCs and Biederman's RBC theory. The first version of the evaluation pipeline for the ablation study was developed by Maike. Furthermore, she contributed significantly to the writing of the publication.

Ebubekir Asan: The source code for the GTSRB evaluations and the first experimental results were created by him. Ebubekir also designed the initial CNN baseline models and made the first evaluations on the real-world adversarial examples, which were later refined to the versions used in the publication.

Thomas Villmann: After we had found the equation to calculate the class hypothesis probabilities by heuristic considerations, several long discussions and brainstorming sessions with him led to the probabilistic framework. To emphasize this: Thomas has made a profound contribution to unraveling and understanding the heuristic equation so that it can now be so beautifully formalized. In addition, the clear mathematical formulation and the relation to prototype-based learning were worked out with him. Additionally, he contributed to the writing of several sections of the publication and helped to select the experiments for publication.

4.1 Motivation

Starting from a cognitive science perspective, Lake, Ullman, Tenenbaum, and Gershman (2017) suggested rethinking current trends in ML to build truly human-like learning and thinking machines: "We argue that these machines should

(1) build causal models of the world that support explanation and understanding, rather than merely solving pattern recognition problems;

(2) ground learning in intuitive theories of physics and psychology to support and enrich the knowledge that is learned; and

(3) harness compositionality and learning-to-learn to rapidly acquire and generalize knowledge to new tasks and situations" (p. 1).

The method proposed in this chapter aims to provide a possible solution for some of these requirements by drawing inspiration from Biederman's RBC theory. Roughly speaking, Biederman's theory describes how humans visually recognize complex objects by assuming that objects can be decomposed into generic parts that operate as structural primitives, called components. Objects are then classified by matching the *extracted decomposition plan* with a *class Decomposition Plan* (DP) for each potential object class. Intuitively, the class DPs describe which components are important

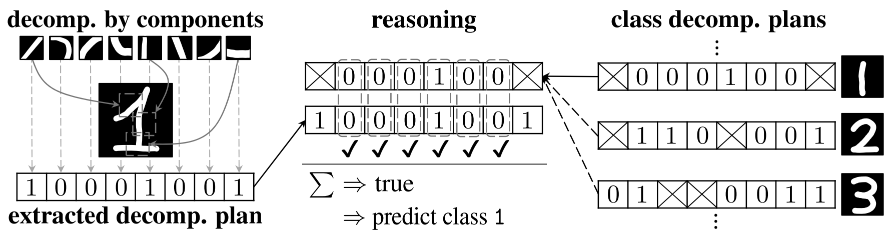


Figure 4.1: An example realization of the classification process of a CBC on a digit classification task. For simplicity, we illustrate a discrete case where “1” corresponds to detection or positive reasoning, “0” to no detection or negative reasoning, and “⊠” to indefinite reasoning.

to be detected and which components are important to not be detected for an object to belong to a specific class. For example, if we consider the classification of a digit, as illustrated in Figure 4.1, the detection of a component representing a vertical bar provides evidence in favor of the class `1`. In other words, we *reason positively* over the vertical bar component for the class `1`. Similarly, we can *reason negatively* over all curved components. In contrast to other works on reasoning, the presented approach extends these two intuitive reasoning states by a third type, the so-called *indefinite reasoning*. In Figure 4.1, not all components will be important for the recognition of a one. For instance, we reason neither positively nor negatively over the serif and bottom stroke because not all writing styles use them.

The next section presents a mathematical framework that models the described classification process in an end-to-end trainable framework so that the components and the class DPs can be learned. This process is a *simplified* realization of the RBC theory. However, we will show that it is a step forward to design an ML model that is not only based on pattern recognition but also has advanced interpretation properties and uses compositionality to some extent.

4.2 Probabilistic modeling of reasoning over a set of components

This section derives the probabilistic model based on the idea of reasoning over a set of components. These components may have a dimension that is not necessarily the same as the input dimension.[4]

[4]Such a relaxation is also possible for prototypes if an appropriate dissimilarity operation is used (e. g., C. Chen et al., 2019).

Definition 4.1 (component)**.** Given a classification task with the input space $\mathbb{R}^{n_x}$, a *component* $\boldsymbol{\kappa}_k$ is an element of the space $\mathbb{R}^{n_\kappa}$. All components are collected into a set

$$\mathcal{K} = \left\{\boldsymbol{\kappa}_k \in \mathbb{R}^{n_\kappa} \mid k = 1, 2, \ldots, \#\mathcal{K}\right\}, \tag{4.1}$$

where $\#\mathcal{K}$ is the overall number of components. A component is called

- *full-size component* if $n_\kappa = n_x$,
- *patch component* if $n_\kappa < n_x$, and
- *oversize component* if $n_\kappa > n_x$.

Compared to the prototype Definition 2.8, the components are not equipped with a class label and, thus, are class-independent. Based on Biederman's RBC theory and the idea that components are structural primitives, the patch definition seems to be the most suitable one. In terms of prototype theory and LVQ models, however, the full-size components are related to prototypes.

The main differences between the different reasoning realizations result from the appropriate handling of the dimension of the components. However, the basic reasoning principle is the same in all realizations and is therefore described in detail in the next part with full-size components. After that, we show how this basic classification paradigm is generalized to patch components, which shows how the principle can be extended to oversize components. The third part deals with approaches to increase the performance of CBCs using multiple components or reasoning strategies.

In general, the CBCs return a *possibility vector* $\mathbf{p}(\mathbf{x}) \in [0,1]^{\#\mathcal{C}}$ and not a probability vector as is usual for NNs. The difference between a possibility vector and a probability vector is that the requirement $\sum_c p_c(\mathbf{x}) = 1$ does not necessarily hold for a possibility vector. In the final part, we show that the possibility vector can be transformed into a probability vector in terms of the probability model to avoid naive normalization. Additionally, we discuss several theoretical aspects of CBCs, including the relationship to NNs, initialization schemes, and so on.

4.2.1 Reasoning over a set of full-size components

The proposed framework relies on a probabilistic model based on a probability tree diagram T. This tree T can be decomposed into sub-trees T_c for each class c with the prior class probability $P(c)$ on the starting edge. Such a sub-tree is depicted in Figure 4.2. The entire probability tree diagram is modeled by the following five random variables:

- $c \in \{1, \ldots, \#\mathcal{C}\}$, indicator variable of the class;

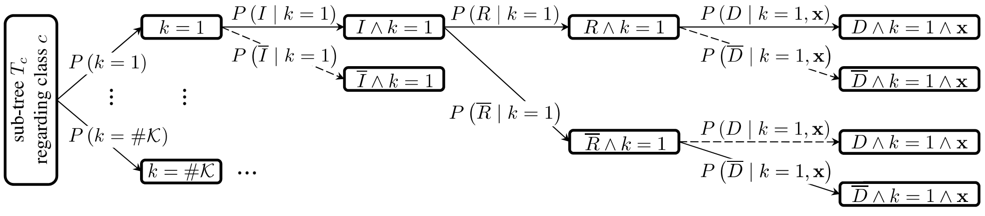

sub-tree T_c regarding class c
$P(k=1)$
$P(k=\#\mathcal{K})$
$k=1$
$k=\#\mathcal{K}$
$P(I \mid k=1)$
$P(\overline{I} \mid k=1)$
$I \wedge k=1$
$\overline{I} \wedge k=1$
$P(R \mid k=1)$
$P(\overline{R} \mid k=1)$
$R \wedge k=1$
$\overline{R} \wedge k=1$
$P(D \mid k=1,\mathbf{x})$
$P(\overline{D} \mid k=1,\mathbf{x})$
$D \wedge k=1 \wedge \mathbf{x}$
$\overline{D} \wedge k=1 \wedge \mathbf{x}$

Figure 4.2: The probability tree diagram T_c representing the reasoning about a class c. For better readability, the variable of class c is dropped in the mathematical expressions and we only show the full sub-tree for the first component. The solid line paths are the paths of agreement.

- $k \in \{1,\dots,\#\mathcal{K}\}$, indicator variable of the component;
- I, binary random variable for importance;
- R, binary random variable for reasoning by detection;
- D, binary random variable for detection.

The probabilities in the tree T_c are interpreted in the following way:

- $P(k)$, prior probability that the k-th component occurs;
- $P(I \mid k, c)$, probability that the k-th component is important for class c;
- $P(R \mid k, c)$, probability that the k-th component has to be detected for class c;
- $P(D \mid k, \mathbf{x})$, probability that the k-th component is detected in the input $\mathbf{x}$.

The horizontal bar indicates the complementary event—for instance, $P\left(\overline{D} \mid k, \mathbf{x}\right)$ is the probability that the k-th component is *not* detected in the input $\mathbf{x}$. Accordingly, the remaining probabilities are derived.

Based on these definitions, we derive the CBC architecture. This mathematical derivation follows the intuitive motivation from Section 4.1. We begin by describing the extraction of the DP depending on an input and a set of components. After that, we specify the class DPs and derive the mathematical model of the reasoning process and thus the equation for calculating a class hypothesis probability. Finally, we introduce the loss function for the training of CBCs and derive the corresponding gradients to formulate the learning rules for a simple SGD.

Extracting the decomposition plan

Given an input $\mathbf{x} \in \mathbb{R}^{n_x}$ and a set of trainable full-size components $\mathcal{K}$, the first part of the network detects the presence of a component $\boldsymbol{\kappa}_k$ in $\mathbf{x}$. This is realized by calculating the probability $P(D \mid k, \mathbf{x})$ for the detection of a component by an appropriate *detection probability function* $d_k(\mathbf{x}) = d(\mathbf{x}, \boldsymbol{\kappa}_k) \in [0,1]$ with the requirement that $\mathbf{x} = \boldsymbol{\kappa}_k$ implies $d_k(\mathbf{x}) = 1$. The interval $[0,1]$ ensures that we can interpret the value as a probability. By the other requirement, we guarantee that it is a detection probability that assigns a probability of one if the component is equal to the input. Note that we do not require that $d_k(\mathbf{x}) = 1$ if and only if $\mathbf{x} = \boldsymbol{\kappa}_k$ (i. e., $d_k(\mathbf{x}) = 1$ can also hold for some $\mathbf{x} \neq \boldsymbol{\kappa}_k$). To finalize the first part of the network, the detection probabilities are collected into the extracted DP in the form of a possibilistic vector

$$\mathbf{d}(\mathbf{x}) = (d_1(\mathbf{x}), \ldots, d_{\#\mathcal{K}}(\mathbf{x}))^{\mathrm{T}} \in [0,1]^{\#\mathcal{K}},$$

similar to the prototype response vector Equation (2.8).

The idea of the detection probability function is to model a kind of similarity measure. Nebel et al. (2017) described a similarity $s(x,y)$ as a function that "increases with x and y sharing more properties and decreases with the number and the degree of discriminating features. Thus, a similarity is taken as a function of object commonalities and differences" (p. 43). This fully reflects the goal and idea of the detection probability function. Additionally, the simple requirements for the detection probability function are sufficient to fulfill their definition of a basic-similarity.

Knowing that the detection probability function is a kind of similarity measure, we can use several known mathematical functions to define a suitable realization. For example, the *cosine similarity* with a special handling of the negative part gives a possible realization:

$$d_k(\mathbf{x}) = \varphi(\cos(\mathbf{x}, \boldsymbol{\kappa}_k)) = \varphi\left(\frac{\mathbf{x}^{\mathrm{T}}\boldsymbol{\kappa}_k}{\|\mathbf{x}\|_E \|\boldsymbol{\kappa}_k\|_E}\right), \tag{4.2}$$

where φ is a transformation function to handle the negative part. The transformation can be the frequently used shift and scaling operation $\varphi(x) = \frac{1}{2}(x+1)$, the clipping of the negative part by $\varphi(x) = \mathrm{ReLU}(x)$, and so on. Another realization for a detection probability is

$$d_k(\mathbf{x}) = \exp\left(-\frac{d_E^2(\mathbf{x}, \boldsymbol{\kappa}_k)}{\sigma}\right). \tag{4.3}$$

The scaling parameter $\sigma \in \mathbb{R}_{>0}$ is important to control the distribution of distances over the given dataset. If this parameter is not carefully set, the gradients of the detection probability function with respect to the components suffer from the problem of vanishing gradients so that the components cannot be learned by SGD (Ghiasi-Shirazi, 2019).

Modeling of the class decomposition plans

The second part of the network models the class DPs for each class c using the three forms of reasoning discussed earlier. Mathematically, the probabilities of these reasoning types are defined as follows.

Positive reasoning: $r^+_{c,k} = P(I, R \mid k, c)$, that is, the probability that the k-th component is important (event I) and must be detected (event R) to support the class hypothesis c.

Negative reasoning: $r^-_{c,k} = P\left(I, \overline{R} \mid k, c\right)$, that is, the probability that the k-th component is important (event I) and must *not* be detected (event $\overline{R}$) to support the class hypothesis c.

Indefinite reasoning: $r^0_{c,k} = P\left(\overline{I} \mid k, c\right)$, that is, the probability that the k-th component is not important (event $\overline{I}$) for the class hypothesis c.

The *reasoning probabilities* $r^+_{c,k}$, $r^-_{c,k}$, and $r^0_{c,k}$ are trainable parameters of the model. Additionally, they form a probability space:

$$\begin{aligned}
r^+_{c,k} + r^-_{c,k} + r^0_{c,k} &= P(R \mid k, c) P(I \mid k) + P\left(\overline{R} \mid k, c\right) P(I \mid k, c) + P\left(\overline{I} \mid k, c\right), \\
&= \left(P(R \mid k, c) + P\left(\overline{R} \mid k, c\right)\right) P(I \mid k, c) + P\left(\overline{I} \mid k, c\right), \\
&= 1.
\end{aligned}$$

All reasoning probabilities are collected class-wise into vectors

$$\mathbf{r}^+_c = (r^+_{c,1}, \ldots, r^+_{c,\#\mathcal{K}})^{\mathrm{T}} \in [0,1]^{\#\mathcal{K}}$$

and $\mathbf{r}^-_c$, $\mathbf{r}^0_c$, respectively. It should be noted that the idea of explicitly modeling the state in which a component does not contribute and thus avoiding the general probabilistic approach $r^+_{c,k} = 1 - r^-_{c,k}$ is related to the Dempster–Shafer theory of evidence (Shafer, 1976).

Reasoning

We compute the class hypothesis probability $p_c(\mathbf{x})$ regarding the paths of agreement under the condition of importance. An *agreement* A is a path in the probability tree diagram T where either a component is detected (event D) and requires reasoning by detection (event R), or a component is not detected (event $\overline{D}$) and requires reasoning by no detection (event $\overline{R}$). The paths of agreement are marked with solid lines in

Figure 4.2. Accordingly, we calculate $p_c(\mathbf{x})$ by $P(A \mid I, \mathbf{x}, c)$:

$$\begin{aligned} P(A \mid I, \mathbf{x}, c) &= \frac{P(A, I \mid \mathbf{x}, c)}{P(I \mid c)}, \\ &= \frac{\sum_k P(A, I \mid k, \mathbf{x}, c) P(k)}{\sum_k P(I \mid k, c) P(k)}. \end{aligned}$$

Using the definition of A, the probability $P(A \mid I, \mathbf{x}, c)$ is obtained as

$$\begin{aligned} P(A \mid I, \mathbf{x}, c) &= \frac{\sum_k \left(P(R, D, I \mid k, \mathbf{x}, c) + P\left(\overline{R}, \overline{D}, I \mid k, \mathbf{x}, c\right)\right) P(k)}{\sum_k \left(1 - P\left(\overline{I} \mid k, c\right)\right) P(k)}, \\ &= \frac{\sum_k \left(P(I, R \mid k, c) P(D \mid k, \mathbf{x}) + P\left(I, \overline{R} \mid k, c\right) P\left(\overline{D} \mid k, \mathbf{x}\right)\right) P(k)}{\sum_k \left(1 - P\left(\overline{I} \mid k, c\right)\right) P(k)}. \end{aligned}$$

Substituting the probabilities with the short form notations, using the relations $P(D \mid k, \mathbf{x}) = d_k(\mathbf{x})$ and $P\left(\overline{D} \mid k, \mathbf{x}\right) = 1 - d_k(\mathbf{x})$, and further assuming that $P(k) = \frac{1}{\#\mathcal{K}}$ leads to[5]

$$P(A \mid I, \mathbf{x}, c) = \frac{\sum_k \left(r_{c,k}^{+} d_k(\mathbf{x}) + r_{c,k}^{-}(1 - d_k(\mathbf{x}))\right)}{\sum_k \left(1 - r_{c,k}^{0}\right)}.$$

A rewriting of this equation with matrix calculus yields

$$\begin{aligned} p_c(\mathbf{x}) &= P(A \mid I, \mathbf{x}, c), \\ &= \frac{(\mathbf{d}(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{r}_c^{+} + (\mathbf{1} - \mathbf{d}(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{r}_c^{-}}{\mathbf{1}^{\mathrm{T}} \cdot \left(\mathbf{1} - \mathbf{r}_c^{0}\right)}, \end{aligned} \tag{4.4}$$

where $\mathbf{1}$ is the one vector of dimension $\#\mathcal{K}$. This equation can be further simplified to

$$p_c(\mathbf{x}) = \frac{(\mathbf{d}(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{r}_c^{+} + (\mathbf{1} - \mathbf{d}(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{r}_c^{-}}{\mathbf{1}^{\mathrm{T}} \cdot \left(\mathbf{r}_c^{+} + \mathbf{r}_c^{-}\right)}, \tag{4.5}$$

and finally to

$$p_c(\mathbf{x}) = (\mathbf{d}(\mathbf{x}))^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^{+} + (\mathbf{1} - \mathbf{d}(\mathbf{x}))^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^{-}, \tag{4.6}$$

where $\bar{\mathbf{r}}_c^{\pm}$ are the *effective reasoning possibility vectors* defined by

$$\bar{\mathbf{r}}_c^{\pm} = \frac{\mathbf{r}_c^{\pm}}{\mathbf{1}^{\mathrm{T}} \cdot \left(\mathbf{r}_c^{+} + \mathbf{r}_c^{-}\right)}.$$

[5] We experimented with networks in which the priors $P(k)$ were kept as trainable parameters. The assumption that they must add up to one was modeled by a softmax squashing. We found no real advantage in keeping these parameters trainable. However, it can be advantageous to train sparse models or to reduce the number of components (pruning) after training by clipping low probability components.

The probabilities for all classes are then collected into the *class hypothesis possibility vector*

$$\mathbf{p}(\mathbf{x}) = \left(p_1(\mathbf{x}), \ldots, p_{\#\mathcal{C}}(\mathbf{x})\right)^{\mathrm{T}} \in [0,1]^{\#\mathcal{C}}$$

to create the network output.

We emphasize again that $\mathbf{p}(\mathbf{x})$ is a *possibility* vector. Throughout this chapter, we consider $\mathbf{p}(\mathbf{x})$ to be the network output and, thus, as the classifier function.[6] The winner-takes-all rule, according to Equation (1.1), applied on $\mathbf{p}(\mathbf{x})$ returns the class prediction of the network.

Training of a CBC

The trainable parameters of a CBC are the components $\boldsymbol{\kappa}_k$ and the reasoning possibility vectors $\mathbf{r}_c^+$ and $\mathbf{r}_c^-$—note that we do not have to learn $\mathbf{r}_c^0$ explicitly as it results from $\mathbf{r}_c^+$ and $\mathbf{r}_c^-$ by $\mathbf{r}_c^0 = \mathbf{1} - \mathbf{r}_c^+ - \mathbf{r}_c^-$. We collect all these trainable parameters in a parameter vector $\boldsymbol{\vartheta}$ that provides the CBC classifier function $\mathbf{p}(\mathbf{x};\boldsymbol{\vartheta})$. Given $\mathbf{p}(\mathbf{x};\boldsymbol{\vartheta})$ and a training sample $(\mathbf{x}, c(\mathbf{x})) \in \mathcal{T}$, we define the probability $p^+(\mathbf{x}) = p_{c(\mathbf{x})}(\mathbf{x})$ of the correct class $c(\mathbf{x})$ of the input $\mathbf{x}$ and the probability $p^-(\mathbf{x}) = \max\{p_c(\mathbf{x}) \mid c \neq c(\mathbf{x})\}$ of the highest probable incorrect class of the input $\mathbf{x}$. We use these probabilities to define the *signed probability gap*:

$$\nu(\mathbf{x}) = p^-(\mathbf{x}) - p^+(\mathbf{x}) \in [-1, 1]. \tag{4.7}$$

This function is similar to the relative distance difference $\mu(\mathbf{x})$ of GLVQ, see Equation (2.12), with the distinction that this function measures an absolute difference instead of a relative difference. Similarly to Equation (2.12), the function is negative for correct classifications and positive otherwise.

We train the CBCs end-to-end by minimizing a contrastive loss function based on the signed probability gap using SGD. In the following, this loss is denoted as *CBC loss function* and defined by

$$l(\mathbf{p}(\mathbf{x};\boldsymbol{\vartheta}), c(\mathbf{x})) = \phi(\nu(\mathbf{x})). \tag{4.8}$$

Similar to the GLVQ loss in Equation (2.15), the function $\phi : [-1,1] \to \mathbb{R}$ is a monotonically increasing, almost everywhere differentiable squashing function. It regulates the generalization-robustness-trade-off over the probability gap between the correct and the highest probable incorrect class.

Given a training sample $(\mathbf{x}, c(\mathbf{x})) \in \mathcal{T}$, the SGD on $l(\mathbf{p}(\mathbf{x};\boldsymbol{\vartheta}), c(\mathbf{x}))$ is based on the following gradients: Assume $c^+ = c(\mathbf{x})$ to be the correct class and c^- to be the

[6]Note that we do not use $\mathbf{f}(\mathbf{x})$ as the symbol for a classifier function to clearly highlight the difference between the probabilistic and the general network output.

highest probable incorrect class of $\mathbf{x}$ according to $p^{-}(\mathbf{x})$. Furthermore, we denote by $\mathbf{r}_{c^{\pm}}^{+}$ and $\mathbf{r}_{c^{\pm}}^{-}$ the corresponding reasoning possibility vectors to $c^{\pm}$. The CBC loss function, see Equation (4.8), has nontrivial gradients for the parameters $\mathbf{r}_{c^{\pm}}^{+}$, $\mathbf{r}_{c^{\pm}}^{-}$, and *all* components $\boldsymbol{\kappa}_k$. These gradients are

$$\frac{\partial}{\partial \mathbf{r}_{c^{\pm}}^{+}} \phi\left(\nu\left(\mathbf{x}\right)\right) = \mp \xi^{\pm}\left(\mathbf{d}\left(\mathbf{x}\right) - p^{\pm}\left(\mathbf{x}\right)\mathbf{1}\right), \tag{4.9}$$

$$\frac{\partial}{\partial \mathbf{r}_{c^{\pm}}^{-}} \phi\left(\nu\left(\mathbf{x}\right)\right) = \mp \xi^{\pm}\left(\mathbf{1} - \mathbf{d}\left(\mathbf{x}\right) - p^{\pm}\left(\mathbf{x}\right)\mathbf{1}\right), \tag{4.10}$$

$$\frac{\partial}{\partial \boldsymbol{\kappa}_k} \phi\left(\nu\left(\mathbf{x}\right)\right) = \left(\xi^{-}\left(r_{c^{-},k}^{+} - r_{c^{-},k}^{-}\right) - \xi^{+}\left(r_{c^{+},k}^{+} - r_{c^{+},k}^{-}\right)\right) \frac{\partial d\left(\mathbf{x}, \boldsymbol{\kappa}_k\right)}{\partial \boldsymbol{\kappa}_k}, \tag{4.11}$$

where

$$\xi^{\pm} = \frac{\phi'\left(\nu\left(\mathbf{x}\right)\right)}{\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_{c^{\pm}}^{+} + \mathbf{r}_{c^{\pm}}^{-}\right)} \geq 0 \tag{4.12}$$

are the gradient scaling factors similar to the GLVQ scaling factors, see Equation (2.19).[7] The learning rules for a basic SGD, according to Equation (1.3), are derived accordingly. Note that this approach requires a differentiable detection probability function. Since the components $\boldsymbol{\kappa}_k$ are not class-specific, we update *all* components at each step. This is an essential difference to the updates in LVQ approaches and results from the class independence of the components.

If the components are trained, it could be necessary to constrain the components to remain in the input space, as discussed in Section 2.2. This can be achieved by a form of projected gradient descent learning where the components are clipped back to the correct range after each update.

As already mentioned, an essential requirement for CBCs is that the trainable reasoning vectors $\mathbf{r}_c^{+}$, $\mathbf{r}_c^{0}$, and $\mathbf{r}_c^{-}$ are elements of $[0,1]^{\#\mathcal{K}}$ and add up to one—that is, $\mathbf{r}_c^{+} + \mathbf{r}_c^{0} + \mathbf{r}_c^{-} = \mathbf{1}$. This condition is preserved by encoding the three reasoning vectors for each class into two vectors that can be learned. These two vectors are denoted as $\mathbf{a}_c$ and $\mathbf{b}_c$ and are defined to be elements of $[0,1]^{\#\mathcal{K}}$. The decoding into the reasoning vectors is defined by

$$\begin{aligned} \mathbf{r}_c^{+} &= \mathbf{a}_c, \\ \mathbf{r}_c^{-} &= \left(\mathbf{1} - \mathbf{a}_c\right) \circ \mathbf{b}_c, \\ \mathbf{r}_c^{0} &= \mathbf{1} - \mathbf{a}_c - \left(\mathbf{1} - \mathbf{a}_c\right) \circ \mathbf{b}_c. \end{aligned} \tag{4.13}$$

[7] In a correct configuration, the expression $\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_c^{+} + \mathbf{r}_c^{-}\right)$ is always greater than zero and therefore the division is well-defined. This is because $\mathbf{r}_c^{+} + \mathbf{r}_c^{-} = \mathbf{0}$ implies $\mathbf{r}_c^{0} = \mathbf{1}$, which means that all components are unimportant and, thus, the class hypothesis probability of class c is ill-defined. Assuming that each class is represented in the dataset and that the CBC is initialized and trained accordingly, this situation is unlikely.

During training of a network, the parameter vectors $\mathbf{a}_c$ and $\mathbf{b}_c$ are constrained to $[0,1]^{\#\mathcal{K}}$ by clipping values outside the interval after each update, which can be considered as a form of projected gradient descent learning.

Another possible coding scheme is the encoding of $\mathbf{r}_c^+$, $\mathbf{r}_c^0$, and $\mathbf{r}_c^-$ into three real-valued vectors. The decoding could then be realized by the application of a probability squashing like a softmax transformation over each triplet $\left(r_{c,k}^+, r_{c,k}^0, r_{c,k}^-\right)$.

In addition, a corresponding back-projection of the updated $\mathbf{r}_c^+$ and $\mathbf{r}_c^-$ can be performed instead of a coding scheme. For example, the following back-projection algorithm can be used, which is applied to all updated reasoning probability pairs $\left(r_{c,k}^+, r_{c,k}^-\right)$ after each update step:

1. Clip $r_{c,k}^+$ and $r_{c,k}^-$ back into the interval $[0,1]$.
2. If $r_{c,k}^+ + r_{c,k}^- > 1$, then normalize both probabilities by the sum of $r_{c,k}^+ + r_{c,k}^-$.

4.2.2 Reasoning over a set of patch components

Suppose a set of patch components $\mathcal{K}$. To measure the detection probability $d_k(\mathbf{x})$ between an input $\mathbf{x} \in \mathbb{R}^{n_x}$ and a patch component $\boldsymbol{\kappa}_k \in \mathbb{R}^{n_\kappa}$, we extend $d_k(\mathbf{x})$ to a sliding operation (Ghiasi-Shirazi, 2019). For instance, this could be done by cropping a patch of size n_κ from $\mathbf{x}$ at position i and calculating the detection probability to the patch component $\boldsymbol{\kappa}_k$. If we do this for all components, we obtain a detection possibility vector $\mathbf{d}_i(\mathbf{x})$ of size $\#\mathcal{K}$. Moreover, by sliding this operation over all positions $i \in \{1, \dots, v_d\}$ accordingly, we get the detection possibility vector at each position, where v_d is the spatial dimension after the sliding operation. We stack all $\mathbf{d}_i(\mathbf{x})$ together and obtain a *detection possibility stack* (extracted spatial DP) $\mathbf{d}(\mathbf{x}) \in [0,1]^{v_d \times \#\mathcal{K}}$.[8] Following Ghiasi-Shirazi (2019), the most commonly used dissimilarity measures can be efficiently extended to sliding operations by linear convolution operations. However, the reasoning Equation (4.5) can only handle one detection probability per component so that the reasoning process must be redefined.

Downsampling

A simple approach is to downsample the detection possibility stack over the spatial dimension v_d such that the output is a detection possibility vector and Equation (4.5) can be applied. This can be achieved by applying global pooling techniques like global max pooling (computation of the component-wise maximum detection probability).

[8]The i-th element of a stack $\mathbf{s} \in \mathbb{R}^{m \times n}$ is a vector $\mathbf{s}_i \in \mathbb{R}^n$ and is therefore shown in bold too. We do not use a new symbol for stacks to keep the equations generic with the one symbol. However, we make sure that the meaning is clear from the respective content.

With this operation, we consider only the best match of each patch component with the input. This strategy is implicitly used in the introductory example, see Figure 4.1.

Spatial reasoning

Another approach is to extend the reasoning process to a calculation over the entire extracted spatial DP, which we call *spatial reasoning.* For this purpose, we retain the detection possibility stack $\mathbf{d}(\mathbf{x})$ of size $v_d \times \#\mathcal{K}$, and we learn a reasoning process for each spatial position $i \in \{1, \ldots, v_d\}$ resulting in a total number of v_d processes. To calculate the class hypothesis probabilities $p_c(\mathbf{x})$, Equation (4.5) is redefined to be a weighted mean over the reasoning at each spatial position $i \in \{1, \ldots, v_d\}$:

$$p_c(\mathbf{x}) = \sum_{i=1}^{v_d} \alpha_{c,i} \frac{(\mathbf{d}_i(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{r}_{c,i}^{+} + (\mathbf{1} - \mathbf{d}_i(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{r}_{c,i}^{-}}{\mathbf{1}^{\mathrm{T}} \cdot \left(\mathbf{r}_{c,i}^{+} + \mathbf{r}_{c,i}^{-}\right)}, \tag{4.14}$$

where $\alpha_{c,i} \in [0,1]$ with $\sum_i \alpha_{c,i} = 1$ are the trainable or non-trainable *class-wise pixel probabilities* that reflect the importance of each pixel position i.[9] The reasoning possibility vectors for position i are denoted by $\mathbf{r}_{c,i}^{\pm} \in [0,1]^{\#\mathcal{K}}$ and are stacked to *reasoning possibility stacks* $\mathbf{r}_c^{\pm} \in [0,1]^{v_r \times \#\mathcal{K}}$. We use v_r to denote the spatial dimension of the reasoning stack. For spatial reasoning, $v_r = v_d$ must hold.

By this reasoning strategy, we force the model to reason over the complete possibility stack. If, for example, $\alpha_{c,i} = \frac{1}{v_d}$ for all i, then the model has to reason perfectly over each spatial position i to achieve an output probability of $p_c(\mathbf{x}) \approx 1$. Perfectly means that the class hypothesis probability at position i must be close to one for each i. This forces the CBC to understand the entire spatial representation of the input because each pixel position i is *equally important.*

If the class-wise pixel probabilities $\alpha_{c,i}$ are trainable parameters of the CBC, then we apply the following coding scheme: By definition, it is required that $\alpha_{c,i} \in [0,1]$ and $\sum_i \alpha_{c,i} = 1$. This is achieved by learning encoded parameters $\tilde{\alpha}_{c,i} \in \mathbb{R}$ that can be decoded using a softmax activation to obtain the pixel probabilities $\alpha_{c,i}$:

$$\alpha_{c,i} = \frac{\exp(\tilde{\alpha}_{c,i})}{\sum_{i'} \exp(\tilde{\alpha}_{c,i'})}.$$

Now, the CBC can learn the importance of each pixel position i for the class decision of class c.

[9] One could use another strategy than a weighted mean to process the class hypothesis probabilities of each position i, for instance, take the maximum probability of all spatial positions i. Throughout this thesis, however, we focus on a weighted mean because the motivation for the spatial reasoning is that a CBC has to understand the input at each spatial position i instead of relying on a potentially weak reasoning over one position.

See Figure 4.3 for an illustration of an image processing CBC with Siamese CNN feature extractor for similarity learning and spatial reasoning applied.

Partial spatial reasoning

In the following, the most generic model of the presented reasoning processes is described. Each reasoning process (spatial reasoning or reasoning over full-size components) can be derived in terms of this generic model. In the initial description of spatial reasoning, it is assumed that the spatial dimension of the reasoning possibility stack v_r is equal to the spatial dimension of the detection possibility stack v_d. We relax this assumption to $v_r \leq v_d$. For example, in the case of image inputs, the detection possibility stack $\mathbf{d}(\mathbf{x})$ has a dimension of $v_d \times h_d \times \#\mathcal{K}$ and the reasoning possibility stack $\mathbf{r}_c^{\pm}$ has a dimension of $v_r \times h_r \times \#\mathcal{K}$ with $v_r \leq v_d$ and $h_r \leq h_d$. Now, instead of calculating only one class hypothesis probability $p_c(\mathbf{x})$ as in spatial reasoning, we slide the spatial reasoning process, see Equation (4.14), with the smaller reasoning possibility stack over the detection possibility stack. The result is a class hypothesis probability map of size $v_p \times h_p$. Additionally, we collect the probabilities of all classes into a stack of size $v_p \times h_p \times \#\mathcal{C}$. As a final step, we downsample the possibility stack to a class hypothesis possibility vector, for instance, by applying global max pooling.

The idea behind this approach is that we search for a match of the learned class DP (expressed by $\mathbf{r}_c^{\pm}$) in the extracted spatial DP $\mathbf{d}(\mathbf{x})$ at different positions. This allows us to handle local shifts and thus enables us to detect objects at different positions. In this case, the reasoning process is only applied over a part of the detection possibility stack and, therefore, we call it *partial spatial reasoning*. This can be efficiently implemented by linear convolution operations as the reasoning process of Equation (4.5) corresponds to an affine transformation, see Section 4.2.4.

4.2.3 Multiple components and reasoning strategies

Similar to multiple prototypes per class in LVQ algorithms, it is obvious to create similar extensions of CBCs. Since CBCs are two-phase classifiers—calculation of the extracted DP followed by reasoning—there are three ways to include multiple-concepts, which we describe below.

Multiple component versions

Usually, we equip each component index k with exactly one component $\boldsymbol{\kappa}_k$. If the reasoning depends on this component, then this component has to match the input regarding the detection probability $d_k(\mathbf{x})$ of the desired reasoning. Depending on the variations that may occur in the input, this can be difficult to model with $d_k(\mathbf{x})$. For example, suppose we have image inputs that show exactly one object per image and the task is to classify the images according to their object class. If the objects in the images appear shifted or rotated, then it is unlikely that this is adequately modeled by the detection probability functions of Equation (4.2) and Equation (4.3).

To overcome this problem, we propose to learn instead of one component for each component index k multiple component versions $\boldsymbol{\kappa}_{k,i}$ and select the best matching component $\boldsymbol{\kappa}_k$ by a respective downsampling over i. For instance, assume we have three versions for each component index k and select the most appropriate component based on the highest detection probability value:

$$\boldsymbol{\kappa}_k = \operatorname*{arg\,max}_{\mathbf{y}\in\{\boldsymbol{\kappa}_{k,1},\boldsymbol{\kappa}_{k,2},\boldsymbol{\kappa}_{k,3}\}} d(\mathbf{x},\mathbf{y}). \tag{4.15}$$

Note that choosing an appropriate downsampling strategy is not trivial because we may induce an unintended bias in the CBC. By the downsampling strategy of Equation (4.15), we bias the model towards positive reasoning since the detection probability function $d_k(\mathbf{x})$ prefers larger output values. Consequently, another downsampling strategy is to select $\boldsymbol{\kappa}_k$ by the minimum instead of the maximum.

Multiple reasoning processes

In the reasoning processes defined above, we learn exactly one set of reasoning probabilities (the class DP) for each class c. This can be extended to multiple reasoning probabilities for each class c. For example, consider a classification task similar to Figure 4.1, where the set of patch components is capable of modeling both the American and Latin writing styles of a one. However, by limiting the model to one class DP per class, the model has to learn a trade-off between the two concepts. By learning multiple class DPs, this can be overcome so that both concepts can be learned.

Assuming that $p_{c,i}(\mathbf{x})$ is the class hypothesis probability of class c and concept i (the i-th class DP)—that is, $p_{c,i}(\mathbf{x})$ is calculated with the i-th reasoning possibility vectors $\mathbf{r}_{c,i}^{+}$, $\mathbf{r}_{c,i}^{0}$, and $\mathbf{r}_{c,i}^{-}$ of class c[10]—we compute the class hypothesis probability of class c by

$$p_c(\mathbf{x}) = \max_i \{p_{c,i}(\mathbf{x})\}.$$

Multiple CBC processes

The combination of the two strategies described above is the learning of multiple CBC processes. This means that we assign a special set of components to the i-th

[10]If there are more trainable parameters (e. g., $\alpha_{c,i}$ in spatial reasoning), then these parameters can be extended to multiple versions as well.

reasoning probabilities of all classes. More precisely, the i-th set of components belongs exclusively to the i-th reasoning possibility vectors $\mathbf{r}_{c,i}^{+}$, $\mathbf{r}_{c,i}^{0}$, and $\mathbf{r}_{c,i}^{-}$. The result is that the multiple reasoning processes are decoupled with respect to the sets of components. Therefore, this strategy learns multiple CBC processes. Similar to multiple reasoning processes, the class hypothesis possibility vector is computed by calculating class-wise the maximum class hypothesis probability.

4.2.4 General remarks

This section contains some remarks about the proposed CBC architecture. In the first part, we discuss similarities between the proposed CBC architecture and NNs. The results are useful to conclude that the CBC architecture is a kind of two-layer NN, which can be used as the final layers of almost every classification NN. After that, we present theoretical results about the classification behavior of the class hypothesis probability Equation (4.5). In the third part, we discuss the derived learning rules regarding attraction and repulsion forces. Following this, we describe the initialization process of a CBC and suitable data normalization techniques. The fifth part shows how the class hypothesis probability vector can be transformed into a probability vector in terms of the probability tree diagram. Finally, we discuss why we do not train with a loss function based on the relative signed probability gap.

Interpretation of CBCs as NNs

If we consider Equation (4.6) to calculate $p_c(\mathbf{x})$, we can conclude that the computation of a class hypothesis probability is an affine transformation of $\mathbf{d}(\mathbf{x})$:

$$p_c(\mathbf{x}) = (\mathbf{d}(\mathbf{x}))^{\mathrm{T}} \left(\bar{\mathbf{r}}_c^{+} - \bar{\mathbf{r}}_c^{-}\right) + \mathbf{1}^{\mathrm{T}} \bar{\mathbf{r}}_c^{-}.$$

Furthermore, if we stack all the vectors $\bar{\mathbf{r}}_c^{+} - \bar{\mathbf{r}}_c^{-}$ to a matrix and all the scalars $\mathbf{1}^{\mathrm{T}} \bar{\mathbf{r}}_c^{-}$ to a vector, we can conclude that this operation is equivalent to the operation of a fully connected layer with $\#\mathcal{K}$ input units and $\#\mathcal{C}$ output units. Additionally, the detection probability function $\mathbf{d}(\mathbf{x})$ can be interpreted as a kind of fully connected layer as well. For instance, if we consider the cosine similarity defined in Equation (4.2) and assume that the inputs are normalized to a Euclidean norm of one, then the normalized components $\frac{\boldsymbol{\kappa}_k}{\|\boldsymbol{\kappa}_k\|_2}$ are the weights of the linear transformation and φ is the activation function of the layer. A similar interpretation holds for reasoning over patch components where the respective NN layer is a convolution layer instead of a fully connected layer.

In summary, the CBC architecture can be understood as a two-layer NN. The output layer has the identity activation and both layers are implicitly constrained.

This constraint is defined by the probabilistic CBC framework and can generally be applied to the last two layers of suitable NNs. The detection probability layer is an extension of a convolution layer with the requirement to measure the detection of convolution filters called components, expressed in probabilities.[11] Additionally, the final reasoning layer is still affine but follows a special implicit constraint defined by the probability model. The overall output is a probability value for each class without an artificial squashing. That this framework can be applied to deeper NN architectures is part of Section 4.3 and the evaluation presented in Section 4.5.

Analysis of the class hypothesis probability

Lemma 4.1. *Let $\mathbf{x} \in \mathbb{R}^{n_x}$ be an input sample, d_{min} be the minimum probability of detection or no detection across all components, that is,*

$$d_{min} = \min\left\{d_k\left(\mathbf{x}\right), 1 - d_k\left(\mathbf{x}\right) \mid k = 1, 2, \ldots, \#\mathcal{K}\right\},$$

and d_{max} be the maximum probability of detection or no detection across all components, that is,

$$d_{max} = \max\left\{d_k\left(\mathbf{x}\right), 1 - d_k\left(\mathbf{x}\right) \mid k = 1, 2, \ldots, \#\mathcal{K}\right\}.$$

Moreover, assume $\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_c^+ + \mathbf{r}_c^-\right) > 0$ for all $c \in \mathcal{C}$. Then the following holds for all $c \in \mathcal{C}$:

$$d_{min} \leq p_c\left(\mathbf{x}\right) \leq d_{max}.$$

Proof. Since d_{max} is the maximum element, it follows that

$$\begin{aligned} p_c\left(\mathbf{x}\right) &= \frac{\left(\mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \mathbf{r}_c^+ + \left(\mathbf{1} - \mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \mathbf{r}_c^-}{\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_c^+ + \mathbf{r}_c^-\right)}, \\ &\leq \frac{d_{max} \cdot \mathbf{1}^{\mathrm{T}}\mathbf{r}_c^+ + d_{max} \cdot \mathbf{1}^{\mathrm{T}}\mathbf{r}_c^-}{\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_c^+ + \mathbf{r}_c^-\right)}, \\ &\leq d_{max}. \end{aligned}$$

Additionally, since d_{min} is the minimum element, it follows that

$$\begin{aligned} p_c\left(\mathbf{x}\right) &= \frac{\left(\mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \mathbf{r}_c^+ + \left(\mathbf{1} - \mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \mathbf{r}_c^-}{\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_c^+ + \mathbf{r}_c^-\right)}, \\ &\geq \frac{d_{min} \cdot \mathbf{1}^{\mathrm{T}}\mathbf{r}_c^+ + d_{min} \cdot \mathbf{1}^{\mathrm{T}}\mathbf{r}_c^-}{\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_c^+ + \mathbf{r}_c^-\right)}, \\ &\geq d_{min}, \end{aligned}$$

which completes the proof. □

[11] This includes fully connected layers as they can be modeled by convolution layers.

This lemma states that the class hypothesis probability is bounded by the detection probability with respect to an input. Consequently, it states that the classifier cannot be more discriminative than the detection probability function. Hence, if we want a truly discriminative classifier, we need a highly discriminative detection probability function.

Lemma 4.2. *Let $\mathbf{x} \in \mathbb{R}^{n_x}$ be an input sample, d_{min} and d_{max} defined as in the previous Lemma 4.1, and also $\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_c^+ + \mathbf{r}_c^-\right) > 0$ for all $c \in \mathcal{C}$. Then the following holds for all $k \in \{1, 2, \dots, \#\mathcal{K}\}$: $d_{max} = p_c(\mathbf{x})$ if and only if*

$$r_{c,k}^+ \begin{cases} = 0 & \text{if } d_k(\mathbf{x}) < d_{max}, \\ \geq 0 & \text{otherwise}, \end{cases} \qquad r_{c,k}^- \begin{cases} = 0 & \text{if } 1 - d_k(\mathbf{x}) < d_{max}, \\ \geq 0 & \text{otherwise}, \end{cases} \tag{4.16}$$

and $d_{min} = p_c(\mathbf{x})$ if and only if

$$r_{c,k}^+ \begin{cases} = 0 & \text{if } d_k(\mathbf{x}) > d_{min}, \\ \geq 0 & \text{otherwise}, \end{cases} \qquad r_{c,k}^- \begin{cases} = 0 & \text{if } 1 - d_k(\mathbf{x}) > d_{min}, \\ \geq 0 & \text{otherwise}. \end{cases} \tag{4.17}$$

Proof. We prove the lemma for $d_{max} = p_c(\mathbf{x})$. By definition of $p_c(\mathbf{x})$, this is equivalent to

$$\underbrace{\left(d_{max} \cdot \mathbf{1} - \mathbf{d}(\mathbf{x})\right)^{\mathrm{T}} \mathbf{r}_c^+}_{\geq 0} + \underbrace{\left(d_{max} \cdot \mathbf{1} - (\mathbf{1} - \mathbf{d}(\mathbf{x}))\right)^{\mathrm{T}} \mathbf{r}_c^-}_{\geq 0} = 0$$

and, moreover, equivalent to the following two conditions:

$$\begin{aligned} 0 &= \left(d_{max} \cdot \mathbf{1} - \mathbf{d}(\mathbf{x})\right)^{\mathrm{T}} \mathbf{r}_c^+, \\ &= \sum_k \underbrace{\left(d_{max} - d_k(\mathbf{x})\right)}_{\geq 0} \underbrace{r_{c,k}^+}_{\geq 0}, \end{aligned}$$

and

$$\begin{aligned} 0 &= \left(d_{max} \cdot \mathbf{1} - (\mathbf{1} - \mathbf{d}(\mathbf{x}))\right)^{\mathrm{T}} \mathbf{r}_c^-, \\ &= \sum_k \underbrace{\left(d_{max} - (1 - d_k(\mathbf{x}))\right)}_{\geq 0} \underbrace{r_{c,k}^-}_{\geq 0}. \end{aligned}$$

Since all values in the products are greater than or equal to zero, we can conclude that the sums are zero if and only if at least one of the factors in each product is zero. This is equivalent to the conditions for $r_{c,k}^+$ and $r_{c,k}^-$ in the lemma. A similar proof holds for $d_{min} = p_c(\mathbf{x})$. □

Because $\mathbf{1}^{\mathrm{T}}\left(\mathbf{r}_c^+ + \mathbf{r}_c^-\right) > 0$ has to be valid, the lemma states that to reach the extreme values of $p_c(\mathbf{x})$ at least one of the reasoning possibility vectors has to be nontrivial and orthogonal to the corresponding shifted detection possibility vector. Surprisingly, the lemma does not state that the nontrivial reasoning probabilities have to be equal to one, which means that the extreme values of $p_c(\mathbf{x})$ can be reached with *small* reasoning probabilities.

Interpretation of the learning rules

If we consider the Equations (4.9), (4.10), and (4.11), the gradients can be related to attraction and repulsion forces similar to LVQ methods, see Section 2.2. These forces attract and repel the reasoning factors and the components. For example, to update the positive reasoning vector $\mathbf{r}_{c^+}^+$ of the correct class c^+, the gradient of Equation (4.9) reinforces all probabilities to which

$$d_k(\mathbf{x}) - p^+(\mathbf{x}) > 0$$

applies. Thus, if the component $\boldsymbol{\kappa}_k$ has a greater detection probability than the predicted class hypothesis probability $p^+(\mathbf{x})$, then the reasoning probability $r_{c^+,k}^+$ is increased. Otherwise, $r_{c^+,k}^+$ is decreased. This observation is in accordance with Lemma 4.1 and Lemma 4.2 because as long as $p^+(\mathbf{x})$ is not equal to d_{max} and, hence, has not reached the optimal value, the update increases and decreases the reasoning probabilities such that $p^+(\mathbf{x})$ converges to the limit and the reasoning probabilities to the solution of Equation (4.16). The interpretation of the update of the negative reasoning probabilities $\mathbf{r}_{c^+}^-$, see the gradient of Equation (4.10), is the same as the previous explanation except that the complementary event $1 - d_k(\mathbf{x})$ of $d_k(\mathbf{x})$ is used to decide whether a reasoning probability has to be increased or decreased. Moreover, the gradient scaling factors of Equation (4.12) normalize the gradients regarding the total amount of probability mass used to reason for the class. For the highest probable incorrect class c^-, the interpretations are reversed and $p^-(\mathbf{x})$ is updated towards d_{min} so that $\mathbf{r}_{c^-}^\pm$ converges to the solution of Equation (4.17).

As mentioned in Lemma 4.2, the nontrivial entries of $\mathbf{r}_{c^\pm}^\pm$ have to be nonzero to reach an extreme value of $p(\mathbf{x})$. In general, we update the values with a non-infinitesimal small learning rate so that the SGD update usually oscillates around the extreme values. Therefore, the SGD will tend to find a stable solution in the sense of a fixed point iteration. This is achieved if the nontrivial entries are as large as possible and thus as close as possible to a value of one. In this case, a small deviation from the optimal value of zero has the least effect on the output probability. Consequently, the training will tend to generate zero-one vectors for the reasoning possibility vectors.

If we consider Equation (4.11), we can see that the components are attracted by the input if

$$\xi^{-}\left(r^{+}_{c^{-},k}-r^{-}_{c^{-},k}\right)<\xi^{+}\left(r^{+}_{c^{+},k}-r^{-}_{c^{+},k}\right)$$

and repelled otherwise. Thus, the reasoning probabilities determine whether the component is adjusted to be more similar or dissimilar to the input. For instance, if the correct class relies on positive reasoning and the highest probable incorrect class relies on negative reasoning, then the component is attracted to increase the output probability of the correct class and decrease the output probability of the highest probable incorrect class. If both classes reason over the component in the same way, then the update is performed in favor of the more important class.

Initialization of CBCs and data normalization

Similar to LVQ models, a correct initialization of CBCs is important to achieve fast and stable convergence during training. As we will see in the evaluations, it is possible to train CBCs based on randomly initialized components, even if this is slower and requires a more careful selection of the learning rate. However, due to the clear interpretation of CBCs, this random initialization scheme is not optimal in most cases. Usually, we have access to the training dataset so that we can use training samples as initialization. Furthermore, we could also calculate averages with a k-means algorithm and use these to initialize the components. In both cases, the components are initialized to be similar to a certain class so that we could initialize the respective positive reasoning probabilities with a bias towards these components while keeping the positive reasoning probabilities significantly lower for all other components. The negative reasoning probabilities can be set to values close to zero. The remaining probability mass for each component is then concentrated in the respective indefinite reasoning probabilities. Note that these initialization schemes can be extended to patch components and spatial reasoning.

In addition to a suitable initialization, the components and data samples must be normalized accordingly, as is common in ML. The gradient of the cosine similarity, according to Equation (4.2), with respect to a component is

$$\nabla_{\boldsymbol{\kappa}} d\left(\mathbf{x},\boldsymbol{\kappa}\right)=\varphi'\left(\cos\left(\mathbf{x},\boldsymbol{\kappa}\right)\right)\left(\frac{\mathbf{x}}{\left\|\mathbf{x}\right\|_{E}\left\|\boldsymbol{\kappa}\right\|_{E}}-\cos\left(\mathbf{x},\boldsymbol{\kappa}\right)\frac{\boldsymbol{\kappa}}{\left\|\boldsymbol{\kappa}\right\|_{E}^{2}}\right).$$

If the components or the data are not adequately normalized, the similarity could suffer from exploding or vanishing gradients due to the division with the Euclidean norms. A similar effect could occur if a similarity function based on the Euclidean distance is used, for instance, see Equation (4.3). Without a proper normalization of the data and scaling of the distance by the variance σ, the detection probability will always be close to zero so that the gradients vanish. For the cosine similarity, the normalization of all input values to the unit interval worked well in the experiments. However, we cannot give a general recommendation for a good variance value σ or a normalization approach of the data for a similarity function based on the Euclidean distance since both are strongly data- and task-dependent.

Probability transformation of the class hypothesis possibility vector

For some applications, it is useful to have a class probability vector instead of a class hypothesis possibility vector—for instance, for the model training with a cross-entropy loss or a Kullback–Leibler divergence. The normalization into a class probability vector can be achieved by the derivation of the probability for a class c under the condition of an agreement A and importance I:

$$\begin{aligned} P\left(c \mid A, I, \mathbf{x}\right) &= \frac{P\left(c, A, I, \mathbf{x}\right)}{P\left(A, I, \mathbf{x}\right)}, \\ &= \frac{P\left(c, A \mid I, \mathbf{x}\right)}{P\left(A \mid I, \mathbf{x}\right)}, \\ &= \frac{P\left(A \mid I, \mathbf{x}, c\right) P\left(c\right)}{\sum_{c' \in \mathcal{C}} P\left(A \mid I, \mathbf{x}, c'\right) P\left(c'\right)}, \\ &= \frac{p_c\left(\mathbf{x}\right) P\left(c\right)}{\sum_{c' \in \mathcal{C}} p_{c'}\left(\mathbf{x}\right) P\left(c'\right)}, \end{aligned} \tag{4.18}$$

where $P\left(c\right)$ is the prior probability of class c. Hence, the transformation is obtained by dividing each class hypothesis probability by the sum of all class hypothesis probabilities. It should be noted that this normalization is derived using the probability tree diagram and is not the result of a naive normalization or squashing.

In general, this transformation is similar to the frequently applied softmax transformation in NNs except that we are not applying an exponential squashing. We will not further study the application of this transformation in this thesis for the following two reasons:

- A suitable estimate for $P\left(c\right)$ is usually not available.
- We consider a possibility output as more desirable because it enables an unbiased class probability comparison. For example, assume a network output where all $p_{c \neq c(\mathbf{x})}\left(\mathbf{x}\right)$ are zero and $p_{c(\mathbf{x})}\left(\mathbf{x}\right)$ is nonzero but arbitrarily small (i. e., $p_{c(\mathbf{x})}\left(\mathbf{x}\right) = \epsilon > 0$). The winner-takes-all rule applied to the probability and possibility output returns the correct class prediction in both cases, but should we trust these predictions? In case of a class probability vector, the output for the correct class would be $P\left(c\left(\mathbf{x}\right) \mid A, I, \mathbf{x}\right) = 1$. So the probability vector

would be a unit vector that could be interpreted as a certain classification decision. In contrast, the possibility output $\mathbf{p}(\mathbf{x})$ is close to a zero vector and predicts correctly an uncertain classification. In particular, the class hypothesis probability for the correct class is $p_{c(\mathbf{x})}(\mathbf{x}) = \epsilon$.

Training with a loss based on the relative signed probability gap

Similar to the relative distance difference $\mu(\mathbf{x})$ of GLVQ, see Equation (2.12), we could train CBCs with a loss based on the *relative* signed probability gap defined by

$$\frac{p^-(\mathbf{x}) - p^+(\mathbf{x})}{p^-(\mathbf{x}) + p^+(\mathbf{x})} \in [-1, 1]$$

instead of a loss based on the *absolute* signed probability gap $\nu(\mathbf{x})$, see Equation (4.7). The reason why we do not use this relative gap can be explained with the following example: Assuming we have $p^-(\mathbf{x}) = 0$ and $p^+(\mathbf{x}) = 0.1$, then the relative gap would be -1 and thus the minimum value. Now, suppose we have $p^-(\mathbf{x}) = 0.9$ and $p^+(\mathbf{x}) = 1$, in which case the relative gap would be approximately -0.05. However, both configurations have an absolute signed probability gap of -0.1. This example shows that such a transformation scales the same absolute signed probability gap differently depending on its position within the unit interval. Therefore, such a transformation leads to a biased optimization towards class hypothesis probabilities close to zero.

4.3 Joint training with a trainable feature extractor

In the CBC architectures described above, we assumed that the input and the components are processed directly by the detection probability function. This can be extended by a feature extraction step before we measure the detection probability. Together with the detection probability layer $\mathbf{d}(\mathbf{x})$, this is comparable to similarity learning through a Siamese NN (Bromley, Guyon, LeCun, Säckinger, & Shah, 1994). A Siamese NN architecture is an architecture where we apply the same transformation $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^{m_x}$ to more than one input at the same time to obtain the output—as the name suggests, $\mathbf{f}$ is an NN.[12] For CBC architectures, this similarity learning can be formalized by

$$d_k(\mathbf{x}) = d(\mathbf{f}(\mathbf{x}), \mathbf{f}(\boldsymbol{\kappa}_k)),$$

where we require that $\mathbf{f}(\mathbf{x}) = \mathbf{f}(\boldsymbol{\kappa}_k)$ implies $d_k(\mathbf{x}) = 1$. If the feature extractor has trainable weights, then we can learn them in parallel with the weights of the

[12]Note again that we do not use $\mathbf{f}(\mathbf{x})$ as the symbol for a classifier function—for CBCs, this is $\mathbf{p}(\mathbf{x})$—but as the symbol for a feature extractor.

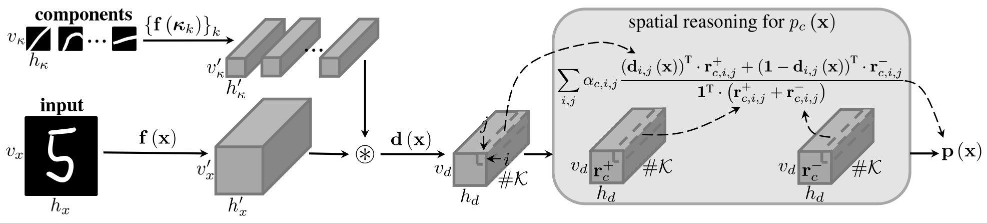

components
$\{\mathbf{f}(\boldsymbol{\kappa}_k)\}_k$
v_κ
h_κ
v'_κ
h'_κ
input
$\mathbf{f}(\mathbf{x})$
v_x
h_x
v'_x
h'_x
$\mathbf{d}(\mathbf{x})$
v_d
h_d
j
i
$\#\mathcal{K}$
spatial reasoning for $p_c(\mathbf{x})$
$\sum_{i,j} \alpha_{c,i,j} \frac{(\mathbf{d}_{i,j}(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{r}^+_{c,i,j} + (\mathbf{1} - \mathbf{d}_{i,j}(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{r}^-_{c,i,j}}{\mathbf{1}^{\mathrm{T}} \cdot (\mathbf{r}^+_{c,i,j} + \mathbf{r}^-_{c,i,j})}$
$\mathbf{r}^+_c$
$\mathbf{r}^-_c$
$\mathbf{p}(\mathbf{x})$

Figure 4.3: Example CBC for image processing with patch components, a Siamese NN as feature extractor, and spatial reasoning.

CBC architecture. This concept of similarity learning is useful for handling larger variations within the data and training CBCs on more complex datasets.

We can use an arbitrary feature extractor for full-size components, while we have to make additional requirements for patch components. Suppose the feature extractor $\mathbf{f}$ processes different input sizes up to a minimum dimension (receptive field size) of n_0, as most CNNs do. Additionally, we define the set of patch components $\mathcal{K}$ with $n_x \geq n_\kappa \geq n_0$ so that $\mathbf{f}(\boldsymbol{\kappa}_k) \in \mathbb{R}^{m_\kappa}$, where $m_x \geq m_\kappa$. Then the techniques from Section 4.2.2 can be applied. For instance, in Figure 4.3, we visualize a CBC with a CNN feature extractor for image processing inputs and spatial reasoning. The input is an image of size $v_x \times h_x$, which is transformed by the CNN to the spatial size $v'_x \times h'_x$. In parallel, the Siamese CNN architecture processes the patch components of size $v_\kappa \times h_\kappa$ to size $v'_\kappa \times h'_\kappa$. After that, we calculate $d_k(\mathbf{x})$ as a sliding operation, denoted as $\circledast$. The result is a detection possibility stack (extracted spatial DP) of size $v_d \times h_d \times \#\mathcal{K}$, which is further used for spatial reasoning.

End-to-end training of a CBC with a Siamese feature extractor

A remarkable observation is that the components usually do *not* correspond to an element of the dataset. This is important for training CBCs that use a Siamese setup to extract features from inputs and components: One path of the Siamese feature extractor processes the input samples and the other the components. For adequate learning, it is essential that the feature extractor is only updated by the dataset samples and not by the non-dataset samples (the components) because the feature extractor is supposed to extract useful features regarding the dataset distribution. Consequently, the gradient backflow from the components to the parameters of the feature extractor must be stopped. The gradient backflow from the components is only used to update the parameters of the components (if they are trainable). Although this is a slight deviation from the usual training procedure of a Siamese architecture, it improves the training of interpretable components significantly and is theoretically justified.

Definition of the components in the feature space

Instead of defining the components in the input space according to Definition 4.1, it is also possible to define the components directly in the feature space $\mathbb{R}^{m_\kappa}$ in order to reduce the Siamese architecture to an ordinary feedforward NN.[13] In this case, the components are similar to convolution filters of the penultimate layer. The advantage of this approach is that the computational effort to train the CBC is reduced as we only have to evaluate the feature extractor for one input. However, we lose the interpretability of the components in the input space. C. Chen et al. (2019) proposed a back-projection of learned prototypes by approximating $\mathbf{f}^{-1}$ to recover the interpretability in the input space—this technique can also be applied to components. Regardless of the exact network architecture, we can always project the components into the feature space after training to remove the computational overhead of the Siamese architecture during inference without losing interpretability.

4.4 Evaluation without a feature extractor

In this section, we compare CBCs with GLVQ, defined in Section 2.2.2, on MNIST—see Section 3.4.2 for a dataset description. The goal is to show that CBCs can classify with the same strategy as GLVQ but can also discover other classification principles.

Evaluation setup and models

We trained three models: a GLVQ model with 10 prototypes (one prototype per class) and two CBCs with 10 full-size components. The GLVQ model uses the squared Euclidean distance as dissimilarity and was trained with the identity activation function (i. e., $\phi(x) = x$) for the GLVQ loss according to Equation (2.15). The CBCs use the cosine similarity according to Equation (4.2) with $\varphi(x) = \mathrm{ReLU}(x)$ activation as detection probability function and were trained with the *margin loss* defined as Equation (4.8) with

$$\phi(\nu(\mathbf{x})) = \mathrm{ReLU}(\nu(\mathbf{x}) + \beta), \tag{4.19}$$

where β is the margin parameter. This loss function optimizes for a fixed margin (signed probability gap) β between the correct and highest probable incorrect class. We trained two CBC versions: one with $\beta = 0.3$ and one with $\beta = 0.1$. All three models were trained from scratch, and we initialized the reasoning probabilities with

[13] After back-propagating the component size into the input space, Definition 4.1 can be used to determine the component type.

uniform random noise values from the interval $[0, 1]$ and the component and prototype values with uniform random noise values from the interval $[0.45, 0.55]$. We selected a random initialization to challenge the CBC architecture.

The input spaces are defined over $[0, 1]$ and the datasets are normalized appropriately. The components and the prototypes are defined in the input space and constrained to it. Consequently, we applied a projected gradient descent learning. The reasoning probabilities were coded by Equation (4.13). All models were trained for 150 epochs with a batch size of 128 and the default Adam optimizer from KERAS with an initial learning rate of 0.003. During training, we monitored the validation loss and automatically adjusted the learning rate accordingly. If the validation loss has not decreased over five epochs, we have reduced the learning rate by a factor of 0.9. Moreover, we applied basic data augmentations in the form of random shifts of up to ± 2 pixels and random rotations of up to ± 15 degrees. Each model was trained three times and we calculated the mean and standard deviation of the accuracies.

We compare the different CBCs by calculating the average probability gap over a given test set. The *probability gap* for a sample $\mathbf{x}$ is defined as

$$\max\left\{p_c(\mathbf{x}) \mid c \in \mathcal{C}\right\} - \max\left\{p_{c'}(\mathbf{x}) \mid c' \in \mathcal{C}, c' \neq \arg\max_{c \in \mathcal{C}} p_c(\mathbf{x})\right\}. \tag{4.20}$$

This value differs from the signed probability gap $\nu(\mathbf{x})$ that is optimized by the CBC loss function because it is always positive and does not require a labeled input sample.

Accuracy results

The CBC with a margin β of 0.3 achieves a test accuracy of $(83.5 \pm 0.12)\,\%$ and slightly exceeds the GLVQ model, which has a test accuracy of $(81.7 \pm 0.22)\,\%$.[14] If the margin is reduced to 0.1, the accuracy increases to $(89.5 \pm 0.12)\,\%$ and the CBC significantly surpasses the learned GLVQ model.[15] We suspect that the reason for this is that the network with a smaller margin has the freedom to classify by weak decisions, while the margin of 0.3 forces the network to make robust decisions, which has disadvantages in terms of accuracy. In contrast, GLVQ is a hypothesis margin maximizer—see Section 3.5—so that the model is optimized for a trade-off between incorrect classifications and a large hypothesis margin across all samples. Therefore, the model is optimized for robust decisions.

Empirical evidence supporting this weak decision hypothesis can be found by calculating the average probability gap of the CBC classifications. While this gap

[14]The lower GLVQ accuracy than in the GTLVQ robustness evaluation in Section 3.5 is due to the box-constraint applied to the prototypes.

[15]Note that this accuracy is still less than the accuracy of the GTLVQ model in Section 3.5. However, the CBC has about 12 times fewer parameters than the GTLVQ model.

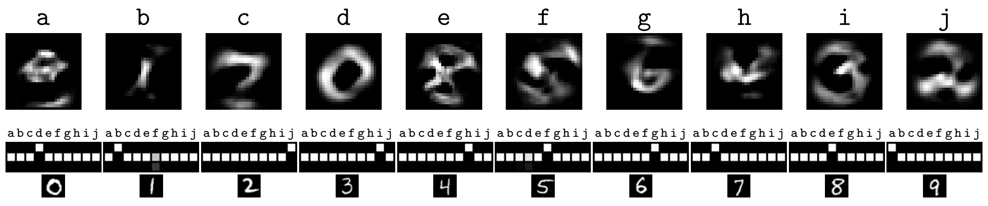


Figure 4.4: Learned reasoning process of a CBC without feature extractor, 10 components, and trained with a margin of $\beta = 0.3$ on MNIST. The top row shows the learned components and the bottom row shows the learned reasoning probabilities collected in reasoning matrices. In addition, the class of each reasoning matrix is indicated by the MNIST digit below. The top row of a reasoning matrix corresponds to $r^+_{c,k}$, the middle row to $r^0_{c,k}$, and the bottom row to $r^-_{c,k}$. A white square represents a probability of one and a black square represents zero.

is 0.16 ± 0.11 for a margin of 0.3, it is only 0.1 ± 0.05 for a margin of 0.1. In this calculation, only the correctly classified images were taken into account so that the direct comparison is possible.

Interpretation of the reasoning with a margin of 0.3

In Figure 4.4, we show the learned reasoning process with a margin of $\beta = 0.3$. The network consistently converges towards the BMPP and turns the class-independent components into prototypical (class-specific) components. The learned components can be directly recognized as class-specific components as they resemble digit shapes. The BMPP is learned when the indefinite reasoning probabilities for all but one component are close to one and the remaining component is used for positive reasoning. Only for the class `1`, the CBC performs weak negative reasoning over the component `f`, which is indicated by the grayish color. It is a remarkable observation that the CBC consistently converges towards the BMPP without a regularization or constraint over the reasoning process. This behavior shows that the BMPP is a stable solution for a robust classification process.

The learned components are similar to the learned prototypes of the GLVQ model in Figure 4.5 (e. g., compare the strong component of the class `5` with the learned GLVQ prototype). This provides additional evidence to support the claim that a CBC can act similarly to a prototype-based classifier, but only if this is the best classification principle for the task. The differences between the learned prototypes and components are the result of the different similarity measures. If we use the function of Equation (4.3) as a detection probability function in the CBC, then the learned components cannot be distinguished from the learned prototypes.

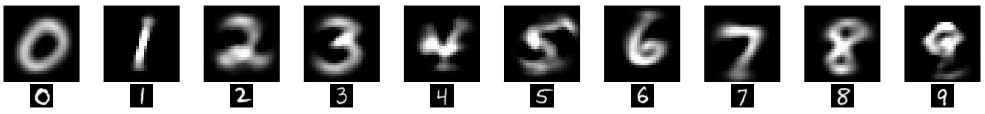


Figure 4.5: Prototypes learned by the GLVQ model. The class is indicated by the MNIST digit below.

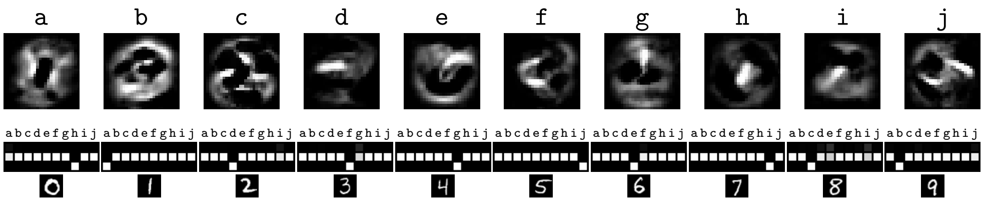


Figure 4.6: Learned reasoning process of a CBC without feature extractor, 10 components, and trained with a margin of $\beta = 0.1$ on MNIST. The figure description is the same as in Figure 4.4.

Interpretation of the reasoning with a margin of 0.1

In contrast to the learned reasoning process with a margin of $\beta = 0.3$, the reasoning process over a low margin is complementary to the BMPP, see Figure 4.6. Almost all the decisions are based on strong negative reasoning over one component. Only a few classes show weak positive reasoning over a few components. Hence, the learned components can be interpreted as negative examples of the classes: To make a correct classification of a given image, the strong (negative) component of the correct class should have the lowest detection probability in the image. The interpretation of negative samples is difficult because it is not an intuitive human reasoning process.

Although somewhat less intuitive, it is still possible to understand the classification decision. For instance, the reasoning process of the class `0` is based on strong negative reasoning over the component `h`. This component has a bright dot in the middle and the rest remains almost black. It is not difficult to see that this is indeed a negative example of the class `0` as the zero is the only digit that has no stroke in the middle. The same interpretation can be applied to the class `1` and the respective component. A more complex interpretation is the reasoning of the class `7` and the corresponding negative component `i`. The black top horizontal line shows a unique property of a seven and a five, the horizontal top stroke. In order to avoid confusion to a five, the component strongly highlights the area where a five has the transition from the vertical stroke to the arc—which means this should not be present. The property that the classification is sometimes based on finding unique parts and not understanding the whole input has generally been observed in some low margin experiments, even beyond MNIST.

4.5 Evaluation with a feature extractor

Depending on the experiment, different CNNs are used as feature extractors for the CBC architecture. However, all feature extractors except the one used in the ImageNet experiment have some settings in common. In particular, the convolution filters in a feature extractor are constrained to have a Euclidean norm of one[16] and are activated by the Swish activation function defined by

$$\operatorname{Swish}(x) = x \cdot \operatorname{sigmoid}_1(x), \tag{4.21}$$

for instance, see the study of Ramachandran, Zoph, and Le (2018). Additionally, the feature extractors do not contain batch normalization (Ioffe & Szegedy, 2015). This setting is chosen based on the results of the ablation study presented in Section 4.5.1. In the following, this setting of a CNN feature extractor in combination with the margin loss Equation (4.19), a margin parameter β of 0.3, and the cosine similarity with ReLU activation as detection probability function is referred to as the *default setting.* On the contrary, if a CNN feature extractor is used that is more in line with common CNN architectures (i. e., no convolution filter constraint, ReLU activation, and batch normalization), we refer to it as the *non-default setting.* Unless otherwise specified, CBCs with the non-default setting still use the margin loss and the ReLU activated cosine similarity as detection probability function. All other settings (evaluation setup, etc.) follow the description in Section 4.4.

Parallel to the cosine similarity as detection probability function, we studied in some experiments the use of the Euclidean distance activated by the negative exponential, see Equation (4.3). Using this approach with σ as a trainable parameter, a CBC with the Euclidean distance could achieve similar results to a CBC with the cosine similarity. However, the stability of the training is sensitive to the initialization of σ and tended to diverge so that we will not consider this similarity measure in the following evaluation.

4.5.1 MNIST: Ablation study

In the first experiments with trainable feature extractors, we found a strong variation in the interpretability of the components. This study shows how different settings of a CBC influence the interpretability of the components.

[16]The idea is similar to the weight normalization approach of Salimans and Kingma (2016). However, we do not encode the constraint into the weights but apply a projected gradient descent.

Experimental setup

We used 10 full-size components and the following four-layer CNN as basic feature extractor architecture:[17]

1. Convolution: 32 filters, kernel size 3×3, stride 1×1, bias, and no padding;
2. Convolution: 64 filters, kernel size 3×3, stride 1×1, bias, and no padding;
3. Max pooling: pool size and stride 2×2;
4. Convolution: 64 filters, kernel size 3×3, stride 1×1, bias, and no padding;
5. Convolution: 128 filters, kernel size 3×3, stride 1×1, bias, and no padding;
6. Max pooling: pool size and stride 2×2.

We trained all CBCs that can be constructed by combinations of the following parameters and their configurations (parameter description: configuration-1, configuration-2, and so on):

- activation function of convolution layers: ReLU, Swish;
- constraint of convolution filters to the Euclidean norm of one: true, false;
- application of batch normalization after the first max pooling layer: true, false;
- activation of the cosine similarity to provide the detection probability function: $\varphi(\mathbf{x}) = \mathrm{ReLU}(x)$, $\varphi(\mathbf{x}) = (\mathrm{ReLU}(x))^2$;
- squashing function ϕ of the CBC loss Equation (4.8): ELU[18] function, margin loss Equation (4.19) with $\beta = 0.1$, margin loss with $\beta = 0.3$, margin loss with $\beta = 0.5$, margin loss with $\beta = 0.7$, margin loss with $\beta = 0.9$.

Evaluation

After the networks were trained, we extracted the final components for each combination. We asked 10 human experts to give scores to the interpretability of components by visual examination. For each combination, we asked for exactly one score for all resulting components. The score values were described according to the following definitions and supplemented by the sample images in Figure 4.7.

[17]Note that the architecture is equivalent to the first four layers of the CNN in Section 3.5.2.

[18]The *Exponential Linear Unit* (ELU) is defined as

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x \geq 0, \\ \exp(x) - 1 & \text{otherwise,} \end{cases}$$

and is applied element-wise for vectors (Clevert, Unterthiner, & Hochreiter, 2016).

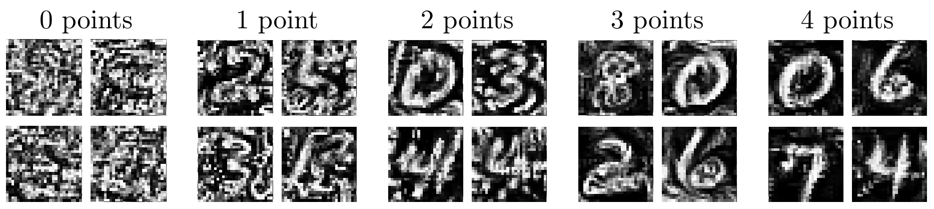


Figure 4.7: Example components for the visual evaluation of the interpretability. We excluded examples with a score of five because they are assumed to be real images.

0 points: The images resemble unstructured noise without visible digit shapes.

1 point: The images show something that resembles digits, but each image contains a lot of noise.

2 points: The images show something that resembles digits, but each image may also contain artifacts of other digits.

3 points: The images show real looking digits, but the background contains a lot of noise.

4 points: The images show real looking digits and the background contains almost no noise.

5 points: The images are indistinguishable from real inputs.

The total interpretability score is calculated as the arithmetic mean, and the maximum test accuracy is calculated as the maximum test accuracy over all runs and epochs.

Results

Since the overall goal of the ablation study is to find a CBC setting in which the components are learned to be interpretable by humans, the first evaluation criterion is the interpretability score. Two of the combinations achieved an interpretability score of 3.8—the highest score that was given to a model by a single expert was 4. The difference between these two models is the loss function with which the models were trained, see Figure 4.8. The first model was trained with the ELU function for ϕ, while the other model was trained with the margin loss and a margin value of $\beta = 0.3$. To decide which model performs better, we compared their final test accuracies. The model with the margin loss performed significantly better: $(99.27 \pm 0.1)\,\%$ average test accuracy with the margin loss and $(97.86 \pm 0.19)\,\%$ with the ELU function.

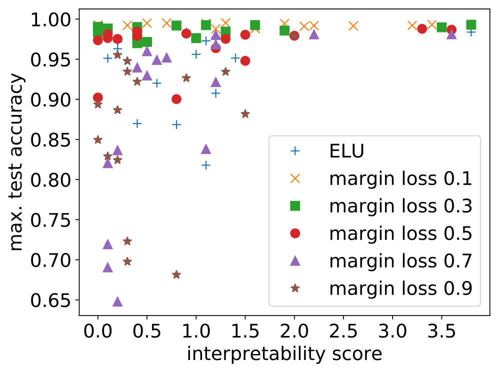


Figure 4.8: Impact of the squashing function ϕ on the interpretability score and the test accuracy.

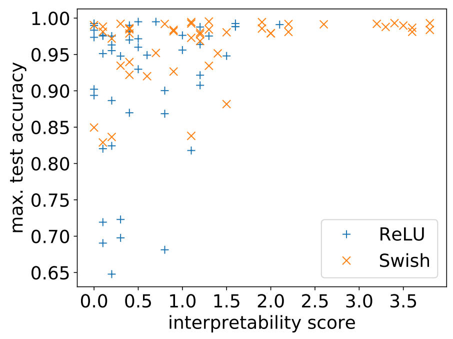
Therefore, the parameter combination of this point was chosen as the *default setting* for CBCs in all experiments. Interestingly, the whole evaluation showed that the accuracy is not correlated with the interpretability score, see Figure 4.8—that is, a model with non-interpretable components could achieve high accuracies.

Figure 4.9 shows the effect of the other parameters. There is no clear trend regarding the cosine similarity activation, but there is a clear trend with respect to the other parameters. In summary, the defined default setting clearly surpasses all other parameter combinations.

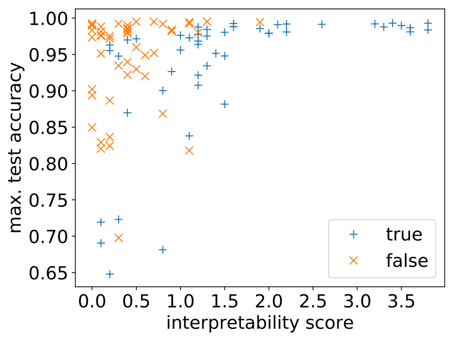
Analysis of the results

In general, the Euclidean constraint seems to have the strongest impact on the interpretability, see Figure 4.9. We observed the positive effect of this constraint during an experiment with convolution filters predefined as Sobel operator. The idea behind this constraint is to make all filters comparable and normalize them in such a way that each filter can distribute 100 % energy to its weights. Later we noticed that in the non-default setting (i. e., ReLU activation of convolution layers, batch normalization, and no constraint), the activations seem to explode with increasing network depth of the feature extractor—see Figure 4.10a and especially the different scales of the horizontal axes of the two plots. Regardless of this, the CBC with the non-default setting achieved an acceptable accuracy of $(98.86 \pm 0.26)\,\%$, which is slightly less than the accuracy with the default setting. To understand how the network in the non-default setting classifies, we make some conclusions about the CBC used:

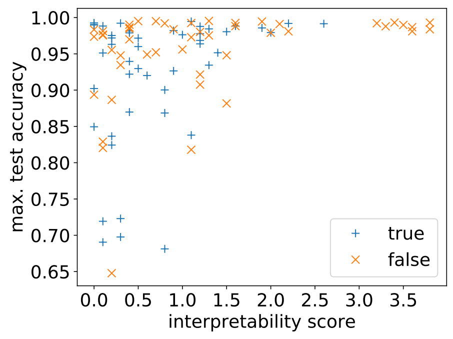
1. The cosine similarity is equivalent to a dot product (Euclidean inner product) after the two input vectors were normalized to a Euclidean norm of one.



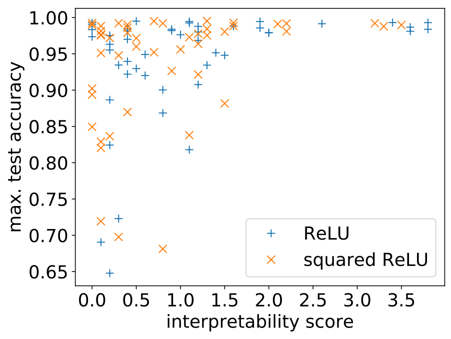
(a) Impact of the convolution activation.



(b) Impact of the Euclidean constraint.



(c) Impact of batch normalization.



(d) Impact of the cosine similarity activation.

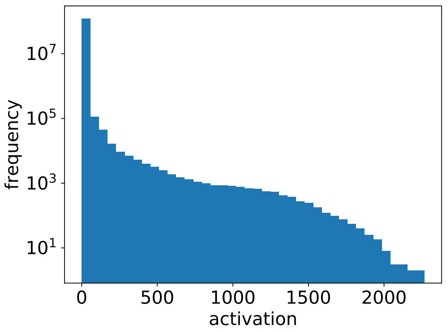
Figure 4.9: Results of the ablation study regarding the impact of the parameters convolution activation, Euclidean constraint, batch normalization, and cosine similarity activation on the interpretability score and the test accuracy.

2. If there are elements in the vector that are orders of magnitude larger than all the other elements, then after the normalization only the largest values are nonzero and only these values can contribute to a high output similarity.

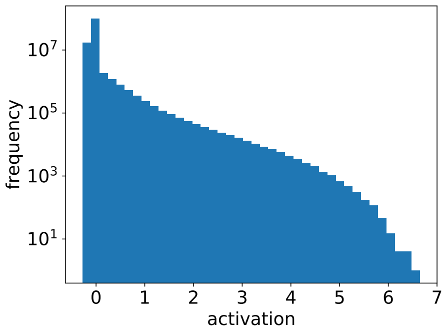
3. After the normalization, a high detection probability of $d_k(\mathbf{x}) \approx 1$ implies $\frac{\mathbf{f}(\mathbf{x})}{\|\mathbf{f}(\mathbf{x})\|_2} \approx \frac{\mathbf{f}(\boldsymbol{\kappa}_k)}{\|\mathbf{f}(\boldsymbol{\kappa}_k)\|_2}$ and, thus, that the input is similar to the k-th component.

4. If the model classifies by the BMPP using the prototypical component $\boldsymbol{\kappa}_k$ for class $c(\mathbf{x})$ and $p_{c(\mathbf{x})}(\mathbf{x}) \approx 1$ is valid, then $d_k(\mathbf{x}) \approx 1$.

Both networks classify by the BMPP with high output probabilities of $p_{c(\mathbf{x})}(\mathbf{x}) \approx 1$ for correct classifications using prototypical components—see Figure 4.11 for example components of both networks. The components of the CBC with the non-default



(a) CBC with the non-default setting.



(b) CBC with the default setting.

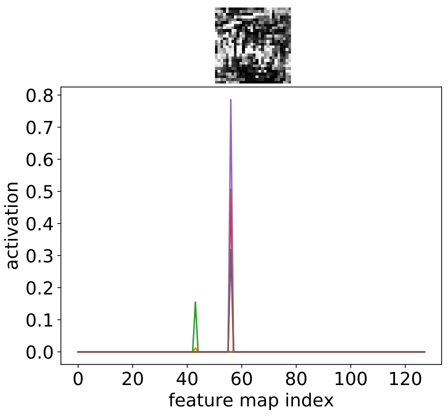
Figure 4.10: Distribution of the input values to the detection probability function (i. e., output activations of the feature extractor) with the non-default setting and the default setting shown as histograms. We calculated the histograms on the entire training dataset.

setting are not comprehensible to humans, whereas the components of the CBC with the default setting are. From Conclusion 4 and 3 we know that after the normalization an input has to be nearly identical to the strong prototypical component. Furthermore, we know from Conclusion 2 that only nonzero values contribute to the classification decision.

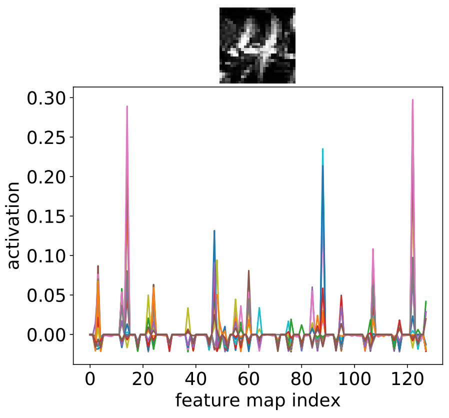
In the non-default setting, the most commonly used setting for CNNs, the nonzero values are those of high activation. Therefore, the model outputs high activations to suppress small values. In addition, the network relies on only a few significant features for each class—see Figure 4.11a for an example activation pattern after normalization. The complete classification of a four is based on a few features from two different feature maps.[19] By this trick, the network does not learn to extract useful features that can be used across classes but instead reinforces a particular feature map for each class. The activation pattern of a particular class is not understandable because the components cannot be interpreted.

In the default setting, the classification works differently. The network cannot overemphasize single features because the filters are normalized by the Euclidean norm so that the filters cannot have all weights zero (dead filters). Thus, the network has to incorporate each filter and, therefore, learns highly discriminative features. Looking at the activation patterns across all classes, we observe that the network learns features that are used across various classes and that it classifies by

[19]A *feature map* is the output generated by a filter of a convolution layer. Moreover, the *feature map index* refers to the filter index.



(a) CBC with the non-default setting.



(b) CBC with the default setting.

Figure 4.11: Activations of the feature maps of the last convolution layer in the feature extractor after Euclidean normalization. The activations are shown for the strong positive component of the class 4 for the non-default setting and the default setting. The colors within a plot correspond to the spatial dimensions and show the activation of different pixels within the 4×4 feature stack. Additionally, the corresponding learned components are displayed above the plots.

combining several features—see the activation pattern after the normalization shown in Figure 4.11b. We suspect that the components become human interpretable because the network with the default setting learns to decode the whole digit into a high-level representation, such as strokes and arcs. In contrast, the network with the non-default setting learns only a unique pattern, such as possibly the distribution of intensities.

The different losses do not show a clear trend regarding a specific setting except that a too large margin value in the margin loss is harmful to both interpretability and accuracy. We know from Xing et al. (2003) and Globerson and Roweis (2006) that optimizing a similarity measure (or metric) with a contrastive loss can cause the feature extractor to project all points of a given class into a single point. This effect is known as collapsing dimensions—see also Section 3.3.3—and is the main reason why we do not apply losses such as mean squared error or cross-entropy since they are based on the optimization towards one-hot labels. Furthermore, this effect leads to highly nonlinear regions in the learned mapping. We believe that the model needs space to distribute the feature vectors smoothly and well scattered to converge to the desired interpretability. A compromise between optimizing for correct classifications and increasing the margin is the loss induced by the ELU squashing. The ELU

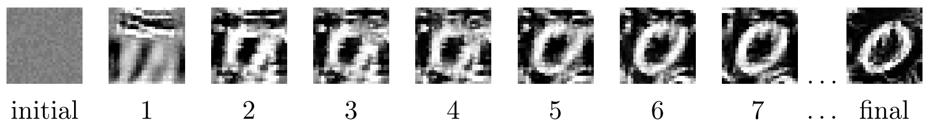


Figure 4.12: Evolution of a component over the number of epochs.

function has a derivative of one for incorrectly classified samples and slowly reduces the updates of already correctly classified images to avoid a dimensional collapse. The loss induced by the ELU squashing is an alternative to the margin loss and worked well in all experiments. However, the accuracies of the trained CBCs were always slightly below the accuracies of the CBCs trained with the margin loss. Additionally, we observe that even when optimized for a fixed margin of $\beta = 0.3$, the resulting margin is often much larger, as shown in Figure 4.16. This is an indicator that the feature vectors are smoothly distributed across the feature space.

So far, we do not know why batch normalization seems to be harmful to the interpretability. In contrast, we believe that the better behavior of Swish is due to the property that the function is differentiable everywhere and has almost always nonzero derivatives.

4.5.2 MNIST: Varying the number of components

In this section, we present the results of CBCs with 10, 9, and 8 full-size components and show that CBCs can handle a varying number of components and still classify well on MNIST. After that, we show that the sparseness of a CBC (fewer components) is associated with reduced performance since the models show a much smaller probability gap, see Equation (4.20). This means that the models start to classify by weaker decisions. For this experiment, the default setting for CBCs with the four-layer CNN feature extractor from the previous section is used, see Section 4.5.1.

CBC with 10 components

To give an impression on how the model learns the components, Figure 4.12 presents the evolution of a component starting from the initialization (random noise). After a few epochs, the rough shapes of the modeled digit are visible and the remaining epochs are used for fine-tuning to remove the background noise. In general, the training of the model is stable and consistently converges to high test accuracies of $(99.27 \pm 0.1)\,\%$. Interestingly, the model does not always converge clearly to the BMPP with prototypical components, as shown in Figure 4.13. It occurred quite frequently that the model solves the reasoning for the class `1` with strong negative

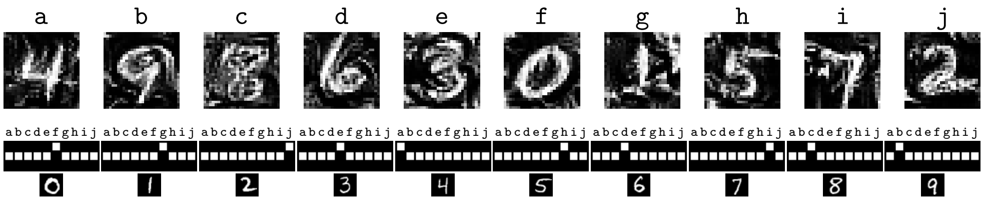


Figure 4.13: Learned reasoning process of a CBC with feature extractor and 10 components on MNIST. The figure description is the same as in Figure 4.4.

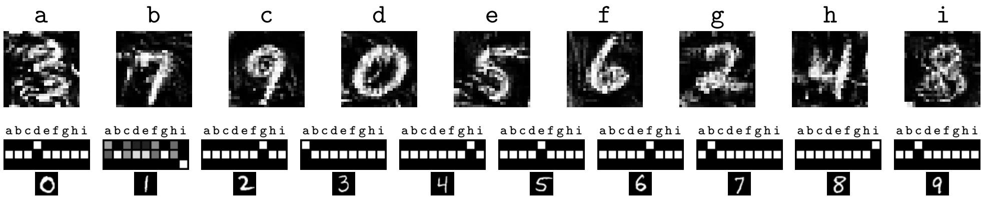


Figure 4.14: Learned reasoning process of a CBC with feature extractor and 9 components on MNIST. The figure description is the same as in Figure 4.4.

reasoning over one component. In this case, 9 components were clearly class-specific in a human-comprehensible sense and the strong component for the class `1` remained in a state similar to the strong negative component in Figure 4.14.

CBC with 9 components

To analyze if the network is able to classify by sharing the components across classes, we restricted the number of components to be smaller than the number of classes. Figure 4.14 shows the learned reasoning process of a CBC with 9 components. Similar to the 10 component version, the CBC learns to classify as many classes as possible by the BMPP. In the example, these are all classes except the class `1`, for which the CBC uses weak positive reasoning over the components `a`, `c`, `f`, and `h` but mostly depends on negative reasoning over the component `i`. This indicates that if an input image is classified as a one, the network requires it to not look like an eight. A comparison of the shapes of the digits one and eight supports this observation, the eight consists only of curved edges, while the one contains none. In addition, the one has on average the fewest white pixels, while the eight requires the most. This result shows again that by incorporating the negative and indefinite reasoning state, the CBCs can learn both the well-understood BMPP and unrestricted approaches beyond the intuitive classification principles by themselves. The test accuracy of the CBC is $(99.27 \pm 0.1)\,\%$ and, thus, close to state-of-the-art results.

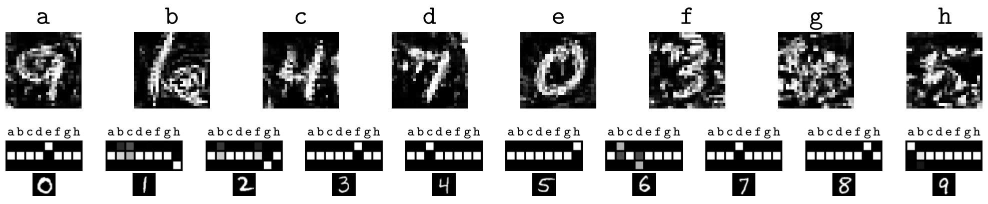


Figure 4.15: Learned reasoning process of a CBC with feature extractor and 8 components on MNIST. The figure description is the same as in Figure 4.4.

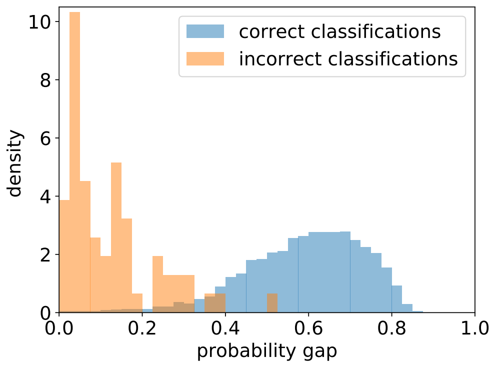
CBC with 8 components

If we train a CBC with 8 components and compare it to the models discussed before, we can observe a slight drop in the accuracy to $(99.07 \pm 0.1)\,\%$. To classify MNIST with only 8 components, the model learns to reason negatively over a couple of components. Similar to the other CBCs, a one is classified via negative reasoning. Moreover, the model classifies a two with negative reasoning too. In contrast to all models before, the CBC classifies a six with a combination of positive *and* negative reasoning. Although some of the components are easy to interpret, it is not easy to answer how the model uses the component `g` for positive reasoning for the class `8` and for negative reasoning for the class `2`.

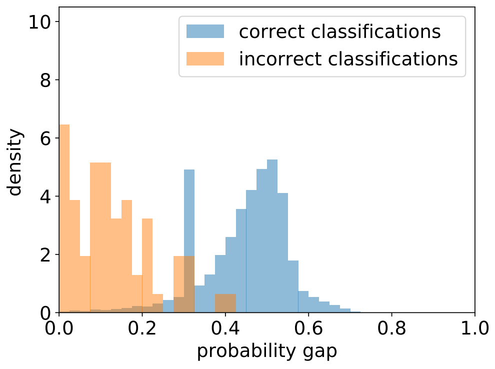
Comparison of the models

The reduced number of components has some drawbacks: First, the models become harder to interpret, as the comparison between the 10 component model in Figure 4.13 and the 8 component model in Figure 4.15 shows. Second, the networks begin to classify by weaker decisions when the number of components is reduced. However, this is not directly reflected in the accuracy, which is still above 99 % for all three models.

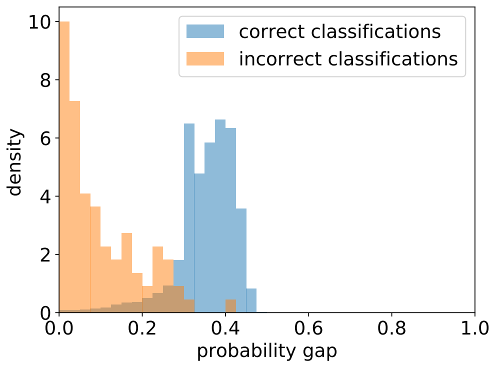
All networks are optimized with the margin loss in such a way that a probability margin of 0.3 should arise between the correct and the highest probable incorrect class. However, the trained CBC with 10 components has an average probability gap of 0.59 ± 0.14 over correctly classified images. This shows that even though we only optimize for a margin of 0.3, the networks can become more discriminative by themselves and distribute the probabilities over the whole range if possible, see Figure 4.16a. Furthermore, this figure clearly shows that incorrectly classified images are classified with high uncertainty because the probability gap is much smaller. This indicates that the probabilities for the runner-up class and the predicted class are almost equal. In other words, the model is not sure of its classification decision.



(a) CBC with 10 components.



(b) CBC with 9 components.



(c) CBC with 8 components.

Figure 4.16: Distributions of probability gaps of incorrectly and correctly classified images by CBCs on MNIST with different numbers of components. We visualize discrete approximations of the continuous density functions.

If we decrease the number of components in the CBCs, we observe that the probability gap of correctly classified images becomes smaller, see the shift of the distribution in Figure 4.16b and Figure 4.16c. At the same time, the distribution of incorrectly classified images does not change that much. Overall, the densities clearly show that the decreased number of components reduces the credibility of the classification decision.

4.5.3 MNIST: Initial robustness and rejection evaluation

During the ablation study, we found that for some settings of the feature extractor, the learned components were easier to interpretable than with other settings. Interestingly, the network was able to achieve an almost perfect classification with both interpretable and non-interpretable components. Since the CBCs of the ablation study are based only on positive reasoning by the BMPP and the components are defined in the same space as the input, we know that the components should be interpretable if $p_{c(\mathbf{x})}(\mathbf{x}) \approx 1$ because this implies $d_k(\mathbf{x}) \approx 1$ for the prototypical component $\boldsymbol{\kappa}_k$. Therefore, we make the following hypothesis: If the CBC converged to the BMPP, but the learned components are not interpretable even though $p_{c(\mathbf{x})}(\mathbf{x}) \approx 1$, then it is safe to assume that the network could be susceptible to adversarial attacks.

We performed a robustness evaluation to provide empirical evidence that this hypothesis is correct. Additionally, based on these results, we show that the distribution of class hypothesis probabilities of a CBC can be used to detect whether an image has been manipulated. As we show later, this property can be used for rejection strategies. The same property is observed with GTSRB on real-world adversarial examples, see Section 4.5.5.

Evaluation setup and models

Two CBCs and two CNNs are used for the evaluation. The first CBC architecture corresponds to the CBC with the default setting from the ablation study in Section 4.5.1 with a high interpretability score of 3.8 and is called CBC-4 in the following. The second CBC architecture corresponds to the CBC with the non-default setting that was also used in the ablation study. This model achieved an interpretability score of 0.4 and is called CBC-0.

The two CNNs are constructed by taking the feature extractor networks of the CBCs and adding two fully connected layers on top. The fully connected layers have 512 and 10 units and are separated by a dropout layer with a dropout rate of 0.5. The activation function of the first fully connected layer is the same as that of the convolution layers, while for the final fully connected layer the softmax activation function is used to generate a probability vector. Both CNNs were trained with

Table 4.1: Results of the robustness evaluation. The attacks are grouped according to the norm that they optimize. Moreover, the boxes indicate whether the attack is a white-box or black-box attack. For each model, we report the baseline accuracy in percentage, the median adversarial distance median-$\delta_{\mathfrak{a}}$ (left value) and the threshold accuracy acc-$\mathfrak{a}$ (right value) in percentage for each attack, and the worst-case analysis of L^p-attacks by presenting the median-δ_p^* value (left value) and the acc-$\mathfrak{a}_p^*$ score (right value) in percentage. Higher scores mean better robustness. For each attack, the best median-$\delta_{\mathfrak{a}}$ is printed in bold.

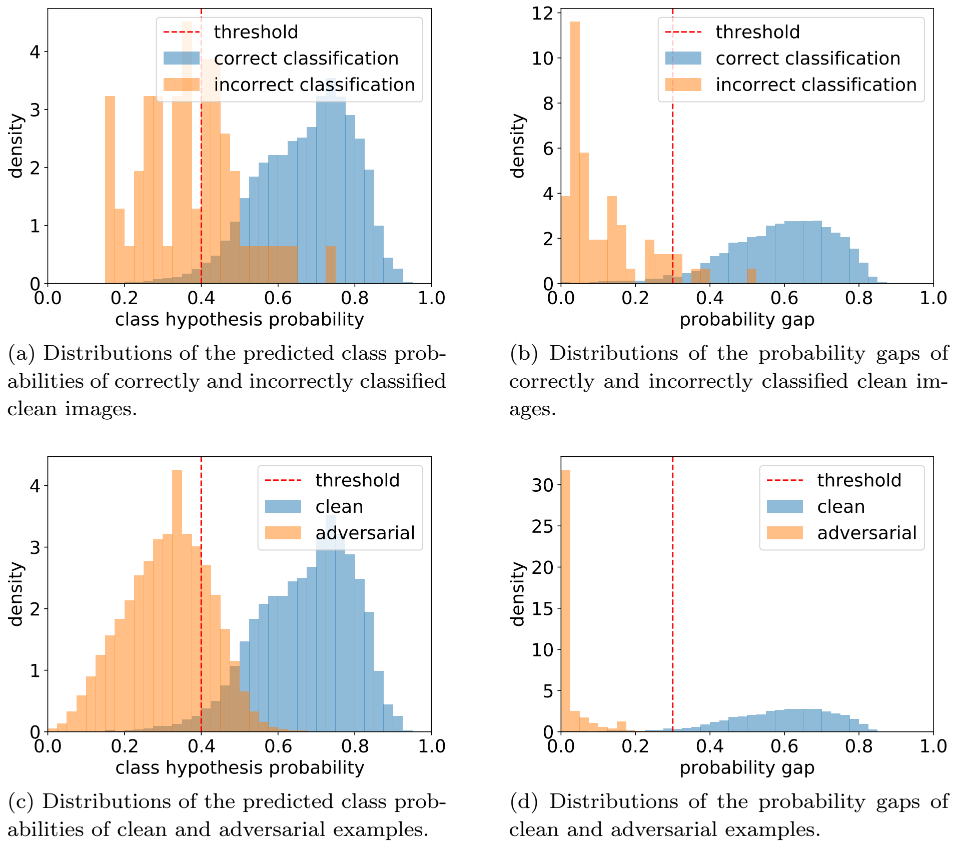
			CNN-0		CNN-4		CBC-0		CBC-4	
baseline accuracy				99.6		99.6		98.6		99.4
L^2	DeepFool	□	1.01	9.6	1.2	22	0.64	30.3	**1.98**	71
	C&W	□	0.86	5.2	0.99	5.1	0.58	28.5	**1.27**	29.6
	Pointwise	■	2.62	88.9	2.79	93.2	1.59	53.5	**3.19**	94.4
	Boundary	■	1.12	19.2	1.29	28.2	0.32	3	**1.69**	66
	worst-case		0.84	1.5	0.98	4.8	0.28	0.12	**1.26**	27.5
L^∞	FGSM	□	0.19	21	0.16	12.6	0.11	25.2	**0.24**	30.6
	DeepFool	□	0.1	0.1	0.11	0	0.08	24.4	**0.18**	15.2
	PGD	□	0.09	6	0.1	2.5	0.04	1.9	**0.15**	0.2
	worst-case		0.09	0	0.09	0	0.04	1.9	**0.15**	0.02
L^0	Pointwise	■	5	3.3	5	5.9	2	0.1	**7**	16
	S&P	■	29	78	32	82.2	11	46.6	**60**	94.2
	worst-case		5	3.3	5	5.9	2	0.01	**7**	16

the cross-entropy loss. The other parameters of the two CNN baseline models were chosen in such a way that the CNN-0 is consistent with the CBC-0 and the CNN-4 is consistent with the CBC-4.

Since the CBCs were not designed with a specific thread model in mind, the robustness was evaluated with the setup from Section 3.5 of the GTLVQ robustness evaluation. It should be noted that the CNN-0 from Section 3.5 corresponds to the CNN-0 evaluated here and, therefore, the numerical results are also the same.

Results of the robustness evaluation

The results of the robustness evaluation are presented in Table 4.1. As shown in bold, the CBC-4 with interpretable components far surpasses all other models in all nine attacks (including the worst-case). It should also be noted that the CBC-0 has by far the lowest robustness scores across all attacks. This empirically confirms the hypothesis that the interpretability of components is a good indicator for the



(a) Distributions of the predicted class probabilities of correctly and incorrectly classified clean images.

(b) Distributions of the probability gaps of correctly and incorrectly classified clean images.

(c) Distributions of the predicted class probabilities of clean and adversarial examples.

(d) Distributions of the probability gaps of clean and adversarial examples.

Figure 4.17: Discrete approximations of the continuous density functions of different distributions of correctly and incorrectly classified images and of clean and adversarial images. Additionally, we display the used rejection thresholds. To generate the plots, we used the adversarial examples from all nine attacks.

robustness of CBCs.

In general, the CNN baseline models have a higher robustness than the CBC-0. However, in contrast to the relationship between the CBC-0 and the CBC-4, we do not observe a significant improvement in robustness when we use the preferred feature extraction parameters in the CNN baselines. Consequently, this suggests that the preferred default setting is specific to the CBCs.

Compared to the robustness results of GLVQ and GTLVQ, the CBC-4 model scores significantly lower. Considering the fact that we have not proven that CBCs are

Table 4.2: TP and FP rates in percentage for the four different rejection strategies using the adversarial examples from all nine attacks. Threshold values of $t_{p^*} = 0.4$ and $t_\beta = 0.3$ were used.

dataset		count	t_{p^*}	t_β	t_{p^*} OR t_β	t_{p^*} AND t_β
clean (FP)	correct classified	9 938	2.0	3.8	4.1	1.7
	incorrect classified	62	58.1	91.9	91.9	58.1
adversarial (TP)		90 000	77.3	99.5	99.5	77.3

margin maximizers, this result is not surprising. Nevertheless, these first robustness results in combination with the interpretable components are an interesting outcome and could be useful for future work.

Rejection of adversarial examples

In Figure 4.17, a collection of statistics is shown that illustrate the differences between clean and adversarial examples of the CBC-4 model.[20] Figure 4.17a and Figure 4.17c visualize the distributions of the predicted probabilities. The probability gaps are illustrated in Figure 4.17b and Figure 4.17d.

The distribution of adversarial images is similar to the distribution of incorrectly classified images. Both have a significantly lower prediction probability *and* a smaller probability gap compared to correctly classified clean images. Consequently, using these two properties, it should be possible to construct a rejection strategy in which the *False Positives* (FPs)—rejected clean examples—are mostly misclassified anyway. Table 4.2 shows the *True Positives* (TPs) and FPs of four such strategies: The *prediction-reject strategy* rejects all input samples where the predicted class hypothesis probability (winner-takes-all rule applied to $\mathbf{p}(\mathbf{x})$) is below a probability threshold t_{p^*}. The *margin-reject strategy* rejects all inputs with a probability gap, see Equation (4.20), below t_β. These two rejection strategies are combined using both the logical AND and OR operations.

From the results in Table 4.2, we can conclude that it is indeed possible to use the class hypothesis probability and the probability gap of an input to efficiently reject adversaries. Using only the margin-reject strategy with $t_\beta = 0.3$, it is possible to reject almost all adversarial examples generated for the CBC-4 model (99.5 % of 90 k samples) at a low cost in terms of FP rate of correctly classified clean samples (3.8 %). Only the application of the prediction-reject strategy with $t_{p^*} = 0.4$ results in an even lower FP rate for correctly classified clean samples (2.0 %) but fails to reject a large portion of the adversarial examples (22.7 %). Due to the excellent

[20] A clean image, example, or sample is a non-manipulated image from the original dataset.

performance of the margin-reject strategy, the combinations of the two strategies do not yield significant benefits in terms of FP and TP rates.[21]

A clear benefit from combining both rejection strategies can be observed if we repeat the robustness evaluation of the CBC-4 model performed in the previous section in combination with the rejection strategies—that is, an adversarial example is only considered valid if it is misclassified and not rejected by the strategy. We realized the generation of such adversarial examples by updating the optimization goal of adversarial attacks to the goal of fooling the rejection strategy and the classification of the model. The FOOLBOX implementations of the C&W, DeepFool, and FGSM attacks were consistently unable to find an adversarial example when the OR-combined rejection strategy was used. Non-gradient based approaches, which are less restricted by the inclusion of the rejection strategy, required almost twice the adversarial distance compared to the previous robustness evaluation to fool both the model and the rejection strategy.

4.5.4 MNIST: Interpretation of the reasoning process

In this section, we show the interpretability of CBCs. Similar to interpretation techniques from NNs, we do this by considering input-dependent and input-independent visualizations. In its simplest form, we can interpret the learned reasoning process by visualizing the reasoning probabilities or the learned components or both, as we showed several times before, for instance, in Section 4.5.2. If the CBC uses patch components, it is also possible to visualize the learned model using heatmaps and reconstructions. In the following, we will focus on the creation of these visualizations and their relation to the probabilistic model and the corresponding probability tree diagram T. Furthermore, to stress the visualizations to make it clear that they really show how a model classifies, we

- train two CBCs with patch components similar to Figure 4.3 and interpret the learned classification process,
- generate an adversarial example for both models with the Boundary attack and explain the success of the attack and why the attack explicitly manipulates the specific region in the input sample to fool the CBC, and
- create artificial input images in which we remove all regions marked as irrelevant for the detection of a certain class and show that this removal does not influence the classification decision.

[21]For further research, we plan to study rejection strategies based on the measure probability gap *times* predicted class probability.

While this section focuses primarily on CBCs with patch components, some of the visualization techniques can also be applied to CBCs with non-patch components, even if the components are not defined in the input space. Additionally, we can construct further interpretation techniques by exploiting the similarity properties, as shown in Section 4.5.1.

CBC architectures

The first CBC—denoted as α-CBC—has trainable pixel probabilities $\alpha_{c,i,j}$ and the second—denoted as $\varnothing$-CBC—has non-trainable pixel probabilities, which are set to $\alpha_{c,i,j}$ equals $(v_r \cdot h_r)^{-1}$. In general, the models follow the Siamese architecture depicted in Figure 4.3 except that we apply a max pooling operation of pool size and stride 3×3 after the measuring of the detection probability to downsample the detection possibility stack to a spatial size of $v_d \times h_d$ equal to 7×7. Both models share the same two-layer CNN feature extractor architecture:

1. Convolution: 32 filters, kernel size 5×5, stride 1×1, bias, and no padding;
2. Convolution: 64 filters, kernel size 3×3, stride 1×1, bias, and no padding.

The eight patch components are defined in the input space with a size of $v_\kappa \times h_\kappa$ equal to 7×7.[22] Thus, the feature stacks have a spatial size of $v'_x \times h'_x$ equal to 22×22 and $v'_\kappa \times v'_\kappa$ equals 1×1 after the feature extraction—the depth of the stacks is 64, according to the number of convolution filters. We applied multiple reasoning, see Section 4.2.3, with two reasoning stacks per class and increased the number of training epochs to 300, accordingly. All other parameters (the training procedure and the initialization) were set to the default setting.

The $\varnothing$-CBC is restricted to reason over each pixel position because, by definition, each pixel position has the same importance. In contrast, the α-CBC can select which pixel positions are important to make a correct prediction. Therefore, this model can exclude pixels from the reasoning process.

Theoretical basis of the visualizations

The interpretability of the CBCs is based on visualizations of how the probability mass is distributed over the probability tree diagram T. The class hypothesis probability $p_c(\mathbf{x})$, see Equation (4.6), is the probability of *agreement under the condition of importance*—denoted by $A|I$—and is computed by

$$P(A \mid I, \mathbf{x}, c) = (\mathbf{d}(\mathbf{x}))^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^{+} + (\mathbf{1} - \mathbf{d}(\mathbf{x}))^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^{-}.$$

[22]The idea is to learn patches of four quarters of a circle plus two diagonal, horizontal, and vertical lines.

This event describes the correct matching of the extracted DP with the class DP. Moreover, we decompose an agreement into the positive and the negative agreement:

- A *positive agreement* A^+ is a path in the tree diagram T where a component is detected (event D) and requires reasoning by detection (event R).
- A *negative agreement* A^- is a path in the tree diagram T where a component is not detected (event $\overline{D}$) and requires reasoning by no detection (event $\overline{R}$).

These events are used to split the probability $P\left(A \mid I, \mathbf{x}, c\right)$ into two parts:

- The *positive agreement under the condition of importance* is the event that a component is detected that should be detected—denoted by $A^+|I$—and the probability is calculated by
$$P\left(A^+ \mid I, \mathbf{x}, c\right) = \left(\mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^+. \tag{4.22}$$
- The *negative agreement under the condition of importance* is the event that a component that should not be detected is not detected—denoted by $A^-|I$—and the probability is calculated by
$$P\left(A^- \mid I, \mathbf{x}, c\right) = \left(\mathbf{1} - \mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^-. \tag{4.23}$$

Both events can be related to paths in the trees T_c from the root to the leaves. In particular, $A^+|I$ is the upper solid line path and $A^-|I$ is the lower solid line path in Figure 4.2. In addition, the two probabilities represent the probability mass *in favor* of class c: $P\left(A^+ \mid I, \mathbf{x}, c\right)$ is the probability in favor of class c with respect to positive reasoning and $P\left(A^- \mid I, \mathbf{x}, c\right)$ is the probability in favor of class c with respect to negative reasoning.

Similarly, we can consider the complementary event of $A|I$, which is *disagreement under the condition of importance*—denoted by $\overline{A}|I$—and occurs when the extracted DP does not match the class DP. The probability of this event is computed by

$$\begin{aligned} P\left(\overline{A} \mid I, \mathbf{x}, c\right) &= 1 - P\left(A \mid I, \mathbf{x}, c\right), \\ &= \left(\mathbf{1} - \mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^+ + \left(\mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^-. \end{aligned}$$

Again, we decompose a disagreement into the positive and the negative disagreement:

- A *positive disagreement* $\overline{A}^+$ is a path in the tree diagram T where a component is not detected (event $\overline{D}$) and requires reasoning by detection (event R).
- A *negative disagreement* $\overline{A}^-$ is a path in the tree diagram T where a component is detected (event D) and requires reasoning by no detection (event $\overline{R}$).

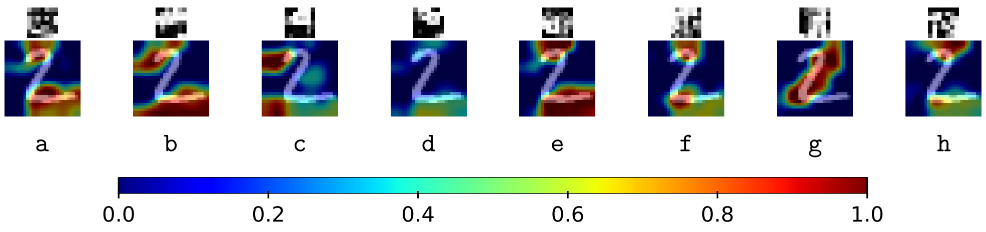


Figure 4.18: Detection probability heatmaps of the α-CBC on a test sample. At the top of each heatmap, we show the respective patch component. We use the colormap JET to map a probability of zero to blue and a probability of one to red. Each image is the weighted sum of the input sample (weight 0.5) and the heatmap (weight 0.5).

These events are used to split the probability $P\left(\overline{A} \mid I, \mathbf{x}, c\right)$ into two parts:

- The *positive disagreement under the condition of importance* is the event that a component is not detected that should be detected—denoted by $\overline{A}^{+}|I$—and the probability is calculated by

$$P\left(\overline{A}^{+} \mid I, \mathbf{x}, c\right) = \left(\mathbf{1} - \mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^{+}. \tag{4.24}$$

- The *negative disagreement under the condition of importance* is the event that a component is detected that should not be detected—denoted by $\overline{A}^{+}|I$—and the probability is calculated by

$$P\left(\overline{A}^{-} \mid I, \mathbf{x}, c\right) = \left(\mathbf{d}\left(\mathbf{x}\right)\right)^{\mathrm{T}} \cdot \bar{\mathbf{r}}_c^{-}. \tag{4.25}$$

The corresponding paths in the tree T_c in Figure 4.2 are the dashed line paths without non-importance. Additionally, the two probabilities represent the probability mass *against* class c: $P\left(\overline{A}^{+} \mid I, \mathbf{x}, c\right)$ is the probability against class c with respect to positive reasoning and $P\left(\overline{A}^{-} \mid I, \mathbf{x}, c\right)$ is the probability against class c with respect to negative reasoning.

Detection heatmap visualizations

We can use the detection probability function $d_k\left(\mathbf{x}\right)$ for a particular component $\boldsymbol{\kappa}_k$ to get an indication where a component had a detection in the input. In Figure 4.18, all eight detection probability heatmaps for the α-CBC are visualized using a test input. If we consider the component `b`, we see that it models almost a horizontal stroke. This is reflected in the heatmap, which shows that the horizontal lines of the two have a high detection for this component. Since the horizontal stroke is modeled in the upper part of the patch, it might appear in the heatmap that the detection of the stroke is in a lower region and has a slight offset. Another example is the component `g`. This component is almost a vertical line and correctly detects the vertical part of the two. In contrast, the component `d` models the lower left part of a circle and hence has no similarities with parts of the input.

The detection heatmaps are useful to gain an understanding of the similarity measure, especially if the representation of the components are learned in the input space and are interpretable. If this is not the case, it could be difficult to understand what a component really represents. In this case, a back-projection strategy of the components to the nearest (patch of a) training example, see Section 4.3, can be helpful.

Incorporation of pixel probabilities

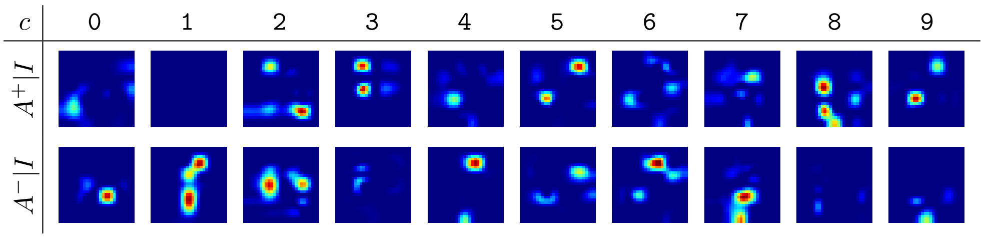
In spatial reasoning, the pixel probabilities have to be included in the visualizations. The pixel probabilities indicate how important a pixel position i, j is for the class c. They also determine how the individual probabilities $p_{c,i,j}(\mathbf{x})$ are combined to provide the class output $p_c(\mathbf{x})$. The pixel probabilities are included into the reasoning heatmaps and reconstructions by a final scaling step. The algorithm is as follows:

1. Collect all the pixel probabilities $\alpha_{c,i,j}$ of a class c into a map.
2. Normalize the map with the maximum probability (i. e., with $\max_{i,j} \alpha_{c,i,j}$).
3. Resize the map to the respective size of the target map (e. g., a particular heatmap).
4. Multiply the resized map with the target map.

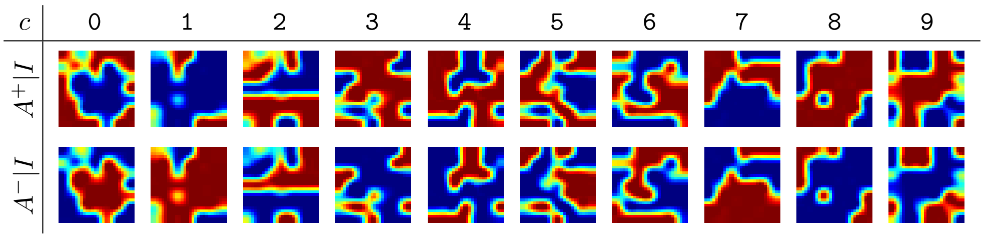
Reasoning heatmaps: Input-independent visualizations

To answer the question of *what a model has learned about a certain class*, we consider input-independent heatmaps. The idea is to stimulate the effective reasoning possibility vectors $\bar{\mathbf{r}}_c^{\pm}$ by the respective optimal detection possibility vector and to visualize the regions in favor of class c regarding the positive and negative reasoning. The optimal detection possibility vector for $A^+|I$ is $\mathbf{d}(\mathbf{x}) = \mathbf{1}$ and $\mathbf{d}(\mathbf{x}) = \mathbf{0}$ for $A^-|I$, see Equation (4.22) and Equation (4.23). With these optimal inputs, we compute a reasoning heatmap for a class c by the following procedure:

1. Resize the effective reasoning possibility stack of size $v_r \times h_r \times \#\mathcal{K}$ to the size $v_x \times h_x \times \#\mathcal{K}$.



(a) α-CBC heatmaps.



(b) $\varnothing$-CBC heatmaps.

Figure 4.19: Input-independent reasoning heatmaps of the α-CBC and the $\varnothing$-CBC for all classes. For each class, we present one reasoning stack—note that we used multiple reasoning and learned two reasoning stacks for each class. The class labels are shown at the top. We use the colormap JET to map a probability of zero to blue and a probability of one to red, see Figure 4.18 for the colorbar.

2. Calculate the pixel-wise dot products with the respective optimal detection possibility vector to obtain the heatmap of size $v_x \times h_x$.

3. Overlay the pixel probability map.

Via these heatmaps, we find the regions of a model where components have to be detected and have to not be detected. Figure 4.19a shows the learned sparse representations of the α-CBC. For example, the model learns to classify handwritten digits of the class 3 by recognizing specific line endings. In contrast, the class 1 is only recognized by negative reasoning—by checking that no component matches around the vertical stroke.

Figure 4.19b shows the reasoning heatmaps for the $\varnothing$-CBC (i. e., the model with non-trainable pixel probabilities). Since all probabilities $\alpha_{c,i,j}$ are identical, the final overlay of the pixel probability maps does not change the visualizations since each pixel probability has the same importance. In addition, the model is forced to reason correctly over each position to achieve a high output probability. The probability model requires that the sum of positive and negative effective reasoning has to be

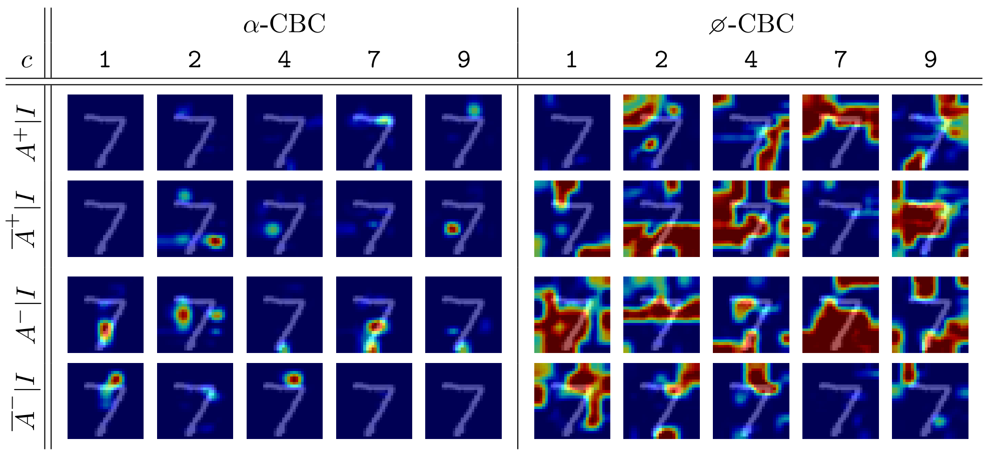


Figure 4.20: Input-dependent reasoning heatmaps of the α-CBC and the $\varnothing$-CBC for particular classes and for a test sample of the class 7 from the MNIST dataset. We use the colormap JET to map a probability of zero to blue and a probability of one to red, see Figure 4.18 for the colorbar. Each image is the weighted sum of the test sample (weight 0.34) and the heatmap (weight 0.66).

one. Therefore, $A^+|I$ and $A^-|I$ are complementary to each other. If a background component is learned, the reasoning heatmaps can be hard to interpret because they could highlight the background with positive reasoning. For instance, if we consider the learned concept of the class 0, we see how the model detects the outer shape of the zero with $A^+|I$ and the black middle with $A^-|I$. Another good example is the class 4, where the shape of a four is clearly visible in the heatmap of $A^+|I$. Furthermore, the model checks with $A^-|I$ that there is nothing detected at the top and bottom to avoid confusion with other digits, such as a nine.

Reasoning heatmaps: Input-dependent visualizations

The input-dependent heatmaps are created in the same way as the input-independent heatmaps except that we take the detection possibility vector from the respective position of the detection possibility stack $\mathbf{d}(\mathbf{x})$ as possibility vector. Additionally, we highlight the input image in the background. Now, we can visualize $A^{\pm}|I$ and $\overline{A}^{\pm}|I$ resulting in four possible visualizations for each class.

Figure 4.20 shows the input-dependent heatmaps for a given input and five different classes. Each heatmap represents the evidence for or against the class. In combination with the optimal heatmaps in Figure 4.19, these heatmaps can be used to see where the decision of a given class deviates from the optimum. For example, the α-CBC requires the detection of the end of the upper arc, the significant lower-left corner, and the end of the lower line to classify an input as two. The failure of the model to detect these features in the given image is illustrated by the $\overline{A}^{+}|I$ heatmap, which highlights the areas where these features should be detected to support the decision of a two. However, the $A^{-}|I$ heatmap shows that the negative reasoning requirements are fulfilled to be classified as a two. In contrast, the $A^{+}|I$ heatmap and the $A^{-}|I$ heatmap of the class 7 show that all requirements are fulfilled to be classified as a seven—the correct class. Almost no $\overline{A}|I$ is observed for this correct class. Note how the heatmaps correctly highlight the similarities between the correct class and every other class, such as the class 9.

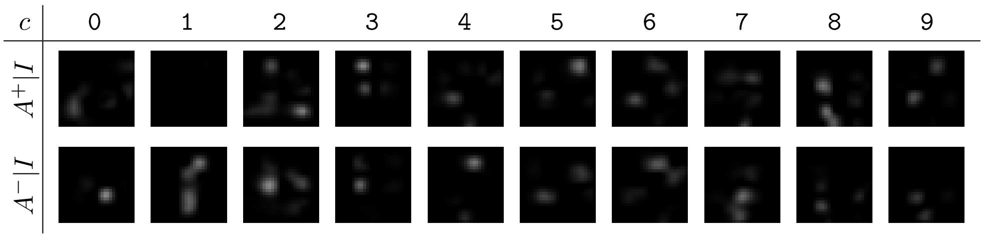
The behavior of the $\varnothing$-CBC is similar to that of the α-CBC except that it requires correct reasoning over the entire image instead of reasoning over a few regions. First, note how the combination of the heatmaps of $A^{\pm}|I$ and $\overline{A}^{\pm}|I$ for a class results in the respective input-independent heatmap of Figure 4.19. Looking at the class 4 heatmaps of the $\varnothing$-CBC, we see that the $A^{+}|I$ heatmap correctly emphasizes that the diagonal line of a seven could be the vertical line of a four. Therefore, the method reasons in favor of the class 4 over this part of the image. However, the $\overline{A}^{+}|I$ heatmap emphasizes that the left part of the four is not detected. Additionally, the important plausibility check that there is no upper stroke at a four fails because the $\overline{A}^{-}|I$ heatmap clearly highlights this area. Overall, the heatmaps of this class show a lot of disagreement (event $\overline{A}|I$) for the classification decision of a four. In contrast, there is almost no disagreement (event $\overline{A}|I$) for the correct class so that the input is correctly classified as a seven.

Reasoning reconstructions: Input-independent visualizations

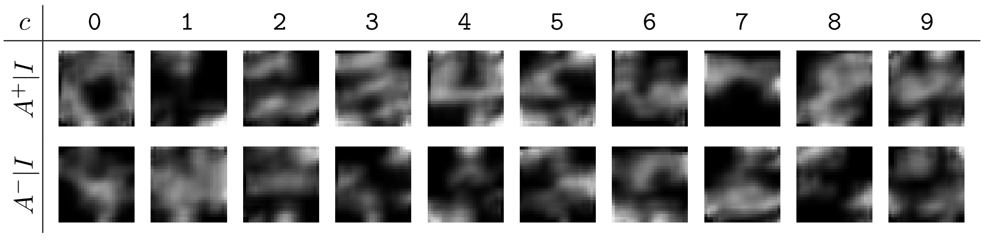
The reconstructions are similar to the heatmaps with the difference that we incorporate the learned patch components. A requirement for this visualization technique is that the components are defined in the input space. Additionally, we assume that the input space is defined over the interval $[0, 1]$. Again, we use the optimal detection possibility vectors and create the reconstructions by the following algorithm:

1. Initialize a matrix of zeros of size $(v_r + v_\kappa) \times (h_r + h_\kappa)$ as target image.[23]
2. For each pixel position i, j in $v_r \times h_r$ and component $\boldsymbol{\kappa}_k$ do:
    (a) Scale the component $\boldsymbol{\kappa}_k$ by the probability i, j, k from the effective reasoning possibility stack.

[23]It is possible to resize the reasoning stack first to get a higher resolution in the target image. In the results presented, we first resized to a size of $v'_x \times h'_x \times \#\mathcal{K}$, resulting in a target image of size $v_x \times h_x$.



(a) α-CBC reconstructions.



(b) $\varnothing$-CBC reconstructions.

Figure 4.21: Input-independent reasoning reconstructions of the α-CBC and the $\varnothing$-CBC for all classes. For each class, we present one reasoning stack—note that we used multiple reasoning and learned two reasoning stacks for each class. The class labels are shown at the top.

(b) Add the scaled component of size $v_\kappa \times h_\kappa$ to the target image at the corresponding receptive field. This is the area

$$\{i, i+1, \ldots, i+v_\kappa-1\} \times \{j, j+1, \ldots, j+h_\kappa-1\}.$$

3. Determine for each pixel the frequency with which a filter of size $v_\kappa \times h_\kappa$ covers that pixel during a convolution of an image with the target size, and use these values to normalize each intensity value in the target image.

4. Overlay the pixel probability map.

The visualizations generated by this principle contain the same information as the corresponding heatmaps. Consequently, it is just another way of visualizing the learned concepts. The advantage is that the reconstructions of the $\varnothing$-CBC are easier to interpret. Figure 4.21b, for instance, clearly shows digit shapes in $A^+|I$ for the most classes. Additionally, the learned plausibility checks by $A^-|I$ are easier to understand. For the class 3, the model checks that the ends of the three are not closed, which is an important difference to an eight. As opposed to the heatmaps,

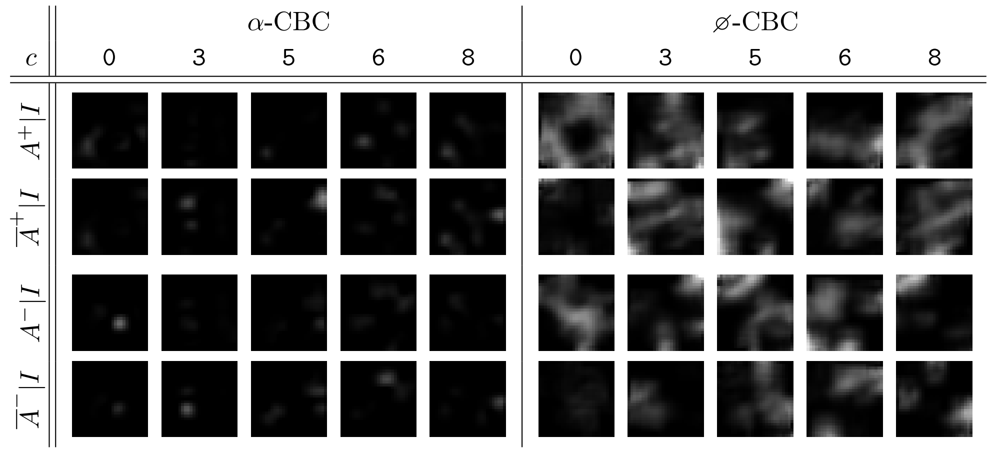


Figure 4.22: Input-dependent reasoning reconstructions of the α-CBC and the $\varnothing$-CBC for particular classes and for the third test sample from the MNIST dataset—the sample belongs to the class 0.

the summation of the reconstructions $A^+|I$ and $A^-|I$ does not result in a white image since we included the components that normally consist of black regions or shapes that are not perfectly white.

In contrast to the $\varnothing$-CBC reconstructions, the α-CBC reconstructions are almost black images with white blobs. This underlines the sparse representation that is learned for MNIST. For example, the classification of a six is based on the recognition of the intersection between the lower and upper part, the end of the top stroke, and that no stroke connects the end of the upper stroke with the lower circle (to distinguish it from an eight).

Reasoning reconstructions: Input-dependent visualizations

Similar to the heatmaps, we obtain these visualizations by exchanging the optimal possibility vector with the detection possibility vector from the respective position of the detection possibility stack $\mathbf{d}(\mathbf{x})$. Except for this change, we follow the algorithm for input-independent reconstructions. In Figure 4.22, we show the reconstructions of a test sample from the MNIST dataset. We present the reconstructions for those classes that were not displayed in Figure 4.20. Both models make a correct prediction—as we see in the $\overline{A}^{\pm}|I$ reconstructions—but the way they achieve this is very different and we leave it to the interested reader to interpret why the models do not classify the input as another digit.

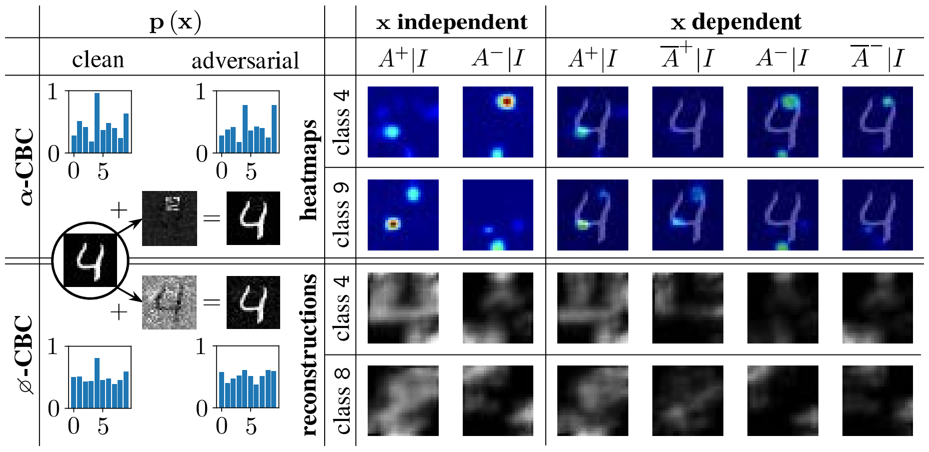


Figure 4.23: Visualization of the α-CBC heatmaps and the $\varnothing$-CBC reconstructions for an adversarial input. For simplicity, we illustrate the more meaningful visualization for each model. The model visualizations correspond to the best matching reasoning stack regarding the input. We use the colormap JET to map a probability of zero to blue and a probability of one to red, see Figure 4.18 for the colorbar. Each image is the weighted sum of the test sample (weight 0.34) and the corresponding heatmap (weight 0.66).

Explaining the success of an adversarial attack

Now, we use the different visualization techniques to explain the success of an adversarial attack. For each model, we generated an adversarial image with the Boundary attack. The left side of Figure 4.23 shows the clean input image (an image of the class 4), the adversarial perturbations (normalized by the minimum and maximum value to the interval $[0, 1]$), and the resulting adversarial images. Additionally, we show the distributions of $\mathbf{p}(\mathbf{x})$ for both the clean and adversarial images. The right side of the figure shows the different visualizations. First, we present the input-independent visualizations again to facilitate the comparison and to understand which concepts the models have learned. Second, we show the input-dependent visualizations to understand how the attack fools the model. For both models, we depict the correct and the adversarial class to see how the model deviates from the correct to the adversarial prediction.

We consider the input-independent visualizations ($\mathbf{x}$ independent) in Figure 4.23, to answer the question of *what the models have learned about the dataset*: For both models, the learned concepts of the clean and adversarial class are visualized by the optimal $A^+|I$ and $A^-|I$. As visible in the heatmaps, the α-CBC learned to recognize only a small number of parts necessary to distinguish the two classes. For the class `4`, this consists of checking that there is no stroke at the top and bottom, see $A^{-}|I$, while there is a corner on the left, see $A^{+}|I$. Such a radically sparse representation is learned for all classes. The reasoning for the class `9` is similar except that it requires $A^{+}|I$ instead of $A^{-}|I$ for the top stroke. In contrast, the $\varnothing$-CBC learned the whole concept of digits and not just a sparse representation since the reconstructions show real digit shapes in the $A^{+}|I$ visualizations. Moreover, the model performs interpretable plausibility checks via $A^{-}|I$, for instance, no top stroke at a four.

The right side of Figure 4.23 presents the input-dependent visualizations ($\mathbf{x}$ dependent) that we use to answer the question of *which parts of the input provide evidence for or against the current classification decision*: By considering the clean probability histogram $\mathbf{p}(\mathbf{x})$ of the α-CBC, we see that the clean input perfectly fits the learned concept of a four because it has a probability of one. The adversarial attack has turned the input into a four and a nine at the same time, see adversarial $\mathbf{p}(\mathbf{x})$. Remarkably, the attack found the high similarity between the two learned concepts and attacks the model by highlighting a few pixels in the upper bar region in the form of a patch—the manipulation only changes *one* pixel in $\mathbf{d}(\mathbf{x})$. Hence, the concept of a four is slightly violated as we see a highlighting of the top stroke region in the $\overline{A}^{-}|I$ heatmap. This causes the probability drop of the class `4`. At the same time, these few pixels provide $A^{+}|I$ for the top stroke of a nine and thus raise the probability. For the $\varnothing$-CBC, the attack behavior is completely different. Since the clean input already does not match the learned concept perfectly as $p_4(\mathbf{x}) \approx 0.8$, the attack fools the model by reducing the contrast via background noise. For example, via the $\overline{A}^{+}|I$ reconstruction, the model highlights that the clear detection of the upper part of a four is not given. It also recognizes that there could be a top or bottom stroke, see $\overline{A}^{-}|I$. A similar interpretation applies to the adversarial class.

The $\varnothing$-CBC with $\alpha_{c,i,j} = (v_d \cdot h_d)^{-1}$ is trained to learn a strong concept as it can only achieve $p_y(\mathbf{x}) \approx 1$ if it reasons perfectly at each pixel position. Therefore, the probability histogram shows a relatively high base probability for each class, as the overlap between encoded digits to a spatial size of $v_d \times h_d$ equal to 7×7 is often around 50 %. Furthermore, this restrictive classification principle violates the motivating example in Figure 4.1 because the model cannot apply indefinite reasoning over a pixel region. In contrast, the α-CBC can model the motivating example but is at the same time a clear example of what happens if we optimize without constraints as usually performed in NNs. Since the model is trained by minimizing an energy function, it learns to classify correctly with the lowest effort and, hence, oversimplifies. Therefore, the classification is performed in a non-intuitive way. Moreover, the interpretation shows that the classification of both CBCs is based on non-robust features of $\mathbf{f}$ as both are highly sensitive to background manipulations.

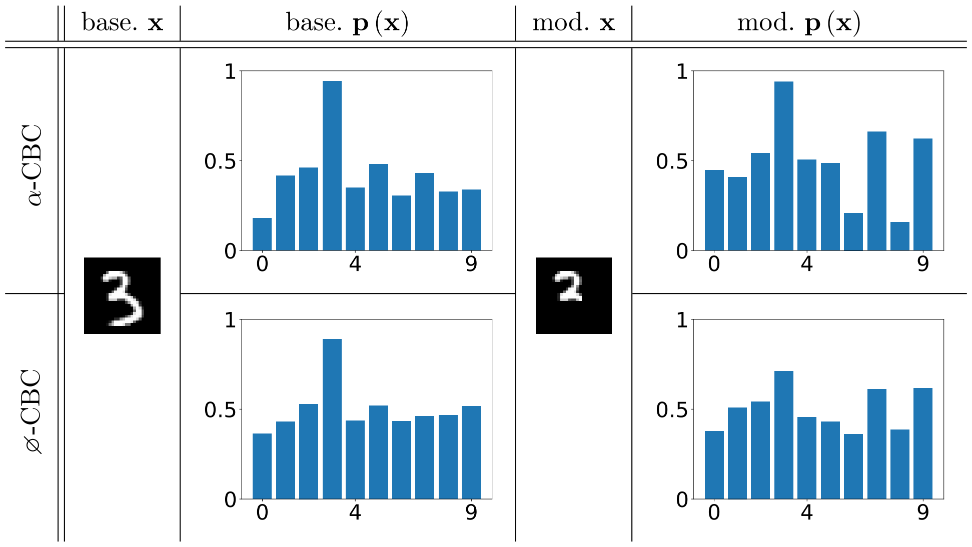


Figure 4.24: Results of the cut-off experiment where we modified (mod.) a baseline (base.) image regarding the import regions marked by the $A|I$ heatmaps.

Cut-off experiment

The idea behind the previous experiment is to make a stress test of the visualizations by explaining why an adversarial attack fooled the model. However, we can perform an even more aggressive test: Based on Figure 4.19 and Figure 4.21, we know exactly which regions of a given digit are the most important parts for a particular class. So if the visualizations are correct, it should be possible to remove all unimportant parts without affecting the classification decision.

In Figure 4.24, we visualize the results of such an experiment for both models. The input sample is an image from the test dataset. Without a special tuning, we removed the parts of a three that are not marked as important by the α-CBC, see Figure 4.19. As we see in the baseline probability distributions of $\mathbf{p}(\mathbf{x})$, both models classify the original input image correctly, but the α-CBC predicts a more confident classification because the margins to other classes are higher. If we remove parts of the input that are marked to not contribute to the class decision of the α-CBC for a three, we do not observe a real reduction of the output probability, although the probability for other classes changes. The image is still classified as a three, even though it looks more like a small two.

Unlike the α-CBC, the $\varnothing$-CBC realizes the manipulation because the probability decreases, but the image is still classified as a three. This is because, for the $\varnothing$-CBC, each pixel position contributes equally and, thus, every part of the three is important, as shown in Figure 4.21. In summary, this once again underlines the explanatory power of the visualization techniques.

Summary

In the previously presented experiments, we studied the interpretability of patch CBCs with spatial reasoning using two different models. The proposed visualization tools, consisting of heatmaps and reconstructions, provide a powerful tool to gain insights into the learned reasoning process to interpret the models. While the reconstructions are associated with specific requirements of the CBC architecture, the proposed heatmaps are applicable independently of the CBC architecture. To show that the proposed visualization techniques really represent the learned concepts, we presented two experiments in which we questioned the visualizations: First, we created an adversarial example for both models and showed that the visualization techniques can be used to explain the success of the adversarial attacks and the manipulation of the respective regions by the adversarial attacks. Second, we designed an experiment in which we removed the areas of an image that are marked as unimportant for the classification decision to show that the deletion does not affect the classification result.

With an accuracy of $(97.33 \pm 0.19)\,\%$, the performance of the models is significantly lower than the state-of-the-art accuracy. One reason for this might be that the models are quite sparse—both have only about 35 k trainable parameters. The accuracy can be increased if we use a deeper feature extractor. A problem that emerges, in this case, is that the receptive field size n_0 increases and, therefore, the minimum patch size. Of course, this is not a general problem, but it was obstructive in the experiments. When we increased the network depth and thus the patch size, the network began to rely mainly on positive reasoning. Furthermore, the reconstructions became less meaningful in some cases. However, the goal of this section was to show that the models classify by the three forms of reasoning and that the reconstruction principle works. Therefore, we decided to accept the models with low accuracy.

The reconstructions of inputs via back-projecting the components seem to be a visualization that should work in principle. However, so far, we have not been able to achieve acceptable results, neither for datasets of colored images nor for deep NNs with more than four layers. The reason for this is not clear yet and we hope to be able to improve these results in the future.

With the α-CBC and $\varnothing$-CBC, we proposed two different models that behave completely different in the classification process. The α-CBC can classify the data by reasoning over only a few pixel positions. We consider this model type as an example of how common NNs classify since we often do not restrict an NN regarding

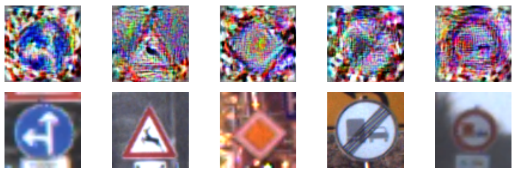


Figure 4.25: Five learned components on GTSRB. The top row shows the learned components and the bottom row presents a similar training sample $\mathbf{x}$ for each $\boldsymbol{\kappa}_k$.

how much of the image has to be understood. In contrast, the $\varnothing$-CBC is restricted to understand the entire image because each pixel position is equally important. At the moment, we are not sure what the superior method is—if there is one—or if a compromise between the two by regularizing the probabilities $\alpha_{c,i,j}$ is the right way. However, we suspect that the $\varnothing$-CBC might be the better method to detect outliers or manipulations in images since such things always led to a reduction of the probability $p_{c(\mathbf{x})}(\mathbf{x})$ *and* the margin to the runner-up class in the experiments, see also Figure 4.23.

4.5.5 GTSRB

The *German Traffic Sign Recognition Benchmark* (GTSRB) created by Stallkamp, Schlipsing, Salmen, and Igel (2012) is a dataset consisting of colored images of traffic signs. There are 43 different classes and the image sizes vary between 15×15 and 250×250. Additionally, the images are not necessarily squared and not all traffic signs are in the middle of the image. The official dataset contains 39 209 training and 12 630 test images. With this experiment, we show that CBCs scale to RGB images with background noise where each class can be represented by a prototype.

We used a CBC with 43 full-size components and a slightly modified four-layer CNN feature extractor—compared to the experiments on MNIST, see Section 4.5.1—with the following architecture:

1. Convolution: 32 filters, kernel size 7×7, stride 1×1, bias, and no padding;
2. Convolution: 64 filters, kernel size 3×3, stride 1×1, bias, and no padding;
3. Max pooling: pool size and stride 2×2;
4. Convolution: 64 filters, kernel size 3×3, stride 1×1, bias, and no padding;
5. Convolution: 128 filters, kernel size 3×3, stride 1×1, bias, and no padding.

The overall CBC setting was the default setting for CBCs. We trained the network with increasing margins in three steps, starting with β equal to 0.1 and continuing with 0.2 and 0.3. For each margin, we trained for 150 epochs. The images were resized to a spatial size of $v_x \times h_x$ equal to 64×64 and normalized to $[0,1]^{64\times 64\times 3}$ using the following procedure:

1. Normalize each image by the mean and standard deviation of the image.
2. Clip each value of the normalized image to the interval $[-2, 2]$.
3. Project the resulted image back to $[0,1]^{64\times 64\times 3}$.

If we trained with the target margin of 0.3 from the beginning, the model converged to a local minimum of low accuracy. In this case, the model has not learned a strong positive component for each class because some of the components resembled the same class. With only 43 components for 43 classes and the learned BMPP, it is therefore impossible to classify each class correctly.

The baseline CNN is constructed by replacing the detection probability and reasoning layer by an ordinary convolution and a fully connected layer such that the CBC and the baseline CNN are architecturally equivalent. To be more precise, the baseline CNN is obtained by applying the following two layers after the feature extraction:

1. Convolution: 43 filters, kernel size 22×22, stride 1×1, bias, no padding, and ReLU activation;
2. Fully connected: 43 units, bias, and softmax activation.

We trained the baseline CNN model with the cross-entropy loss for 450 epochs and with an initial learning rate of 10^{-4}. All other parameters were set to the non-default setting of CBCs.

Both networks achieved a comparable accuracy of $(97.2 \pm 0.77)\,\%$ for the CBC and $(97.5 \pm 0.18)\,\%$ for the baseline CNN. The CBC has a similar probability gap distribution as the CBC with 10 components trained on MNIST, see Figure 4.16a, and an average probability gap of 0.58 ± 0.17. In contrast, the baseline CNN has an average probability gap of 1 ± 0.03 and thus returns almost a one-hot coding. Similar to MNIST, the network discovers the BMPP and begins to form human-understandable, prototypical components, see Figure 4.25. Compared to MNIST, however, we observed stronger variations in the components between the different runs—for example, the background noise varied strongly. But in general, the components were always interpretable.

To underline the initial robustness evaluations performed on MNIST, see Section 4.5.3, we tested the robustness of the CBC and the baseline CNN on the physical

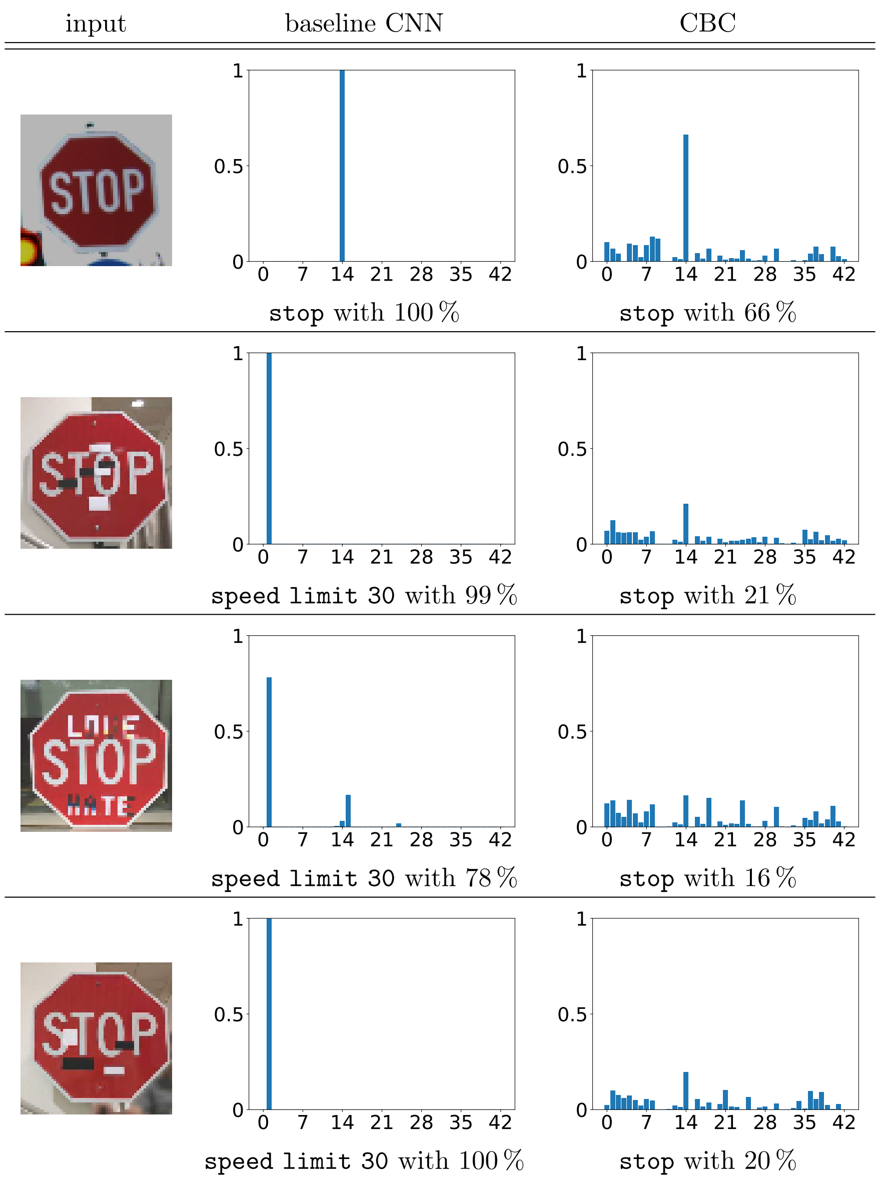


Figure 4.26: Physical stop-sign adversaries applied to the baseline CNN and the CBC. The inputs $\mathbf{x}$ and the corresponding output distributions $\mathbf{p}(\mathbf{x})$ are depicted. Additionally, the predicted label with the corresponding prediction probability is presented below each distribution. The first image is an example from the GTSRB test dataset and shows the probability distribution of both models on a clean image.

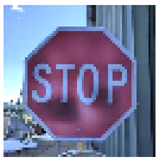
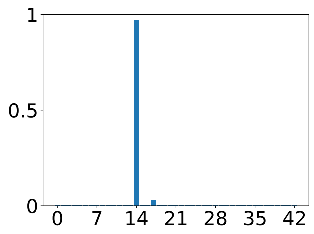
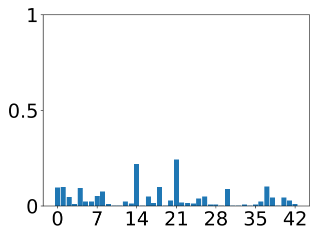
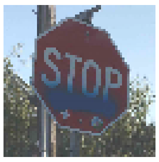
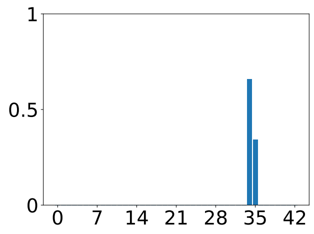
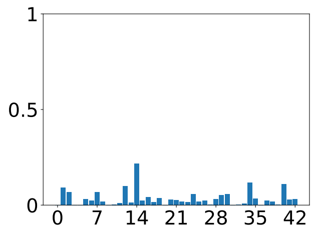
input	baseline CNN	CBC
	 `stop` with 97 %	 `double curve` with 24 %
	 `turn left ahead` with 66 %	 `stop` with 22 %

Figure 4.27: Physical stop-sign adversaries applied to the baseline CNN and the CBC—continuation of Figure 4.26.

stop-sign adversaries of Eykholt et al. (2018), see Figure 4.26 and Figure 4.27. With the exception of one example, the baseline CNN is consistently fooled by the adversarial examples and is always overconfident regarding its classification decision—such behavior of CNNs on these examples is known from the literature (e. g., Eykholt et al., 2018). Similar to the baseline CNN, the CBC does not classify all examples correctly. In contrast, however, it does not predict an overconfident output probability for an adversarial example. Moreover, compared to the probability distribution on the clean input, we see a clear difference to the probability distributions of the adversaries: All probabilities drop and the model is clearly uncertain about its decision. This result is consistent with the observed behavior on MNIST and can be a promising property for further research in outlier detection and rejection.

4.5.6 CIFAR-10

CIFAR-10[24] consists of 32×32 colored images from 10 different classes and 5 k training and 1 k test images for each class (Krizhevsky, 2009). The dataset is a

[24]Canadian Institute For Advanced Research 10 classes dataset.

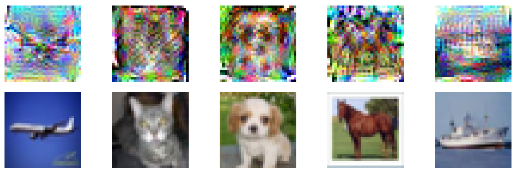


Figure 4.28: Visualizations of five learned components on CIFAR-10. The top row shows the learned components and the bottom row presents a similar training sample $\mathbf{x}$ for each $\boldsymbol{\kappa}_k$.

labeled subset of the 80 MILLION TINY IMAGES dataset, created by Torralba, Fergus, and Freeman (2008), with classes that are completely mutually exclusive. Compared to GTSRB, CIFAR-10 is harder to classify because the individual classes cannot be represented by a single prototype due to large intra-class variations and stronger background noise.

We used a CBC with the default setting, 10 full-size components, and the feature extractor architecture of Section 4.5.1. For a successful training on CIFAR-10, the 10 components were initialized with the mean image of each class. Additionally, we started the training with the mean squared error based on $\mathbf{p}(\mathbf{x})$ and a one-hot class label vector for 25 epochs—this helps the model to converge. However, when we trained only with the mean squared error, the network converged to a local minimum with a low accuracy of about 73 %. After the few initial epochs with the mean squared error, we switched to the training strategy used for GTSRB with the margin loss, increasing margins, and an initial learning rate of 0.001.

Similar to the GTSRB experiment in Section 4.5.5, the base CNN is the architecturally equivalent CNN of the CBC with the following layers after feature extraction:

1. Convolution: 10 filters, kernel size 5×5, stride 1×1, bias, no padding, and ReLU activation;
2. Fully connected: 10 units, bias, and softmax activation.

We trained the baseline CNN model with the cross-entropy loss for 475 epochs and with an initial learning rate of 0.001. All other parameters were set to the non-default setting of CBCs.

The CBC has an accuracy of (77.2 ± 0.07) %, which is close to the accuracy of (79.9 ± 0.3) % of the baseline CNN.[25] As expected, the CBC classifies the inputs

[25]If we allow the CBC to classify the data by weak decisions by lowering the margin β to 0.1, the accuracy increases to about 82 %.

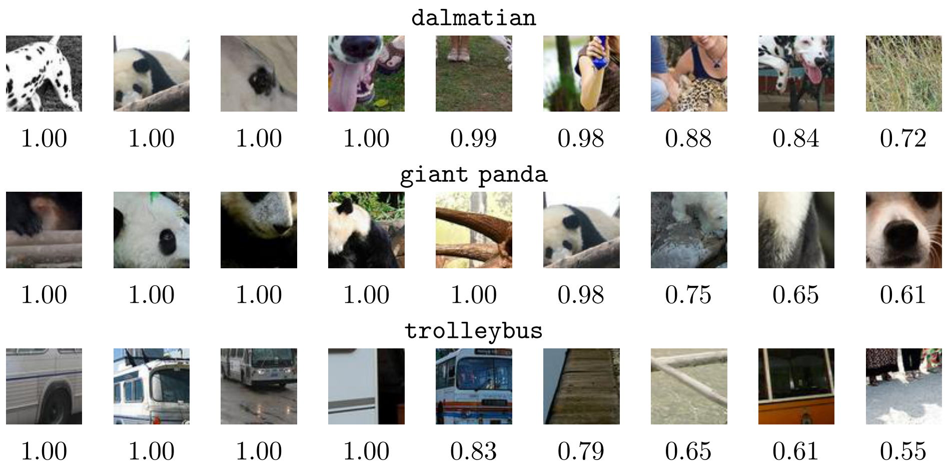


Figure 4.29: The nine components with the highest positive reasoning probability $r^{+}_{c,k}$ for three different classes of the ImageNet dataset. Below each component, we present the probability $r^{+}_{c,k}$ (rounded to two digits) for the respective class.

by the BMPP using prototypical components. In Figure 4.28, we visualize five such components. The components show different common characterizing properties of the classes, such as texture (e. g., fur of the cat), rough shapes (e. g., main shape of the horse), important arrangements of features (e. g., ears, eyes, and muzzle of the dog), the generally dominant colors at certain positions (e. g., white and blue of the ship), and so on. Across all runs, the learned components looked identical.

The probability distribution of the test dataset shows that the CBC has a much smaller average probability gap of 0.29 ± 0.17 than we observed for MNIST and GTSRB. This result is also reflected in the low accuracy compared to state-of-the-art results. The fact that we could not train on CIFAR-10 in the same way as on GTSRB is a good indicator that the margin loss function is not always a good choice and can be improved.

4.5.7 ImageNet

To evaluate CBC architectures on more complex data, we trained a CBC on the ILSVRC-2012 dataset from the ImageNet project—abbreviated as ImageNet dataset. The project provides images regarding the WordNet hierarchy for visual object recognition of Miller (1995). ImageNet is a dataset with 1 000 classes, about 1.3 M training images, and 50 k test images. Because the RGB images are natural images that come from different sources, the size and format vary greatly between images.

The CBC was implemented using a pretrained ResNet-50 as a *non-trainable* feature extractor.[26] In contrast to the CBCs discussed earlier, the patch components of dimension $m_\kappa = 2 \cdot 2 \cdot 2\,048$ are defined *directly* in the feature space.[27] This removes the relation between the components and the input space but drastically reduces the training time. After downsampling the detection possibility stack of spatial size $v_d \times h_d$ equal to 6×6 by global max pooling, the reasoning is applied—this reasoning strategy is the same as the downsampling strategy of Section 4.2.2.

The components were initialized by cropping the center of five images from each class and then processing them through the feature extractor, resulting in 5 k patch components. If the component $\boldsymbol{\kappa}_k$ was initialized by a sample of the class c, then we initialized $r^+_{c,k}$ as a uniform random value of $[z, 1]$ where z equals 0.75 and otherwise as a uniform random value of $[0, 1 - z]$. After that, the initialization of $r^-_{c,k}$ was determined by $r^+_{c,k} \cdot \left(1 - r^+_{c,k}\right)$. Consequently, we biased the model with positive reasoning to components that were sampled from the respective class.

The CBC was trained with the margin loss and $\beta = 0.1$. In compliance with earlier work on ImageNet, the input images were resized by first resizing the shortest side to 224 and then performing a center cropping of size 224×224. For the same reason, no image augmentation was used.

In Figure 4.29, the nine components with the highest positive reasoning probabilities for three exemplary classes are presented. After training the components in the feature space, the input representation of a component is determined by searching for the highest detection probability in the training dataset for the given component and cropping the corresponding image area in the input space. This method is similar to the approach of C. Chen et al. (2019).

In general, the components with a high positive reasoning probability (above the initialization bound of z) are found to be conceptually meaningful for the respective class. Further investigation of the components shows that the detection of the component with the second-highest positive reasoning probability for the class `dalmatian` in an image also provides evidence in favor of the class `giant panda`. Similarly, the component with the fifth-highest positive reasoning probability for the class `dalmatian` is also highly important for the classes `hyena`, `snow leopard`, and `English setter`, while the component with the fifth-highest positive reasoning probability for the class `trolleybus` is also important for the class `trolley car`. Similar shared components

[26]Pretrained ResNet-50 model from the Keras Applications library.
[27]The size $2 \times 2 \times 2\,048$ is the size of the output feature stack of the ResNet-50.

can be found across many classes, which shows that the CBC is capable of learning complex class-independent structures.

Averaged over all classes, a positive reasoning probability greater than z was learned for 5.2 ± 0.8 components per class, while $2\,781.8 \pm 23.3$ of 5 k components per class were assigned to a negative reasoning probability greater than z. As can be seen in Figure 4.29, the positive reasoning probabilities assigned to the components are close to 1.00 in most cases. This includes components that were not initialized with a bias towards the class in question. For instance, the component with the fifth-highest positive reasoning probability for the class `dalmatian` was initially biased towards the class `English setter`. The ratio between the number of positive and negative reasoning components suggests that the model relies heavily on negative reasoning to establish a baseline for its classification decision. We assume that in this higher-dimensional setting with a large number of components, positive reasoning is primarily utilized to fine-tune the classification decision of the model after a rough categorization by negative reasoning.

To evaluate the performance of CBCs, we compare both the accuracy and inference time to that of a CNN. The resulting CBC had an inference time of 371 ± 6 images per second, similar to 369 ± 2 images per second of a normal ResNet-50 with global average pooling and fully connected layer. This shows that the CBC generates no significant computational overhead. The top-five validation accuracy of the CBC is 82.4 %, which is similar to earlier CNN generations—for example, AlexNet with 82.8 % accuracy (Krizhevsky, Sutskever, & Hinton, 2012). Note that the CBC had a non-trainable feature extractor and no parameter tuning was performed. We are confident that the accuracy of CBC architectures on ImageNet can be improved with further studies.

4.6 Related work

Recently, Bouchacourt and Denoyer (2019) proposed an NN architecture that is similar to CBCs with a feature extractor. Their NN architecture learns a concept extractor, a concept classifier, and a final classifier through a special training procedure. The intermediate representation of the concept extractor is similar to the detection probability function and detects whether a concept is present or not. Unlike CBCs, however, the calculated representations are binary vectors of the form $\{0,1\}^{\#\mathcal{K}}$. Moreover, similar to the reasoning of CBCs, the final classifier is an affine function and classifies the data based on the extracted representations. The concept classifier is used to regularize the NN in such a way that the concepts are homogeneous and distinguishable from each other. Similar to CBCs, this classifier is used to provide interpretation techniques. Although the proposed approach is mathematically formalized as a probability framework, it is rather heuristic since the whole approach uses several softmax squashings. Furthermore, the method does not use indefinite nor negative reasoning, which is an important property of CBCs.

Reasoning in NNs

In its simplest form, it can be said that NNs already provide decisions based on reasoning. If one considers an NN as a multilayer perceptron network, the sign of each weight can be interpreted as either negative or positive reasoning about the corresponding feature. In this case, a weight of zero would model indefinite reasoning. However, the use of ReLU activations forces NNs to rely entirely on positive reasoning. Additionally, the weights and feature scores are unbounded so that they are difficult to interpret. Nevertheless, this interpretation of weights is used in interpretation techniques such as class activation mappings proposed by Zhou et al. (2016), making them similar to CBC heatmap visualizations.

Explicit modeling of reasoning

The use of components and the inclusion of negative and indefinite reasoning can be seen as an extension of the work of C. Chen et al. (2019). However, CBCs do not rely on the complicated three-step training procedure of their work and build on a probabilistic reasoning model. Tokmakov, Wang, and Hebert (2019) proposed a form of reasoning similar to the indefinite reasoning state by occluding parts of the learned representation. However, its components (attributes) are modeled in a textual form. In general, the reasoning process has slight similarities with the ideas mentioned by Akata, Perronnin, Harchaoui, and Schmid (2013) and the modeling of knowledge via graph structures (e. g., X. Chen, Li, Fei-Fei, & Gupta, 2018; Jiang, Xu, Liang, & Lin, 2018; Marino, Salakhutdinov, & Gupta, 2017).

Feature visualization techniques

If the components are defined as trainable parameters in the input space, the learned components are similar to feature visualization techniques of NNs (e. g., Erhan et al., 2009; Nguyen et al., 2019; Zeiler & Fergus, 2014). In contrast to these common visualization techniques, the components are the *direct visualizations of the penultimate layer weights* (detection probability layer), are *not* calculated by a post-processing step, and have a probabilistic interpretation. In addition, we are *not* applying regularizations to the components to resemble realistic images.

Prototype-based classification rules in NNs and similarity learning

An essential part of the proposed network is the use of a Siamese architecture to learn a similarity measure (e. g., Bromley et al., 1994; Chopra, Hadsell, & LeCun, 2005; Koch, 2015; Salakhutdinov & Hinton, 2007) and the idea to integrate a kind of prototype-based classification rule into NNs (e. g., Arik & Pfister, 2019; Li, Liu, Chen, & Rudin, 2018; Mensink, Verbeek, Perronnin, & Csurka, 2012; Papernot & McDaniel, 2018; Plötz & Roth, 2018; Snell, Swersky, & Zemel, 2017; Yang, Zhang, Yin, & Liu, 2018). Recently, the prototype classification principle is gaining a lot of attention in few-shot learning due to its ability to learn fast from few data (e. g., Gidaris & Komodakis, 2018; Mensink et al., 2012; Santoro, Bartunov, Botvinick, Wierstra, & Lillicrap, 2016; Snell et al., 2017; Vinyals, Blundell, Lillicrap, Kavukcuoglu, & Wierstra, 2016). The idea of replacing full-size prototypes with patches in similarity learning is also becoming more attractive—for instance, the object tracking method of Bertinetto, Valmadre, Henriques, Vedaldi, and Torr (2016).

4.7 Summary and discussion

In this chapter, we have presented a probabilistic classification model called classification-by-components network together with several possible realizations. Reduced to the essential change we made, these are the following.

From the perspective of prototype-based learning: The relaxation of the predefined class label assignment of prototypes. The detection probability function is related to a similarity measure so that the components perform a clustering. Together with the reasoning probabilities, the components have an implicit soft class label so that they can be combined to form the overall classification output.

From the perspective of NNs: The definition of a probabilistic framework for the final and penultimate layer of an NN. The detection probability layer is an extension of a convolution layer with the requirement to measure the detection of convolutional filters—called components—expressed in probabilities. Additionally, the final reasoning layer is still affine but follows a special implicit constraint defined by the probability model.

In summary, CBCs have the following important characteristics:

- The method classifies its input by applying positive, negative, and indefinite reasoning over an extracted DP. To the best of our knowledge, this is the first time that optionality of components (features) is explicitly modeled.

- The method uses a probabilistic reasoning process that directly outputs class hypothesis probabilities without requiring heuristic squashing methods such as softmax.
- The reasoning process is easily interpretable and simplifies the understanding of the classification decision.
- In combination with a trainable NN-based feature extractor, the method retains advantages of NNs such as being end-to-end trainable on large scale datasets and achieving high accuracies on complex tasks.

Parallel to the theoretical description and derivation of CBCs, we conducted several evaluations. In the first experiment, we have shown that CBCs can classify by the same principle as GLVQ and have a similar accuracy and interpretability on MNIST. However, by lowering the margin value β, CBCs can discover classification strategies that are highly efficient in terms of accuracy and are not realizable by ordinary prototype-based methods.

Most of the evaluations were performed with CBCs with a trainable NN-based feature extractor. We have shown that CBCs can help to understand the classification performed by NNs and can boost the performance of prototype-based approaches towards state-of-the-art results. All interpretation techniques are based on the probability framework. As shown in the experiments on MNIST, GTSRB, and CIFAR-10 with Siamese architectures, the method can generate human-understandable components and can converge to the BMPP without an explicit regularization. Additionally, we have shown by an experiment with patch components on MNIST that the models can answer questions about the classification decisions. More precisely, the developed interpretation techniques can answer the question of what causes the model to fail on an adversarial example. The conclusion that can be drawn here supports the recently published results of Ilyas et al. (2019): The feature extractor tends to extract features that are not robust in human visual perception.

A drawback of the Siamese architecture is the training overhead and the possible introduction of many parameters due to components in the input space. If the architecture is not Siamese, CBCs have almost no disadvantages compared to NNs. However, to be able to use all the presented interpretation techniques, the back-projection strategy presented by C. Chen et al. (2019) must first be applied—as we have shown on ImageNet. The evaluation on ImageNet also showed that CBCs are capable of learning high-dimensional components that can be utilized by multiple classes. Investigation of these shared components can provide additional insight into the classification approach of the model. The heatmap visualizations are always applicable and extend the well-known class activation mapping method of Zhou et al. (2016) by the possibility to visualize disagreement.

The CBC architecture is a promising new method for classification and motivates further research. For example, on the following topics.

Initialization strategies and loss functions: Although the proposed margin loss function worked well in some experiments, it was difficult to train CBCs on large datasets of colored images. Why this is the case is not yet clear. We observed several times that the method was unable to overcome the margin bound and converged to an averaged loss value of β, which could be an indication that the margin loss is too restrictive and can be improved. We are also certain that an appropriate initialization technique is important. For instance, due to the Euclidean constraint of the feature extractor, the recently proposed initialization scheme of Arpit, Campos, and Bengio (2019) could be used for future work.

Robustness properties and outlier detection: An initial robustness evaluation and the use of class hypothesis possibility vectors for outlier detection show promising results. However, this evaluation has to be extended to be able to say with certainty whether the CBCs have superior outlier detection capabilities. The robustness of CBCs seems higher than for ordinary NNs. Nevertheless, the scores are lower than the robustness values of GLVQ methods. Consequently, future work should also focus on the study of robustification methods for the feature extractor.

Predefined components: A remarkable property of CBCs is that we can inject knowledge into the classifier through the components. We have done this in the CIFAR-10 and ImageNet experiments. This concept allows us to force a feature extractor to extract specific types of features. For example, the components could be defined as manipulated images to make the feature extractor robust, similar to the study of Geirhos et al. (2019). We have performed such experiments on MNIST, where we initialized the components as inverted MNIST digits (swapping black and white). During training, the components were kept non-trainable, so we forced the feature extractor to rely on edge detection. Moreover, predefined components can be used to extend a CBC after it was trained. For instance, suppose we already trained a CBC with a given set of components. Now, we add some more components to the set that are useful for an existing or new class. By setting the indefinite reasoning probabilities of the new components to one for the classes that we do not want to be affected by this update, we preserve the previous performance. After that, we can start a retraining of the new reasoning probabilities for the desired classes. In general, this is possible because the class hypothesis probabilities are computed independently to each other and the model can set the reasoning over components into the indefinite state. Finally, this property should be beneficial for few-shot and zero-shot learning tasks.

Detection probability functions: Lemma 4.1 states an important result: The discrimination ability of CBCs is bounded by the detection probability function. Thus, it makes sense to search for better detection probability functions. Even though the cosine similarity worked well in the experiments, we doubt that it is generally a good choice. For example, consider Figure 4.26 and note that an almost perfect input is far from a 100 % class hypothesis probability. This indicates that the cosine similarity is not capable of adequately handling the possible variations within a class. A possible alternative function is the structural similarity of Z. Wang, Bovik, Sheikh, and Simoncelli (2004), which we want to study in future work.

Regularizations: As was shown in the patch experiments with the α-CBC and $\varnothing$-CBC, it is worthwhile to investigate possible regularization strategies of trainable parameters. In the case of the two presented patch models, a reasonable compromise between fixed and trainable pixel probabilities seems desirable.

Mathematical analysis: Several aspects of CBCs are closely related to prototype-based classifiers, which are well-understood in their mathematical properties. Therefore, in future work, we want to examine whether we can derive results similar to those of the margin or convergence analysis of GLVQ.

Interpretation techniques: In this chapter, we have developed a number of interpretation techniques that worked well across several experiments and datasets. However, the reconstruction visualizations only worked well in the MNIST patch experiment. We could not successfully apply these visualizations to colored image datasets so that they became meaningful. Additionally, we observed that the components are more difficult to interpret when they are defined as patch components or applied to colored image datasets. The reasons for these problems are still unclear and also require further research.

Chapter 5
Summary and Concluding Remarks

Since the prototype-based classification principle was proposed, the definition of prototypes remained almost unchanged. In this thesis, we made a step forward and proposed two approaches to extend the prototype principle for classification algorithms. The first approach extends simple prototype vectors to set-prototypes—*generalized tangent learning vector quantization*, Chapter 3—and the second relaxes the class-specific prototype principle to components—*classification-by-components networks*, Chapter 4. We introduced these extensions to improve the classification capabilities of prototype-based classification algorithms while preserving their interpretability and robustness properties, and to improve the interpretability of NNs by transferring the interpretability properties of prototype-based classification algorithms. These methods were evaluated by numerical experiments on real-world and toy datasets. Additionally, we have theoretically analyzed the properties of these methods. Exemplarily, we applied both concepts in LVQ-based classification algorithms and showed that the application of both concepts leads to higher classification accuracies. For a detailed summary, we refer to the *summary and discussion* sections of the two main chapters—see Section 3.7 and Section 4.7.

In recent years there was and still is a growing interest in using ML everywhere and for everything. To put it somewhat exaggerated: There is a trend to replace well-suited methods based on decades of knowledge (e.g., control theory) with an NN architecture from last year. This trend is concerning because it is still not entirely clear how NNs achieve their classification decision. Of course, NNs are state of the art in terms of flexibility to model problems or in terms of accuracy. For example, the recent achievements in object detection in images by NNs are outstanding. However, their lack of interpretability hampers their unrestricted use in safety-critical applications such as autonomous driving, and until this is resolved, there are doubts whether a company will soon be able to offer a fully autonomous car. The motivation of this work was to develop methods that can bridge this described gap.

With the derived GTLVQ algorithm, we presented an alternative to NNs and we prefer this algorithm over NNs whenever possible. In all experiments, GTLVQ achieved good accuracies and we have shown that GTLVQ is provably robust against adversarial attacks. By calculating the hypothesis margin value regarding a given sample, we get knowledge about the confidence of the GTLVQ classifier and, at the same time, get a lower bound for the adversarial distance. Additionally, we can interpret the classifier via sampling points from the set-prototypes. These properties are desirable for safety-critical applications and, therefore, a good reason to deploy GTLVQ instead of NNs. However, it is not yet clear how to scale the GTLVQ algorithm for challenging datasets (e. g., ImageNet) such that the method achieves reasonable accuracy. Compared to NNs, GTLVQ becomes parameter greedy when trained on large images, which seems like a natural limitation for its application. Consequently, in future work, we plan to investigate GTLVQ on large image datasets after feature extraction to keep the input dimension feasible.

By drawing inspiration from prototype-based classifiers and cognitive psychology, we derived the CBC architecture. When applied to NNs, this classification principle provides a probabilistic framework to interpret how a classification decision is performed. The presented results are promising and we believe that CBCs could spur a new line of research. But until CBCs can compete with modern NN architectures, there are a lot of questions to be answered and improvements to be made. Up to now, it is much more difficult to train a CBC with ResNet-50 feature extractor on ImageNet than an ordinary ResNet-50 with a fully connected classification layer. Moreover, the accuracy of such CBCs is lower than the accuracy of modern NNs. However, the current state-of-the-art initialization methods, regularization schemes, activation functions, and so on were designed for NNs so that a direct comparison to NNs is difficult. Consequently, future work should focus on the investigation of suitable initialization methods and appropriate loss and detection probability functions. Furthermore, due to the strong relationship between LVQ methods and CBCs, it should be analyzed whether the margin maximization theory can be extended to CBCs. The first experiments show good robustness scores against adversarial attacks, but theoretical guarantees as obtained for GTLVQ are not proven. Additionally, applications of CBCs to non-image data should also be the subject of future studies.

Publications

[2016a] Villmann, T., Kaden, M., Bohnsack, A., Villmann, J.-M., Drogies, T., Saralajew, S., & Hammer, B. (2016). Self-adjusting reject options in prototype based classification. In E. Merényi, M. J. Mendenhall, & P. O'Driscoll (Eds.), *Advances in Self-Organizing Maps and Learning Vector Quantization: Proceedings of the 11th International Workshop WSOM 2016* (Vol. 428 of the Advances in Intelligent Systems and Computing, pp. 269–279). doi:`10.1007/978-3-319-28518-4_24`

[2016b] Saralajew, S., & Villmann, T. (2016). Adaptive tangent distances in generalized learning vector quantization for transformation and distortion invariant classification learning. In *Proceedings of the 2016 International Joint Conference on Neural Networks – IJCNN 2016* (pp. 2672–2679). doi:`10.1109/IJCNN.2016.7727534`

[2016c] Saralajew, S., Nebel, D., & Villmann, T. (2016). Adaptive Hausdorff distances and tangent distance adaptation for transformation invariant classification learning. In A. Hirose, S. Ozawa, K. Doya, K. Ikeda, M. Lee, & D. Liu (Eds.), *Proceedings of the 23rd International Conference on Neural Information Processing – ICONIP 2016* (Vol. 9949 of the Lecture Notes in Computer Science, pp. 362–371). doi:`10.1007/978-3-319-46675-0_40`

[2017a] Saralajew, S., & Villmann, T. (2017a). Restricted tangent distances for local data dissimilarities. In T. Villmann & F.-M. Schleif (Eds.), *Machine Learning Reports 01/2017*. Retrieved from `https://www.techfak.uni-bielefeld.de/%7Efschleif/mlr/mlr_01_2017`

[2017b] Saralajew, S., & Villmann, T. (2017b). Restricted tangent distances in learning vector quantization. In T. Villmann & F.-M. Schleif (Eds.), *Proceedings of the Mittweida Workshop on Computational Intelligence – MIWOCI 2017* (Vol. 02/2017 of the Machine Learning Reports, pp. 20–21). Mittweida, Germany. Retrieved from `https://www.techfak.uni-bielefeld.de/%7Efschleif/mlr/mlr_02_2017`

[2017c] Saralajew, S., & Villmann, T. (2017c). Transfer learning in classification based on manifold models and its relation to tangent metric learning. In *Proceedings of the 2017 International Joint Conference on Neural Networks – IJCNN 2017* (pp. 1756–1765). doi:10.1109/IJCNN.2017.7966063

[2017d] Villmann, T., Biehl, M., Villmann, A., & Saralajew, S. (2017). Fusion of deep learning architectures, multilayer feedforward networks and learning vector quantizers for deep classification learning. In J.-C. Lamirel, M. Cottrell, & M. Olteanu (Eds.), *Proceedings of the 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization – WSOM+ 2017* (pp. 248–255). doi:10.1109/WSOM.2017.8020009

[2018a] Villmann, A., Kaden, M., Saralajew, S., Hermann, W., & Villmann, T. (2018). Reliable patient classification in case of uncertain class labels using a cross-entropy approach. In M. Verleysen (Ed.), *Proceedings of the 26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning – ESANN 2018* (pp. 153–158). Bruges, Belgium: i6doc.com. Retrieved from https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2018-97

[2018b] Villmann, A., Kaden, M., Saralajew, S., & Villmann, T. (2018). Probabilistic learning vector quantization with cross-entropy for probabilistic class assignments in classification learning. In L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, & J. Zurada (Eds.), *Proceedings of the 17th International Conference on Artificial Intelligence and Soft Computing – ICAISC 2018* (Vol. 10841 of the Lecture Notes in Computer Science, pp. 724–735). doi:10.1007/978-3-319-91253-0_67

[2018c] Villmann, T., Ravichandran, J., Saralajew, S., & Biehl, M. (2018). Dropout in learning vector quantization networks for regularized learning and classification confidence estimation. In T. Villmann & F.-M. Schleif (Eds.), *Proceedings of the Mittweida Workshop on Computational Intelligence – MIWOCI 2018* (Vol. 01/2018 of the Machine Learning Reports, pp. 15–21). Mittweida, Germany. Retrieved from https://www.techfak.uni-bielefeld.de/%7Efschleif/mlr/mlr_01_2018

[2018d] Saralajew, S., Nooka, S., Kaden, M., & Villmann, T. (2018). Learning vector quantization capsules. In T. Villmann & F.-M. Schleif (Eds.), *Machine Learning Reports 02/2018*. Retrieved from https://www.techfak.uni-bielefeld.de/%7Efschleif/mlr/mlr_02_2018

[2018e] Saralajew, S., Holdijk, L., Rees, M., & Villmann, T. (2018). *Prototype-based neural network layers: Incorporating vector quantization.* arXiv: `1812.01214 [cs.LG]`

[2019a] Ravichandran, J., Saralajew, S., & Villmann, T. (2019). DropConnect for evaluation of classification stability in learning vector quantization. In M. Verleysen (Ed.), *Proceedings of the 27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning – ESANN 2019* (pp. 19–24). Bruges, Belgium: i6doc.com. Retrieved from `https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2019-131`

[2019b] Saralajew, S., Holdijk, L., Rees, M., & Villmann, T. (2019). Robustness of generalized learning vector quantization models against adversarial attacks. In A. Vellido, K. Gibert, C. Angulo, & J. D. Martín-Guerrero (Eds.), *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization: Proceedings of the 13th International Workshop, WSOM+ 2019* (Vol. 976 of the Advances in Intelligent Systems and Computing, pp. 189–199). doi:`10.1007/978-3-030-19642-4_19`

[2019c] Saralajew, S., Holdijk, L., Rees, M., Asan, E., & Villmann, T. (2019). Classification-by-components: Probabilistic modeling of reasoning over a set of components. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2019* (pp. 2792–2803). Vancouver, BC, Canada: Curran Associates, Inc. Retrieved from `https://papers.nips.cc/paper/8546-classification-by-components-probabilistic-modeling-of-reasoning-over-a-set-of-components`

[2020a] Ravichandran, J., Kaden, M., Saralajew, S., & Villmann, T. (2020). Variants of DropConnect in learning vector quantization networks for evaluation of classification stability. *Neurocomputing*, *403*, 121–132. doi:`https://doi.org/10.1016/j.neucom.2019.12.131`.

Mathematical Symbols

Greek letters

$\alpha_{c,i}$, $\alpha_{c,i,j}$	The class-wise pixel probabilities.
$\tilde{\alpha}_{c,i}$, $\tilde{\alpha}_{c,i,j}$	The encoded class-wise pixel probabilities as elements of $\mathbb{R}$.
β	The margin parameter of the margin loss.
$\delta_{\mathfrak{a}}\left(\mathbf{x}, c\left(\mathbf{x}\right)\right)$	The adversarial distance of the attack $\mathfrak{a}$ with respect to an input sample $\left(\mathbf{x}, c\left(\mathbf{x}\right)\right)$.
$\delta_p^*\left(\mathbf{x}, c\left(\mathbf{x}\right)\right)$	The worst-case adversarial distance for an input $\left(\mathbf{x}, c\left(\mathbf{x}\right)\right)$ with respect to L^p-attacks.
$\boldsymbol{\epsilon}$	The variable for an adversarial perturbation.
η	The learning rate parameter.
$\boldsymbol{\theta}$	The parameter vector of a linear subspace, an affine subspace, or an orthotope.
$\boldsymbol{\vartheta}$	The vector of trainable parameters of a feature extractor or a classifier function.
$\boldsymbol{\kappa}$	The variable for a component.
$\mu\left(\mathbf{x}\right)$	The relative distance difference for an input $\mathbf{x}$.
$\nu\left(\mathbf{x}\right)$	The signed probability gap for an input $\mathbf{x}$.
$\xi^{\pm}$	The scaling factors of the gradients of the GLVQ or CBC loss.
σ	A positive slope or variance parameter.
$\phi\left(\cdot\right)$	A monotonically increasing, almost everywhere differentiable squashing function.

$\varphi(\cdot)$	A function to transform the cosine similarity to a detection probability function.

Latin uppercase letters

A	The binary random variable for agreement.
A^+, A^-	The binary random variable for positive and negative agreement.
$\overline{A}^+$, $\overline{A}^-$	The binary random variable for positive and negative disagreement.
$\mathbf{B}$	A basis matrix of a linear subspace.
$\mathcal{C}$	The set of all class labels.
$\#\mathcal{C}$	The number of classes.
D	The binary random variable for detection.
$E(\cdot,\cdot)$	The empirical risk or the averaged loss function.
$\mathrm{ELU}(\cdot)$	The exponential linear unit function.
$\mathcal{F}(\mathcal{S})$	The set of all non-empty compact subsets of $\mathcal{S}$.
$H(\cdot)$	The Heaviside step function.
I	The binary random variable for importance.
$\mathbf{I}_n$	The n-dimensional identity matrix.
$\mathcal{K}$	The set of all components.
$\#\mathcal{K}$	The number of components.
$\mathcal{P}(\mathcal{S})$	The power set of $\mathcal{S}$ without the empty set.
$\mathbf{P}$	The orthogonal projector onto the complement of the linear subspace spanned by $\mathbf{B}$.
$\mathbf{Q}$	The transformation matrix in d_Q.
R	The binary random variable for reasoning by detection.
$\mathcal{R}$	A centered and axis-aligned hyperrectangle.
$\mathrm{ReLU}(\cdot)$	The rectified linear unit function.

$\mathcal{S}$	A non-empty set.
$\mathrm{Swish}\left(\cdot\right)$	The Swish function.
T	The variable for a probability tree diagram.
T_c	The variable for a probability sub-tree diagram regarding class c.
$\mathcal{T}$	A dataset of labeld data points.
$\#\mathcal{T}$	The number of labeled data points in $\mathcal{T}$.
$\mathcal{W}$	The set of all prototypes.
$\#\mathcal{W}$	The number of prototypes.

Latin lowercase letters

$\mathfrak{a}$	The variable for an adversarial attack.
$\mathbf{a}$	The parameter vector to define a centered and axis-aligned hyper-rectangle.
$\mathbf{a}_c$, $\mathbf{b}_c$	The variables for the coding vectors of the reasoning possibility vectors or stacks of class c during training.
$\text{acc-}\mathfrak{a}\left(\mathcal{T}\right)$	The adversarial threshold accuracy of the attack $\mathfrak{a}$ on the dataset $\mathcal{T}$.
$\text{acc-}\mathfrak{a}_p^*\left(\mathcal{T}\right)$	The worst-case adversarial threshold accuracy of L^p-attacks on the dataset $\mathcal{T}$.
c	A class label of $\mathcal{C}$ or the indicator variable of a class.
$c\left(\mathbf{x}\right)$	The class label of the input $\mathbf{x}$.
$c^*\left(\mathbf{x}\right)$	The predicted class label of $\mathbf{x}$ after applying the winner-takes-all rule.
$d\left(\cdot,\cdot\right)$	A distance, metric, dissimilarity, or detection probability function.
$\mathsf{d}\left(\cdot,\cdot\right)$	A point-set dissimilarity.
$d_E\left(\cdot,\cdot\right)$	The Euclidean distance.
$d_H\left(\cdot,\cdot\right)$	The Hausdorff distance.

$d_Q(\cdot,\cdot)$	The quadratic-dissimilarity.
$d_k(\mathbf{x})$, $\mathsf{d}_k(\mathbf{x})$	The dissimilarity or detection probability function for the k-th prototype or component regarding $\mathbf{x}$.
$d^+(\mathbf{x})$, $\mathsf{d}^+(\mathbf{x})$	The dissimilarity to the closest prototype of the correct class $c(\mathbf{x})$ given an input $(\mathbf{x}, c(\mathbf{x}))$.
$d^-(\mathbf{x})$, $\mathsf{d}^-(\mathbf{x})$	The dissimilarity to the closest prototype of a class other than $c(\mathbf{x})$ given an input $(\mathbf{x}, c(\mathbf{x}))$.
$\mathbf{d}(\mathbf{x})$, $\boldsymbol{\mathsf{d}}(\mathbf{x})$	The prototype response vector, detection possibility vector, or detection possibility stack for an input $\mathbf{x}$.
$\triangle d(\mathbf{x})$	The absolute distance difference regarding $\mathbf{x}$.
$\mathbf{f}(\mathbf{x})$, $\mathbf{f}(\mathbf{x};\boldsymbol{\vartheta})$	A classifier function or a feature extractor that takes an input $\mathbf{x}$ and is parameterized by a vector $\boldsymbol{\vartheta}$.
h_κ	The horizontal spatial dimension of a component $\boldsymbol{\kappa}$.
h_d	The horizontal spatial dimension of a detection possibility stack.
h_p	The horizontal spatial dimension of a class hypothesis possibility stack.
h_r	The horizontal spatial dimension of a reasoning stack.
h_x	The horizontal spatial dimension of an input $\mathbf{x}$.
h'_κ	The horizontal spatial dimension of a component $\boldsymbol{\kappa}$ after feature extraction.
h'_x	The horizontal spatial dimension of an input $\mathbf{x}$ after feature extraction.
k	The indicator variable to the k-th prototype or component.
$l(\cdot,\cdot)$	A loss function.
m_κ	The dimension of a component $\boldsymbol{\kappa}$ after feature extraction.
m_x	The dimension of an input $\mathbf{x}$ after feature extraction or transformation.
$\mathrm{margin}_h(\mathcal{S},\mathcal{W})$	The hypothesis margin regarding a set of prototypes $\mathcal{W}$ and with respect to a set of inputs $\mathcal{S}$.

$\mathrm{margin}_s(\mathcal{S}, \mathcal{W})$	The sample margin regarding a set of prototypes $\mathcal{W}$ and with respect to a set of inputs $\mathcal{S}$.
$\text{median-}\delta_{\mathfrak{a}}(\mathcal{T})$	The median adversarial distance of the attack $\mathfrak{a}$ on the dataset $\mathcal{T}$.
$\text{median-}\delta_p^*(\mathcal{T})$	The worst-case median adversarial distance of L^p-attacks on the dataset $\mathcal{T}$.
n_0	The dimension of a receptive field.
n_κ	The dimension of a component $\boldsymbol{\kappa}$.
n_s	The dimension of a linear subspace, an affine subspace, or an orthotope.
n_x	The dimension of an input $\mathbf{x}$.
$p_c(\mathbf{x})$	The class hypothesis probability of the class c for an input $\mathbf{x}$.
$p^+(\mathbf{x})$	The class hypothesis probability of the correct class $c(\mathbf{x})$ given an input $(\mathbf{x}, c(\mathbf{x}))$.
$p^-(\mathbf{x})$	The highest class hypothesis probability of a class unlike $c(\mathbf{x})$ given an input $(\mathbf{x}, c(\mathbf{x}))$.
$\mathbf{p}(\mathbf{x})$	The class hypothesis possibility vector for an input $\mathbf{x}$.
$r_{c,k}^+$, $r_{c,k}^0$, $r_{c,k}^-$	The positive, indefinite, and negative reasoning probability of class c and component $\boldsymbol{\kappa}_k$.
$\mathbf{r}_c^+$, $\mathbf{r}_c^0$, $\mathbf{r}_c^-$	The positive, indefinite, and negative reasoning possibility vector or stack of class c.
$\bar{\mathbf{r}}_c^+$, $\bar{\mathbf{r}}_c^-$	The positive and negative effective reasoning possibility vector or stack of class c.
$\mathrm{sigmoid}_\sigma(\cdot)$	A sigmoid function, especially the logistic function.
t_p, t_0, t_2, t_∞	The threshold parameters for the adversarial threshold accuracies.
t_{p^*}, t_β	The prediction-reject and margin-reject threshold parameter.
$\mathbf{t}$	A translation vector of an affine subspace.
v_κ	The vertical spatial dimension of a component $\boldsymbol{\kappa}$.
v_d	The vertical spatial dimension of a detection possibility stack.

v_p	The vertical spatial dimension of a class hypothesis possibility stack.
v_r	The vertical spatial dimension of a reasoning stack.
v_x	The vertical spatial dimension of an input $\mathbf{x}$.
v'_κ	The vertical spatial dimension of a component $\boldsymbol{\kappa}$ after feature extraction.
v'_x	The vertical spatial dimension of an input $\mathbf{x}$ after feature extraction.
$\mathbf{w}$	The variable for a prototype vector.
$\boldsymbol{\mathfrak{w}}$	The variable for a set-prototype.
$\mathbf{w}^+$, $\boldsymbol{\mathfrak{w}}^+$	The closest prototype of the correct class $c(\mathbf{x})$ regarding $\mathbf{x}$.
$\mathbf{w}^-$, $\boldsymbol{\mathfrak{w}}^-$	The closest prototype of a class other than $c(\mathbf{x})$ regarding $\mathbf{x}$.
$\mathbf{w}^*$, $\boldsymbol{\mathfrak{w}}^*$	The closest or best matching prototype regarding $\mathbf{x}$.
$\triangle\mathbf{w}^\pm$	The signed gradients of a dissimilarity with respect to the prototypes $\mathbf{w}^\pm$.
$\mathbf{x}$	The variable for an input.
$\tilde{\mathbf{x}}$	An adversarial example of $\mathbf{x}$.

Other symbols

$\mathbf{0}$	A vector of zeros.
$\mathbf{1}$	A vector of ones.
$\|\cdot\|$	A norm.
$\|\cdot\|_E$, $\|\cdot\|_2$	The Euclidean norm.
$\|\cdot\|_F$	The Frobenius norm.
$\|\cdot\|_p$	A p-norm or L^p-norm.
$\circ$	The Hadamard product (element-wise multiplication).

Acronyms

BMPP	Best Matching Prototype Principle, see Section 2.1.2.
C&W	Carlini & Wagner attack, see Section 3.5.2 and Section 4.5.3.
CBC	Classification-By-Components network, see Chapter 4.
CNN	Convolutional Neural Network, see Section 4.3.
DP	Decomposition Plan, see Chapter 4.
FGSM	Fast Gradient Sign Method, see Section 3.5.2 and Section 4.5.3.
FP	False Positive, see Section 4.5.3.
GLVQ	Generalized Learning Vector Quantization, see Section 2.2.2.
GMLVQ	Generalized Matrix Learning Vector Quantization, see Section 2.2.3.
GTLVQ	Generalized Tangent Learning Vector Quantization, see Section 2.2.3.
LVQ	Learning Vector Quantization, see Section 1.1 and Chapter 2.
ML	Machine Learning, see Chapter 1.
NN	Neural Network, see Chapter 1.
PGD	Projected Gradient Descent, see Section 3.5.2 and Section 4.5.3.
RBC	Recognition-By-Components, see Chapter 4.
ResNet	Residual neural Network, see Chapter 4.
S&P	Salt & Pepper noise attack, see Section 3.5.2 and Section 4.5.3.
SGD	Stochastic Gradient Descent, see Section 1.1 and Section 2.2.2.
TP	True Positive, see Section 4.5.3.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, *7*(1), 39–59. doi:10.3233/AIC-1994-7104

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. In S. B. H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2018* (pp. 9505–9515). Montréal, QC, Canada: Curran Associates, Inc. Retrieved from https://papers.nips.cc/paper/8160-sanity-checks-for-saliency-maps

Akata, Z., Perronnin, F., Harchaoui, Z., & Schmid, C. (2013). Label-embedding for attribute-based classification. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition – CVPR 2013* (pp. 819–826). doi:10.1109/CVPR.2013.111

Arik, S. O., & Pfister, T. (2019). *Attention-based prototypical learning towards interpretable, confident and robust deep neural networks.* arXiv: 1902.06292v1 [cs.LG]

Arpit, D., Campos, V., & Bengio, Y. (2019). How to initialize your network? Robust initialization for WeightNorm & ResNets. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2019* (pp. 10902–10911). Vancouver, BC, Canada: Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/9272-how-to-initialize-your-network-robust-initialization-for-weightnorm-resnets

Baumgardner, M. F., Biehl, L. L., & Landgrebe, D. A. (2015). 220 band AVIRIS hyperspectral image data set: June 12, 1992 Indian Pine test site 3. doi:10.4231/R7RX991C

Bengio, Y., & Monperrus, M. (2005). Non-local manifold tangent learning. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in Neural Information Processing Systems 17: Proceedings of the Neural Information Processing Systems Conference – NIPS 2004* (pp. 129–136). Vancouver, BC, Canada: MIT Press. Retrieved from `http://papers.nips.cc/paper/2647-non-local-manifold-tangent-learning`

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. S. (2016). Fully-convolutional Siamese networks for object tracking. In G. Hua & H. Jégou (Eds.), *Workshop proceedings of the European Conference on Computer Vision – ECCV 2016 Workshops* (Vol. 9914 of the Lecture Notes in Computer Science, pp. 850–865). doi:`10.1007/978-3-319-48881-3_56`

Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, *94*(2), 115–147. doi:`10.1037/0033-295X.94.2.115`

Biehl, M., Hammer, B., Schleif, F.-M., Schneider, P., & Villmann, T. (2015). Stationarity of matrix relevance LVQ. In *Proceedings of the 2015 International Joint Conference on Neural Networks – IJCNN 2015* (pp. 1–8). doi:`10.1109/IJCNN.2015.7280441`

Biehl, M., Hammer, B., & Villmann, T. (2016). Prototype-based models in machine learning. *Wiley Interdisciplinary Reviews Cognitive Science*, *7*(2), 92–111. doi:`10.1002/wcs.1378`

Bien, J., & Tibshirani, R. (2011). Prototype selection for interpretable classification. *The Annals of Applied Statistics*, *5*(4), 2403–2424. doi:`10.1214/11-AOAS495`

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in Neural Information Processing Systems 20: Proceedings of the Neural Information Processing Systems Conference – NIPS 2007* (pp. 161–168). Vancouver, BC, Canada: Curran Associates, Inc. Retrieved from `https://papers.nips.cc/paper/3323-the-tradeoffs-of-large-scale-learning`

Bouchacourt, D., & Denoyer, L. (2019). *EDUCE: Explaining model decisions through unsupervised concepts extraction.* arXiv: `1905.11852v1 [cs.LG]`

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization.* doi:`10.1017/CBO9780511804441`

Brendel, W., Rauber, J., & Bethge, M. (2018). Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *Proceedings of the 6th International Conference on Learning Representations – ICLR 2018*, Vancouver, BC, Canada: OpenReview.net. Retrieved from `https://openreview.net/forum?id=SyZI0GWCZ`

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1994). Signature verification using a "Siamese" time delay neural network. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in Neural Information Processing Systems 6: Proceedings of the Neural Information Processing Systems Conference – NIPS 1993* (pp. 737–744). Denver, CO, USA: Morgan Kaufmann. Retrieved from `https://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network`

Bunte, K., Hammer, B., Wismüller, A., & Biehl, M. (2010). Adaptive local dissimilarity measures for discriminative dimension reduction of labeled data. *Neurocomputing*, *73*(7-9), 1074–1092. doi:`10.1016/j.neucom.2009.11.017`

Bunte, K., Schneider, P., Hammer, B., Schleif, F.-M., Villmann, T., & Biehl, M. (2012). Limited rank matrix learning, discriminative dimension reduction and visualization. In *Neural networks* (Vol. 26, pp. 159–173). doi:`10.1016/j.neunet.2011.10.001`

Carlini, N. (2019). *Is AmI (Attacks Meet Interpretability) robust to adversarial examples?* arXiv: `1902.02322v1 [cs.LG]`

Carlini, N., & Wagner, D. A. (2017). Towards evaluating the robustness of neural networks. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy – SP 2017* (pp. 39–57). doi:`10.1109/SP.2017.49`

Chen, C., Li, O., Tao, D., Barnett, A., Su, J., & Rudin, C. (2019). This looks like that: Deep learning for interpretable image recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2019* (pp. 8928–8939). Vancouver, BC, Canada: Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/9095-this-looks-like-that-deep-learning-for-interpretable-image-recognition`

Chen, X., Li, L.-J., Fei-Fei, L., & Gupta, A. (2018). Iterative visual reasoning beyond convolutions. In *Proceedings of the 2018 IEEE/CVF Conference on Computer*

Vision and Pattern Recognition – CVPR 2018 (pp. 7239–7248). doi:`10.1109/CVPR.2018.00756`

Chi, Y. (2013). Nearest subspace classification with missing data. In *Proceedings of the 2013 Asilomar Conference on Signals, Systems and Computers – ACSSC 2013* (pp. 1667–1671). doi:`10.1109/ACSSC.2013.6810583`

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition – CVPR 2005* (pp. 539–546). doi:`10.1109/CVPR.2005.202`

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., & Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning – ICML 2017* (Vol. 70 of the Proceedings of Machine Learning Research, pp. 854–863). Sydney, NSW, Australia: PMLR. Retrieved from `http://proceedings.mlr.press/v70/cisse17a.html`

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by Exponential Linear Units (ELUs). In Y. Bengio & Y. LeCun (Eds.), *Proceedings of the 4th International Conference on Learning Representations – ICLR 2016*, San Juan, Puerto Rico. Retrieved from `http://arxiv.org/abs/1511.07289`

Crammer, K., Gilad-Bachrach, R., Navot, A., & Tishby, N. (2003). Margin analysis of the LVQ algorithm. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15: Proceedings of the Neural Information Processing Systems Conference – NIPS 2002* (pp. 479–486). Vancouver, BC, Canada: MIT Press. Retrieved from `http://papers.nips.cc/paper/2261-margin-analysis-of-the-lvq-algorithm`

Croce, F., Andriushchenko, M., & Hein, M. (2019). Provable robustness of ReLU networks via maximization of linear regions. In K. Chaudhuri & M. Sugiyama (Eds.), *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics – AISTATS 2019* (Vol. 89 of the Proceedings of Machine Learning Research, pp. 2057–2066). Naha, Okinawa, Japan: PMLR. Retrieved from `http://proceedings.mlr.press/v89/croce19a.html`

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Con-*

ference on Computer Vision and Pattern Recognition – CVPR 2009 (pp. 248–255). doi:10.1109/CVPRW.2009.5206848

Devos, A., & Grossglauser, M. (2019). *Subspace networks for few-shot classification.* arXiv: 1905.13613v1 [cs.LG]

Elsayed, G. F., Krishnan, D., Mobahi, H., Regan, K., & Bengio, S. (2018). Large margin deep networks for classification. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2018* (pp. 842–852). Montréal, QC, Canada: Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/7364-large-margin-deep-networks-for-classification

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). *Visualizing higher-layer features of a deep network* (tech. rep. No. 1341). University of Montreal. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, QC, Canada. Retrieved from http://www.iro.umontreal.ca/~lisa/publications2/index.php/publications/show/247

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., … Son, D. (2018). Robust physical-world attacks on deep learning visual classification. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2018* (pp. 1625–1634). doi:10.1109/CVPR.2018.00175

Fukui, K., & Maki, A. (2015). Difference subspace and its generalization for subspace-based methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(11), 2164–2177. doi:10.1109/TPAMI.2015.2408358

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2019). ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *Proceedings of the 7th International Conference on Learning Representations – ICLR 2019*, New Orleans, LA, USA: OpenReview.net. Retrieved from https://openreview.net/forum?id=Bygh9j09KX

Ghiasi-Shirazi, K. (2019). Generalizing the convolution operator in convolutional neural networks. *Neural Processing Letters*, *50*(3), 2627–2646. doi:10.1007/s11063-019-10043-7

Gidaris, S., & Komodakis, N. (2018). Dynamic few-shot visual learning without forgetting. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision* *and Pattern Recognition – CVPR 2018* (pp. 4367–4375). doi:10.1109/CVPR.2018.00459

Globerson, A., & Roweis, S. T. (2006). Metric learning by collapsing classes. In Y. Weiss, B. Schölkopf, & J. C. Platt (Eds.), *Advances in Neural Information Processing Systems 18: Proceedings of the Neural Information Processing Systems Conference – NIPS 2005* (pp. 451–458). Vancouver, BC, Canada: MIT Press. Retrieved from http://papers.nips.cc/paper/2947-metric-learning-by-collapsing-classes

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In Y. Bengio & Y. LeCun (Eds.), *Proceedings of the 3rd International Conference on Learning Representations – ICLR 2015*, San Diego, CA, USA. Retrieved from http://arxiv.org/abs/1412.6572

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Adaptive computation and machine learning. Also online available at http://www.deeplearningbook.org. MIT Press.

Haasdonk, B., & Keysers, D. (2002). Tangent distance kernels for support vector machines. In *Proceedings of the 16th International Conference on Pattern Recognition – ICPR 2002* (pp. 864–868). doi:10.1109/ICPR.2002.1048439

Hammer, B., Nebel, D., Riedel, M., & Villmann, T. (2014). Generative versus discriminative prototype based classification. In T. Villmann, F.-M. Schleif, M. Kaden, & M. Lange (Eds.), *Advances in Self-Organizing Maps and Learning Vector Quantization: Proceedings of the 10th International Workshop, WSOM 2014* (Vol. 295 of the Advances in Intelligent Systems and Computing, pp. 123–132). doi:10.1007/978-3-319-07695-9_12

Hammer, B., Strickert, M., & Villmann, T. (2005). On the generalization ability of GRLVQ networks. *Neural Processing Letters*, *21*(2), 109–120. doi:10.1007/s11063-004-1547-1

Hammer, B., & Villmann, T. (2002). Generalized relevance learning vector quantization. *Neural Networks*, *15*(8-9), 1059–1068. doi:10.1016/S0893-6080(02)00079-5

Hastie, T., Simard, P., & Säckinger, E. (1995). Learning prototype models for tangent distance. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), *Advances in Neural Information Processing Systems 7: Proceedings of the Neural Information Processing Systems Conference – NIPS 1994* (pp. 999–1006). Denver, CO, USA: MIT Press. Retrieved from http://papers.nips.cc/paper/939-learning-prototype-models-for-tangent-distance

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition – CVPR 2016* (pp. 770–778). doi:10.1109/CVPR.2016.90

Hein, M., & Andriushchenko, M. (2017). Formal guarantees on the robustness of a classifier against adversarial manipulation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Proceedings of the Neural Information Processing Systems Conference – NIPS 2017* (pp. 2266–2276). Long Beach, CA, USA: Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/6821-formal-guarantees-on-the-robustness-of-a-classifier-against-adversarial-manipulation

Hein, M., Andriushchenko, M., & Bitterwolf, J. (2019). Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2019* (pp. 41–50). doi:10.1109/CVPR.2019.00013

Henrikson, J. (1999). Completeness and total boundedness of the Hausdorff metric. *MIT Undergraduate Journal of Mathematics*, *1*, 69–80.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., & Madry, A. (2019). Adversarial examples are not bugs, they are features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2019* (pp. 125–136). Vancouver, BC, Canada: Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/8307-adversarial-examples-are-not-bugs-they-are-features

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning – ICML 2015* (Vol. 37 of the Proceedings of Machine Learning Research, pp. 448–456). Lille, France: PMLR. Retrieved from http://proceedings.mlr.press/v37/ioffe15.html

Jiang, C., Xu, H., Liang, X., & Lin, L. (2018). Hybrid knowledge routed modules for large-scale object detection. In S. Bengio, H. M. Wallach, H. Larochelle, K.

Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2018* (pp. 1552–1563). Montréal, QC, Canada: Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/7428-hybrid-knowledge-routed-modules-for-large-scale-object-detection`

Keysers, D. (2000). *Approaches to invariant image object recognition* (Diploma thesis, Lehrstuhl für Informatik VI, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, Germany). Retrieved from `http://www.keysers.net/daniel/`

Keysers, D., Macherey, W., Ney, H., & Dahmen, J. (2004). Adaptation in statistical pattern recognition using tangent vectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(2), 269–274. doi:`10.1109/TPAMI.2004.1262198`

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *Proceedings of the 3rd International Conference on Learning Representations – ICLR 2015*, San Diego, CA, USA. Retrieved from `http://arxiv.org/abs/1412.6980`

Koch, G. (2015). *Siamese neural networks for one-shot image recognition* (Master's thesis, Department of Computer Science, University of Toronto, Toronto, ON, Canada). Retrieved from `http://www.cs.toronto.edu/~gkoch/files/msc-thesis.pdf`

Kohonen, T. (1990). Improved versions of learning vector quantization. In *Proceedings of the 1990 International Joint Conference on Neural Networks – IJCNN 1990* (pp. 545–550). doi:`10.1109/ijcnn.1990.137622`

Kohonen, T. (1995). Self-organizing maps. (Chap. Learning Vector Quantization, Vol. 30 of the Springer Series in Information Sciences, pp. 175–189). doi:`10.1007/978-3-642-97610-0_6`

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images* (tech. rep., Department of Computer Science, University of Toronto, Toronto, ON, Canada). Retrieved from `https://www.cs.toronto.edu/~kriz/`

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25: Proceedings of the Neural Information Processing Systems Conference – NIPS 2012* (pp. 1097–1105). Lake Tahoe, NV, USA: Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*, 1–72. E253. doi:10.1017/S0140525X16001837

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. doi:10.1109/5.726791

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K.-R. (2012). Efficient backprop. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade* (Vol. 7700 of the Lecture Notes in Computer Science, pp. 9–48). doi:10.1007/978-3-642-35289-8_3

Li, O., Liu, H., Chen, C., & Rudin, C. (2018). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In S. A. McIlraith & K. Q. Weinberger (Eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence – AAAI 2018* (pp. 3530–3537). New Orleans, LA, USA: AAAI Press. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17082

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110. doi:10.1023/B:VISI.0000029664.99615.94

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *Proceedings of the 6th International Conference on Learning Representations – ICLR 2018*, Vancouver, BC, Canada: OpenReview.net. Retrieved from https://openreview.net/forum?id=rJzIBfZAb

Marino, K., Salakhutdinov, R., & Gupta, A. (2017). The more you know: Using knowledge graphs for image classification. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition – CVPR 2017* (pp. 20–28). doi:10.1109/CVPR.2017.10

Mensink, T., Verbeek, J. J., Perronnin, F., & Csurka, G. (2012). Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, & C. Schmid (Eds.), *Proceedings of the 12th European Conference on Computer Vision –*

ECCV 2012 (Vol. 7573 of the Lecture Notes in Computer Science, pp. 488–501). doi:`10.1007/978-3-642-33709-3_35`

Mi, J.-X., Huang, D.-S., Wang, B., & Zhu, X. (2013). The nearest-farthest subspace classification for face recognition. *Neurocomputing*, *113*, 241–250. doi:`10.1016/j.neucom.2013.01.003`

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*(11), 39–41. doi:`10.1145/219717.219748`

Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016). DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition – CVPR 2016* (pp. 2574–2582). doi:`10.1109/CVPR.2016.282`

Nar, K., Ocal, O., Sastry, S. S., & Ramchandran, K. (2019). *Cross-entropy loss and low-rank features have responsibility for adversarial examples.* arXiv: `1901.08360v1 [cs.LG]`

Nebel, D., Kaden, M., Villmann, A., & Villmann, T. (2017). Types of (dis-)similarities and adaptive mixtures thereof for improved classification learning. *Neurocomputing*, *268*, 42–54. doi:`10.1016/j.neucom.2016.12.091`

Nguyen, A., Yosinski, J., & Clune, J. (2019). Understanding neural networks via feature visualization: A survey. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, & K.-R. Müller (Eds.), *Explainable AI: Interpreting, explaining and visualizing deep learning* (Vol. 11700 of the Lecture Notes in Computer Science, pp. 55–76). doi:`10.1007/978-3-030-28954-6_4`

Nova, D., & Estévez, P. A. (2014). A review of learning vector quantization classifiers. *Neural Computing and Applications*, *25*(3-4), 511–524. doi:`10.1007/s00521-013-1535-3`

Papernot, N., & McDaniel, P. (2018). *Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning.* arXiv: `1803.04765v1 [cs.LG]`

Plötz, T., & Roth, S. (2018). Neural nearest neighbors networks. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2018* (pp. 1087–1098). Montréal, QC, Canada: Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/7386-neural-nearest-neighbors-networks`

Ramachandran, P., Zoph, B., & Le, Q. V. (2018). Searching for activation functions. In *Workshop proceedings of the 6th International Conference on Learning Representations – ICLR 2018*, Vancouver, BC, Canada: OpenReview.net. Retrieved from `https://openreview.net/forum?id=Hkuq2EkPf`

Rauber, J., Brendel, W., & Bethge, M. (2017). *Foolbox: A Python toolbox to benchmark the robustness of machine learning models.* arXiv: `1707.04131v3 [cs.LG]`

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*, 206–215. doi:`10.1038/s42256-019-0048-x`

Salakhutdinov, R., & Hinton, G. (2007). Learning a nonlinear embedding by preserving class neighbourhood structure. In M. Meila & X. Shen (Eds.), *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics – AISTATS 2007* (Vol. 2 of the Proceedings of Machine Learning Research, pp. 412–419). San Juan, Puerto Rico: PMLR. Retrieved from `http://proceedings.mlr.press/v2/salakhutdinov07a.html`

Salimans, T., & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 29: Proceedings of the Neural Information Processing Systems Conference – NIPS 2016* (pp. 901–909). Barcelona, Spain: Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/6114-weight-normalization-a-simple-reparameterization-to-accelerate-training-of-deep-neural-networks`

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, *2*(6), 459–473. doi:`10.1016/0893-6080(89)90044-0`

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., & Lillicrap, T. P. (2016). Meta-learning with memory-augmented neural networks. In M.-F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd International Conference on Machine Learning – ICML 2016* (Vol. 48 of the Proceedings of Machine Learning Research, pp. 1842–1850). New York, NY, USA: PMLR. Retrieved from `http://proceedings.mlr.press/v48/santoro16.html`

Sato, A., & Yamada, K. (1996). Generalized learning vector quantization. In D. S. Touretzky, M. Mozer, & M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8: Proceedings of the Neural Information Process-*

ing Systems Conference – NIPS 1995 (pp. 423–429). Denver, CO, USA: MIT Press. Retrieved from `http://papers.nips.cc/paper/1113-generalized-learning-vector-quantization`

Schneider, P., Biehl, M., & Hammer, B. (2009). Adaptive relevance matrices in learning vector quantization. *Neural Computation*, *21*(12), 3532–3561. doi:`10.1162/neco.2009.11-08-908`

Schneider, P., Bunte, K., Stiekema, H., Hammer, B., Villmann, T., & Biehl, M. (2010). Regularization in matrix relevance learning. *IEEE Transactions on Neural Networks*, *21*(5), 831–840. doi:`10.1109/TNN.2010.2042729`

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Adaptive Computation and Machine Learning. MIT Press.

Schönemann, P. H. (1966). A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, *31*(1), 1–10. doi:`10.1007/BF02289451`

Schott, L., Rauber, J., Bethge, M., & Brendel, W. (2019). Towards the first adversarially robust neural network model on MNIST. In *Proceedings of the 7th International Conference on Learning Representations – ICLR 2019*, New Orleans, LA, USA: OpenReview.net. Retrieved from `https://openreview.net/forum?id=S1EHOsC9tX`

Schwenk, H., & Milgram, M. (1995). Learning discriminant tangent models for handwritten character recognition. In *Proceedings of the International Conference on Artificial Neural Networks – ICANN 1995* (pp. 585–590). Paris, France: Springer-Verlag.

Seo, S., & Obermayer, K. (2003). Soft learning vector quantization. *Neural Computation*, *15*(7), 1589–1604. doi:`10.1162/089976603321891819`

Shafer, G. (1976). *A mathematical theory of evidence*. Princeton University Press.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*, 354–359. doi:`10.1038/nature24270`

Simard, P., LeCun, Y., & Denker, J. S. (1993). Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in Neural Information Processing Systems 5: Proceedings of the Neural Information Processing Systems Conference – NIPS 1992* (pp. 50–58). Denver, CO, USA: Morgan Kaufmann. Retrieved from `http://papers.`

nips.cc/paper/656-efficient-pattern-recognition-using-a-new-transformation-distance

Singla, S., & Feizi, S. (2019). *Robustness certificates against adversarial examples for ReLU networks*. arXiv: 1902.01235v2 [cs.LG]

Slade, S. (1991). Case-based reasoning: A research paradigm. *AI Magazine*, *12*(1), 42–55. doi:10.1609/aimag.v12i1.883

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Proceedings of the Neural Information Processing Systems Conference – NIPS 2017* (pp. 4077–4087). Long Beach, CA, USA: Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning

Sona, D., Sperduti, A., & Starita, A. (2000). Discriminant pattern recognition using transformation-invariant neurons. *Neural Computation*, *12*(6), 1355–1370. doi:10.1162/089976600300015402

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, *32*, 323–332. doi:10.1016/j.neunet.2012.02.016

Stutz, D., Hein, M., & Schiele, B. (2019). Disentangling adversarial robustness and generalization. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2019* (pp. 6969–6980). doi:10.1109/CVPR.2019.00714

Suárez, J. L., García, S., & Herrera, F. (2018). *A tutorial on distance metric learning: Mathematical foundations, algorithms and software*. arXiv: 1812.05944v2 [cs.LG]

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In Y. Bengio & Y. LeCun (Eds.), *Proceedings of the 2nd International Conference on Learning Representations – ICLR 2014*, Banff, AB, Canada. Retrieved from http://arxiv.org/abs/1312.6199

Tokmakov, P., Wang, Y.-X., & Hebert, M. (2019). Learning compositional representations for few-shot recognition. In *Proceedings of the IEEE International*

Conference on Computer Vision – ICCV 2019 (pp. 6372–6381). Seoul, South Korea: IEEE.

Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(11), 1958–1970. doi:10.1109/TPAMI.2008.128

Villmann, T., Bohnsack, A., & Kaden, M. (2017). Can learning vector quantization be an alternative to SVM and deep learning? - recent trends and advanced variants of learning vector quantization for classification learning. *Journal of Artificial Intelligence and Soft Computing Research*, *7*(1), 65–81. doi:10.1515/jaiscr-2017-0005

Villmann, T., Ravichandran, J., Villmann, A., Nebel, D., & Kaden, M. (2019). Investigation of activation functions for generalized learning vector quantization. In A. Vellido, K. Gibert, C. Angulo, & J. D. Martín-Guerrero (Eds.), *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization: Proceedings of the 13th International Workshop, WSOM+ 2019* (Vol. 976 of the Advances in Intelligent Systems and Computing, pp. 179–188). doi:10.1007/978-3-030-19642-4_18

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 29: Proceedings of the Neural Information Processing Systems Conference – NIPS 2016* (pp. 3630–3638). Barcelona, Spain: Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning

Wang, D., Li, C., Wen, S., Nepal, S., & Xiang, Y. (2019). *Daedalus: Breaking non-maximum suppression in object detection via adversarial examples.* arXiv: 1902.02067v2 [cs.CV]

Wang, X., & Tang, X. (2004). A unified framework for subspace face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(9), 1222–1228. doi:10.1109/TPAMI.2004.57

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*(4), 600–612. doi:10.1109/TIP.2003.819861

Wei, C., & Ma, T. (2019). *Improved sample complexities for deep networks and robust classification via an all-layer margin.* arXiv: `1910.04284v1 [cs.LG]`

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, *10*, 207–244.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15: Proceedings of the Neural Information Processing Systems Conference – NIPS 2002* (pp. 521–528). Vancouver, BC, Canada: MIT Press. Retrieved from `http://papers.nips.cc/paper/2164-distance-metric-learning-with-application-to-clustering-with-side-information`

Yang, H.-M., Zhang, X.-Y., Yin, F., & Liu, C.-L. (2018). Robust classification with convolutional prototype learning. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2018* (pp. 3474–3482). doi:`10.1109/CVPR.2018.00366`

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In D. J. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Proceedings of the 13th European Conference on Computer Vision – ECCV 2014* (Vol. 8689 of the Lecture Notes in Computer Science, pp. 818–833). doi:`10.1007/978-3-319-10590-1_53`

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., & Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. In S. B. H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2018* (pp. 4939–4948). Montréal, QC, Canada: Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/7742-efficient-neural-network-robustness-certification-with-general-activation-functions`

Zhang, L., Edraki, M., & Qi, G.-J. (2018, May 19). CapProNet: Deep feature learning via orthogonal projections onto capsule subspaces. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2018* (pp. 5814–5823). Montréal, QC, Canada: Curran Associates, Inc. arXiv: `1805.07621v1 [cs.CV]`. Retrieved from `http://papers.nips.cc/paper/7823-cappronet-`

deep-feature-learning-via-orthogonal-projections-onto-capsule-subspaces

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition – CVPR 2016* (pp. 2921–2929). doi:10.1109/CVPR.2016.319

Zhu, R., Fukui, K., & Xue, J.-H. (2017). Building a discriminatively ordered subspace on the generating matrix to classify high-dimensional spectral data. *Information Sciences*, *382–383*, 1–14. doi:10.1016/j.ins.2016.12.001