

# RECURRENT NEURAL NETWORKS FOR POLYPHONIC SOUND EVENT DETECTION IN REAL LIFE RECORDINGS

Giambattista Parascandolo, Heikki Huttunen, Tuomas Virtanen

Department of Signal Processing, Tampere University of Technology

## ABSTRACT

In this paper we present an approach to polyphonic sound event detection in real life recordings based on bi-directional long short term memory (BLSTM) recurrent neural networks (RNNs). A single multilabel BLSTM RNN is trained to map acoustic features of a mixture signal consisting of sounds from multiple classes, to binary activity indicators of each event class. Our method is tested on a large database of real-life recordings, with 61 classes (*e.g.* music, car, speech) from 10 different everyday contexts. The proposed method outperforms previous approaches by a large margin, and the results are further improved using data augmentation techniques. Overall, our system reports an average  $F1$ -score of 65.5% on 1 second blocks and 64.7% on single frames, a relative improvement over previous state-of-the-art approach of 6.8% and 15.1% respectively.

**Index Terms**— Recurrent neural network, bidirectional LSTM, deep learning, polyphonic sound event detection

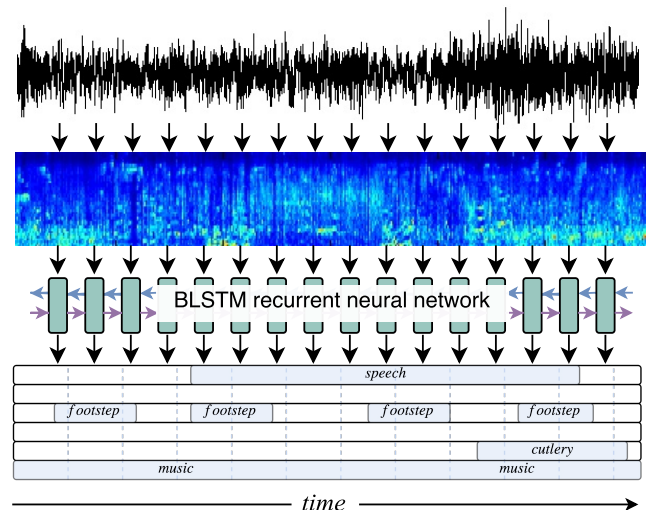
## 1. INTRODUCTION

Sound event detection (SED), also known as acoustic event detection, deals with the identification of sound events in audio recordings. The goal is to estimate start and end times of sound events, and to give a label for each event. Applications of SED include for example acoustic surveillance [1], environmental context detection [2] and automatic audio indexing [3].

SED in single-source environment is called *monophonic* detection, which has been the major area of research in this field [4]. However, in a typical real environment it is uncommon to have only a single sound source emitting at a certain point in time; it is more likely that multiple sound sources are emitting simultaneously, thus resulting in an additive combination of sounds. Due to the presence of multiple and overlapping sounds, this problem is known as *polyphonic* detection, and the goal of such a SED system is to recognize for each sound event its category (*e.g.*, music, car, speech), and its beginning and ending. This task is much more challenging than the monophonic detection problem, because the sounds are overlapping and the features extracted from the mixture do not match with features calculated from sounds in isolation. Moreover, the number of sources emitting at any given moment (polyphony) is unknown and potentially large.

Initial approaches to polyphonic SED include traditional methods for speech recognition, such as the use of mel frequency cepstral coefficients (MFCCs) as features, with Gaussian mixture models (GMMs) combined with hidden Markov models (HMMs) [5, 6]. A different type of approach consists of extracting and matching the sounds in the input to templates in a dictionary of sounds. This

Tuomas Virtanen has been funded by the Academy of Finland, project no. 258708. The authors wish to acknowledge CSC IT Center for Science, Finland, for computational resources.



**Fig. 1:** Polyphonic sound event detection with BLSTM recurrent neural networks.

can be achieved through sound source separation techniques, such as non-negative matrix factorization (NMF) on time-frequency representations of the signals. NMF has been used in [7] and [8] to pre-process the signal creating a dictionary from single events, and later in [6] and [9] directly on the mixture, without learning from isolated sounds. The work in [9] was extended in [10] making learning feasible for long recordings by reducing the dictionary size.

Other approaches are based on spectrogram analysis with image processing techniques, such as the work in [11] that studies polyphonic SED using generalized Hough transform over local spectrogram features.

More recent approaches based on neural networks have been quite successful. The best results to date in polyphonic SED for real life recordings have been achieved by feedforward neural networks (FNNs), in the form of multilabel time-windowed multi layer perceptrons (MLPs), trained on spectral features of the mixture of sounds [12], temporally smoothing the outputs for continuity.

Motivated by the good performance shown by the FNN in [12], we propose to use a multilabel recurrent neural network (RNN) in the form of bi-directional long short-term memory (BLSTM) [13, 14] for polyphonic SED (Fig. 1). RNNs, contrarily to FNNs, can directly model the sequential information that is naturally present in audio. Their ability to remember past states can avoid the need for tailored postprocessing or smoothing steps. These networks have obtained excellent results on complex audio detection tasks, such as speech recognition [15] and onset detection [16] (multiclass), polyphonic

piano note transcription [17] (multilabel).

The rest of the paper is structured as follows. Section 2 presents a short introduction to RNNs and long short-term memory (LSTM) blocks. We describe in Section 3 the features used and the proposed approach. Section 4 presents the experimental set-up and results on a database of real life recordings. Finally, we present our conclusions in Section 5.

## 2. RECURRENT NEURAL NETWORKS

### 2.1. Feedforward neural networks

In a feedforward neural network (FNN) all observations are processed independently of each other. Due to the lack of context information, FNNs may have difficulties processing sequential inputs such as audio, video and text. A fixed-size (causal or non-causal) window, concatenating the current feature vector with previous (and eventually future) feature vectors, is often used to provide context to the input. This approach however presents substantial shortcomings, such as increased dimensionality (imposing the need for more data, longer training time and larger models), and short fixed context available.

### 2.2. Recurrent neural networks

Introducing feedback connections in a neural network can provide it with past context information. This network architecture is known as recurrent neural network (RNN). In an RNN, information from previous time steps can in principle circulate indefinitely inside the network through the directed cycles, where the hidden layers also act as memory. For a sequence of input vectors  $\{\mathbf{x}_1, \dots, \mathbf{x}_T\}$ , a RNN computes a sequence of hidden activations  $\{\mathbf{h}_1, \dots, \mathbf{h}_T\}$  and output vectors  $\{\mathbf{y}_1, \dots, \mathbf{y}_T\}$  as

$$\mathbf{h}_t = \mathcal{F}(\mathbf{W}^{\text{xh}}\mathbf{x}_t + \mathbf{W}^{\text{hh}}\mathbf{h}_{t-1} + \mathbf{b}^{\text{h}}) \quad (1)$$

$$\mathbf{y}_t = \mathcal{G}(\mathbf{W}^{\text{hy}}\mathbf{h}_t + \mathbf{b}^{\text{y}}) \quad (2)$$

for all timesteps  $t = 1, \dots, T$ , where the matrices  $\mathbf{W}^{**}$  denote the weights connecting two layers,  $\mathbf{b}^*$  are bias terms, and  $\mathcal{F}$  and  $\mathcal{G}$  activation functions. In case of a deep network, with multiple hidden layers, the input to hidden layer  $j$  is the output of the previous hidden layer  $j - 1$ .

When instances from future timesteps are available, also future context can be provided to the network by using bi-directional RNN (BRNN) [18]. In a BRNN each hidden layer is split into two separate layers, one reads the training sequences forwards and the other one backwards. Once fully computed, the activations are then fed to the next layer, giving the network full and symmetrical context for both past and future instances of the input sequence.

### 2.3. Long short-term memory

Standard RNNs, *i.e.*, RNNs with simple recurrent connections in each hidden layer, may be difficult to train. One of the main reasons is the phenomenon called *vanishing gradient problem* [19], which makes the influence of past inputs decay exponentially over time.

The long short-term memory (LSTM) [13] architecture was proposed as a solution to this problem. The simple neurons with static self-connections, as in a standard RNN, are substituted by units called *LSTM memory blocks* (Fig. 2). An LSTM memory block is a subnet that contains one self-connected *memory cell* with its *tanh* input and output activation functions, and three *gating neurons*—input, forget and output—with their corresponding multiplicative

units. Eq. 1, defining the hidden activation  $\mathbf{h}_t$ , is substituted by the following set of equations:

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}^{\text{xi}}\mathbf{x}_t + \mathbf{W}^{\text{hi}}\mathbf{h}_{t-1} + \mathbf{W}^{\text{ci}}\mathbf{c}_{t-1} + \mathbf{b}^{\text{i}}) \\ \mathbf{f}_t &= \sigma(\mathbf{W}^{\text{xf}}\mathbf{x}_t + \mathbf{W}^{\text{hf}}\mathbf{h}_{t-1} + \mathbf{W}^{\text{cf}}\mathbf{c}_{t-1} + \mathbf{b}^{\text{f}}) \\ \mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh(\mathbf{W}^{\text{xc}}\mathbf{x}_t + \mathbf{W}^{\text{hc}}\mathbf{h}_{t-1} + \mathbf{b}^{\text{c}}) \\ \mathbf{o}_t &= \sigma(\mathbf{W}^{\text{xo}}\mathbf{x}_t + \mathbf{W}^{\text{ho}}\mathbf{h}_{t-1} + \mathbf{W}^{\text{co}}\mathbf{c}_t + \mathbf{b}^{\text{o}}) \\ \mathbf{h}_t &= \mathbf{o}_t \tanh(\mathbf{c}_t) \end{aligned} \quad (3)$$

where  $\mathbf{c}_t$ ,  $\mathbf{i}_t$ ,  $\mathbf{f}_t$  and  $\mathbf{o}_t$  are respectively the memory cell, input gate, forget gate and output gate activations,  $\sigma$  is the logistic function,  $\mathbf{W}^{**}$  are the weight matrices and  $\mathbf{b}^*$  are bias terms.

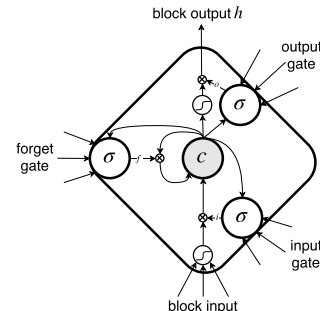


Fig. 2: An LSTM block.

By analogy, the memory cell  $c$  can be compared to a computer memory chip, and the input  $i$ , forget  $f$  and output  $o$  gating neurons represent *write*, *reset* and *read* operations respectively. All gating neurons represent binary switches but use the logistic function—thus outputting in the range  $[0, 1]$ —to preserve differentiability. Due to the multiplicative units, information can be stored over long time periods inside the cell.

A bidirectional long short-term memory (BLSTM) [14] network is obtained by substituting the simple recurrent neurons in a BRNN with LSTM units. More details about LSTM, BLSTM and training algorithms can be found in [20].

## 3. METHOD

The proposed system receives as input a raw audio signal, extracts spectral features and then maps them to binary activity indicators of each event class using a BLSTM RNN (Fig. 1). Each step is described in further detail in this section.

### 3.1. Feature extraction

The input to the system are raw audio signals. To account for different recording conditions, the amplitudes are normalized in each recording to lie in  $[-1, 1]$ . The signals are split into 50 millisecond frames with 50% overlap, and we calculate the log magnitudes within the 40 mel bands in each frame. We then normalize each frequency band by subtracting the mean value of each bin over all recordings and imposing unit variance (computing the constants on the training set), a standard procedure when working with neural networks.

For each recording we obtain a long sequence of feature vectors, which is then split into smaller sequences. We split every original sequence at three different scales, *i.e.*, in non-overlapping length 10, length 25, and length 100 sequences (corresponding to lengths of

0.25, 0.62 and 2.5 seconds respectively). This allows the network to more easily identify patterns at different timescales.

Each frame has a target vector  $\mathbf{d}$  associated, whose binary components  $d_k$  indicate if a sound event from class  $k$  is present or not.

### 3.2. Proposed neural network

We propose the use of multilabel BLSTM RNNs with multiple hidden layers to map the acoustic features to class activity indicator vectors. The output layer has logistic activation functions and one neuron for each class. We use Gaussian input noise injection and early stopping to reduce overfitting, halting the training if the cost on the validation set does not decrease for 20 epochs.

The output of the network at time  $t$  is a vector  $\mathbf{y}_t \in [0, 1]^L$ , where  $L$  is the number of classes. Its components  $y_k$  can be interpreted as posterior probabilities that each class is active or inactive in frame  $\mathbf{x}_t$ . These outputs do not have to sum up to 1, since several classes might be active simultaneously. For this reason, contrarily to most multiclass approaches with neural networks, the outputs are not normalized by computing the softmax. Finally, the continuous outputs are thresholded to obtain binary indicators of class activities for each timestep.

Contrarily to [12], where the outputs are smoothed over time using a median filter on a 10-frame window, we do not apply any post-processing since the outputs from the RNN are already smooth.

### 3.3. Data augmentation

As an additional measure to reduce overfitting, which easily arises in case the dataset is small compared to the network size, we also augment the training set by simple transformations. All transformations are applied directly to the extracted features in frequency domain.

- Time stretching: we mimic the process of slightly slowing down or speeding up the recordings. To do this, we stretch the mel spectrogram in time using linear interpolation by factors slightly smaller or bigger than 1;
- Sub-frame time shifting: we mimic small time shifts of the recordings—at sub-frame scale—linearly interpolating new feature frames in-between existing frames, thus retaining the same frame rate;
- Blocks mixing: new recordings with equal or higher polyphony can be created by combining different parts of the signals within the same context. In frequency domain we directly achieve a similar result using the mixmax principle [21], overlapping blocks of the log mel spectrogram two at the time.

Similar techniques have been used in [22, 23].

The amount of augmentation performed depends on the scarcity of the data available and the difficulty of the task. For the experiments described in Section 4—where specified—we expanded the dataset using the aforementioned techniques by approximately 16 times. A 4-fold increase comes from the time stretching (using stretching coefficients of 0.7, 0.85, 1.2, 1.5), 3-fold increase from sub-frame time shifting and 9-fold increase from blocks mixing (mixing 2 blocks at the time, using 20 non-overlapping blocks of equal size for each context). We did not test other amounts or parameters of augmentations. In order to avoid extremely long training times, the augmented data was split in length 25 sequences only.

## 4. EVALUATION

### 4.1. Dataset

We evaluate the performance of the proposed method on a database consisting of recordings 10 to 30 minutes long, from ten real-life contexts [24]. The contexts are: basketball game, beach, inside a bus, inside a car, hallway, office, restaurant, shop, street and stadium with track and field events. Each context has 8 to 14 recordings, for a total of 103 recordings (1133 minutes). The recordings were acquired with a binaural microphone at 44.1 kHz sampling rate and 24-bit resolution. The stereo signals from the recordings are converted to mono by averaging the two channels into a single one. The sound events were manually annotated within 60 classes, including speech, applause, music, break squeak, keyboard; plus 1 class for rare or unknown events marked as *unknown*, for a total of 61 classes. All the events appear multiple times in the recordings; some of them are present in different contexts, others are context-specific. The average polyphony level—*i.e.* the average number of events active simultaneously—is 2.53, the distribution of polyphony levels across all recordings is illustrated in Fig. 3.

The database was split into training, validation and test set (about 60%, 20% and 20% of the data respectively) in a 5-fold manner. All results are presented as averages of the 5-fold cross validation results, with the same train/validation/test partitions used in previous experiments on the same dataset ([10, 12]). The hyperparameters of the network, *e.g.* the number and size of hidden layers, learning rate, etc., were chosen based on validation results of the first fold.

### 4.2. Neural networks experiments

The network has an input layer with 40 units, each reading one component of the feature frames, and 4 hidden layers with 200 LSTM units each (100 reading the sequence forwards, 100 backwards). We train one network with the original data only, which is the same used in previous works, and one using the data augmentation techniques reported in Section 3.3 to further reduce overfitting. To compare the performance with standard LSTM layers, we also train a similar network architecture without bidirectional units on the same dataset without augmentation.

The network is initialised with uniformly distributed weights in  $[-0.1, 0.1]$  and trained using root mean squared error as a cost function. Training is done by back propagation through time (BPTT) [25]. The extracted features are presented as sequences clipped from the original data—in sequences of 10, 25 and 100 frames—in randomly ordered minibatches of 600 sequences, in order to allow parallel processing. After a mini-batch is processed the weights are updated using RMSProp [26]. The network is trained with a learning rate  $\eta = 0.005$ , decay rate  $\rho = 0.9$  and Gaussian input noise of

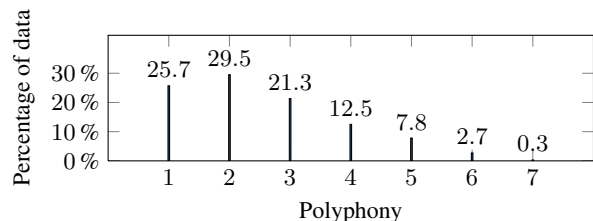


Fig. 3: Distribution of polyphony level across the dataset.

0.2 (hyperparameters chosen based on the validation set of the first fold). At test time we present the feature frames in sequences of 100 frames, and threshold the outputs with a fixed threshold of 0.5, *i.e.*, we mark an event  $k$  as active if  $y_k \geq 0.5$ , otherwise inactive.

For each experiment and each fold we train 5 networks with different random initialisations, select the one that has the highest performance on the validation set and then use it to compute the results on the test data. The networks were trained on a GPU (Tesla K40t), with the open-source toolkit Currennt [27] modified to use RMSprop.

### 4.3. Metrics

To evaluate the performance of the system we compute  $F1$ -score for each context in two ways: average of framewise  $F1$ -score ( $F1_{\text{AvgFram}}$ ) and average of  $F1$ -score in non-overlapping 1 second blocks ( $F1_{1\text{-sec}}$ ) as proposed in [4], where each target class and prediction is marked as active on the whole block if it is active in at least one frame of the block. The overall scores are computed as the average of the average scores for each context.

### 4.4. Results

In Table 1 we compare the average scores over all contexts for the FNN in [12] to our BLSTM and LSTM networks trained on the same data, and BLSTM network trained with the augmented data. The FNN uses the same features but at each timestep reads a concatenation of 5 input frames (the current frame and the two previous and two following frames). It has two hidden layers with 1600 hidden units each, downsampled to 800 with maxout activations.

The BLSTM network achieves better results than the FNN trained on the same data, improving the performance by relative 13.5% for the average framewise  $F1$  and 4.3% for the 1 second block  $F1$ . The unidirectional LSTM network does not perform as well as the BLSTM network, but is still better than the FNN. The best results are obtained by the BLSTM network trained on the augmented dataset, which improves the performance over the FNN by relative 15.1% and 6.8% for the average framewise  $F1$  and for the 1 second block  $F1$  respectively.

In Table 2 we report the results for each context for the FNN in [12] (FNN), our BLSTM trained on the same data (BLSTM) and our

**Table 1:** Overall  $F1$  scores, as average of individual contexts scores, for the FNN in [12] (FNN) compared to the proposed LSTM, BLSTM and BLSTM with data augmentation (BLSTM+DA).

| Method          | $F1_{\text{AvgFram}}$ | $F1_{1\text{-sec}}$ |
|-----------------|-----------------------|---------------------|
| FNN [12]        | 58.4%                 | 63.0%               |
| <b>LSTM</b>     | 62.5%                 | 63.8%               |
| <b>BLSTM</b>    | 64.0%                 | 64.6%               |
| <b>BLSTM+DA</b> | <b>64.7%</b>          | <b>65.5%</b>        |

BLSTM trained on the augmented data (BLSTM+DA). The results show that the proposed RNN, even without the regularisation from the data augmentation, outperforms the FNN in most of the contexts.

The  $F1$ -scores for different polyphony levels are approximately the same, showing that the method is quite robust even when several events are combined. It is interesting to notice that the RNNs have around 850K parameters each, compared to 1.65M parameters of the FNN trained with the same data. The RNNs make a more efficient and effective use of the parameters, due to the recurrent connections and the deeper structure with smaller layers.

## 5. CONCLUSIONS

In this paper we proposed to use multilabel BLSTM recurrent neural networks for polyphonic sound event detection. RNNs can directly encode context information in the hidden layers and can learn the longer patterns present in the data. Data augmentation techniques effectively reduce overfitting, further improving performance. The presented approach outperforms the previous state-of-the-art FNN [12] tested on the same large database of real-life recordings, and has half as many parameters. The average improvement on the whole data set is 15.1% for the average framewise  $F1$  and 6.8% for the 1 second block  $F1$ .

Future work will concentrate on finding novel data augmentation techniques. Concerning the model, further studies will develop on attention mechanisms and extending RNNs by coupling them with convolutional neural networks.

**Table 2:** Results for each context in the dataset for the FNN in [12] (FNN), and our approach without data augmentation (BLSTM) and with data augmentation (BLSTM+DA).

|            | $F1_{\text{AvgFram}}$ |              |              | $F1_{1\text{-sec}}$ |              |              |
|------------|-----------------------|--------------|--------------|---------------------|--------------|--------------|
|            | FNN [12]              | BLSTM        | BLSTM+DA     | FNN [12]            | BLSTM        | BLSTM+DA     |
| basketball | 70.2%                 | 77.4%        | <b>78.5%</b> | 74.7%               | 79.0%        | <b>79.9%</b> |
| beach      | <b>49.7%</b>          | 46.6%        | 49.6%        | <b>58.1%</b>        | 48.7%        | 51.5%        |
| bus        | 43.8%                 | 45.1%        | <b>49.4%</b> | <b>52.7%</b>        | 47.3%        | <b>52.7%</b> |
| car        | 53.2%                 | 67.9%        | <b>71.8%</b> | 52.4%               | 66.4%        | <b>69.5%</b> |
| hallway    | 47.8%                 | <b>58.1%</b> | 54.8%        | 55.0%               | <b>59.9%</b> | 57.1%        |
| office     | 77.4%                 | <b>79.9%</b> | 74.4%        | 77.7%               | <b>79.8%</b> | 74.8%        |
| restaurant | 69.8%                 | 76.5%        | <b>77.8%</b> | 73.7%               | 76.9%        | <b>77.7%</b> |
| shop       | 51.5%                 | <b>61.2%</b> | 61.1%        | 57.6%               | 60.9%        | <b>61.7%</b> |
| street     | 62.6%                 | <b>65.3%</b> | 65.2%        | 62.9%               | 63.3%        | <b>63.9%</b> |
| stadium    | 58.2%                 | 61.7%        | <b>64.3%</b> | 64.9%               | 64.2%        | <b>66.2%</b> |
| average    | 58.4%                 | 64.0%        | <b>64.7%</b> | 63.0%               | 64.6%        | <b>65.5%</b> |

## 6. REFERENCES

- [1] Aki Härmä, Martin F McKinney, and Janto Skowronek, “Automatic surveillance of the acoustic activity in our living environment,” in *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.
- [2] Selina Chu, Shrikanth Narayanan, and CC Jay Kuo, “Environmental sound recognition with time–frequency audio features,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1142–1158, 2009.
- [3] Min Xu, Changsheng Xu, Lingyu Duan, Jesse S Jin, and Suhuai Luo, “Audio keywords generation for sports video analysis,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 4, no. 2, pp. 11, 2008.
- [4] Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen, “Context-dependent sound event detection,” *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, pp. 1–13, 2013.
- [5] Annamaria Mesaros, Toni Heittola, Antti Eronen, and Tuomas Virtanen, “Acoustic event detection in real life recordings,” in *18th European Signal Processing Conference*, 2010, pp. 1267–1271.
- [6] Toni Heittola, Annamaria Mesaros, Tuomas Virtanen, and Moncef Gabbouj, “Supervised model training for overlapping sound events based on unsupervised source separation,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8677–8681.
- [7] Satoshi Innami and Hiroyuki Kasai, “NMF-based environmental sound source separation using time-variant gain features,” *Computers & Mathematics with Applications*, vol. 64, no. 5, pp. 1333–1342, 2012.
- [8] Arnaud Dessein, Arshia Cont, and Guillaume Lemaitre, “Real-time detection of overlapping sound events with non-negative matrix factorization,” in *Matrix Information Geometry*, pp. 341–371. Springer, 2013.
- [9] Onur Dikmen and Annamaria Mesaros, “Sound event detection using non-negative dictionaries learned from annotated overlapping events,” in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013.
- [10] Annamaria Mesaros, Toni Heittola, Onur Dikmen, and Tuomas Virtanen, “Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 606–618.
- [11] Jonathan Dennis, Huy Dat Tran, and Eng Siong Chng, “Overlapping sound event recognition using local spectrogram features and the generalised hough transform,” *Pattern Recognition Letters*, vol. 34, no. 9, pp. 1085–1093, 2013.
- [12] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen, “Polyphonic sound event detection using multi label deep neural networks,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015.
- [13] Sepp Hochreiter and Jürgen Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [14] Alex Graves and Jürgen Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.
- [15] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, “Speech recognition with deep recurrent neural networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.
- [16] Florian Eyben, Sebastian Böck, Björn Schuller, and Alex Graves, “Universal onset detection with bidirectional long short-term memory neural networks,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 589–594.
- [17] Sebastian Böck and Markus Schedl, “Polyphonic piano note transcription with recurrent neural networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 121–124.
- [18] Mike Schuster and Kuldip K Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [19] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [20] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.
- [21] Arthur Nadas, David Nahamoo, Michael Picheny, et al., “Speech recognition using noise-adaptive prototypes,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 10, pp. 1495–1503, 1989.
- [22] Jan Schlüter and Thomas Grill, “Exploring data augmentation for improved singing voice detection with neural networks,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- [23] Brian McFee, Eric J. Humphrey, and Juan P. Bello, “A software framework for musical data augmentation,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- [24] Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen, “Audio context recognition using audio event histograms,” in *Proc. of the 18th European Signal Processing Conference (EUSIPCO)*, 2010, pp. 1272–1276.
- [25] Paul J Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
- [26] Tijmen Tieleman and Geoffrey Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural Networks for Machine Learning*, vol. 4, 2012.
- [27] Felix Weninger, “Introducing currennt: The munich open-source cuda recurrent neural network toolkit,” *Journal of Machine Learning Research*, vol. 16, pp. 547–551, 2015.