JOHANNES KEPLER
UNIVERSITY LINZ

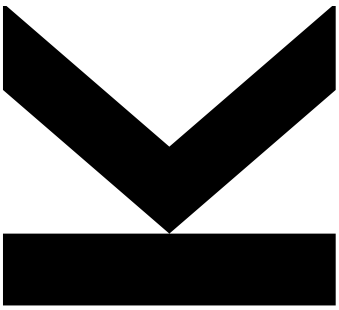
Submitted by
Mathias Wöß

Submitted at
Institute of Computational Perception

Supervisor
Paul Primus, Dipl.-Ing.

February 2021

Polyphonic Sound Event Detection using Convolutional Neural Networks



Bachelor Thesis
to obtain the academic degree of
Bachelor of Science
in the Bachelor's Program
Computer Science

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Bachelorarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Bachelorarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Ort, Datum:
Linz, 25.02.2021

Unterschrift:

ABSTRACT

Polyphonic sound event detection (SED) is still a tedious and unsolved task. The performance highly depends on the size of the dataset and the right choice of input features used. In our work, we compare popular input feature types and look into post-processing techniques for improving the results. We give a short introduction to polyphonic SED, approaches, and applied methods from peers, followed by an overview of our implemented Python framework for the third task of the 2016's DCASE Challenge, using the TUT-SED 2016 dataset and a simple convolutional neural network approach. We also give some insight into how to perform data augmentation with spectrograms. Analysing the results of our experiments, we conclude that perceptual weighting is vital to achieving high scores in terms of F1-score and error rate when attempting to predict sound events annotated by humans. Finally, we give some suggestions for future work based on our findings.

ZUSAMMENFASSUNG

Das erkennen von Events in polyphonen Audio-Dateien ist noch immer eine schwere und ungelöste Aufgabe. Die Leistung ist stark abhängig von der Größe des Datensatzes und der richtigen Wahl der Input-Features. In unserer Arbeit vergleichen wir einige beliebte Arten von Input-Features und geben einen Einblick in ein paar post-processing Methoden um die Ergebnisse zu verbessern. Wir geben eine kleine Einführung in das Erkennen von Events in polyphonen Aufnahmen, Ansätze und angewandte Methoden von anderen Arbeiten, gefolgt von einem Überblick über unser implementiertes Python-Framework für die dritte Aufgabe in der DCASE2016 challenge, mithilfe des TUT-SED 2016 Datensatzes und einem einfachen Convolutional Neuronal Network Ansatzes. Als nächstes erwähnen wir ein paar Möglichkeiten für Daten-Augmentation im Spektralbereich. Nach Analyse der Ergebnisse unserer Experimente, kommen wir zu dem Schluss das Wahrnehmungsgewichtung essentiell ist, um gute Werte für F1-score und Fehlerrate zu bekommen, wenn der Datensatz von Menschen annotiert wurde. Zuletzt geben wir noch ein paar Empfehlungen für zukünftige Arbeiten, basiert auf unseren Beobachtungen.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

1 INTRODUCTION

The human auditory system has the amazing ability to detect and classify all kinds of sounds in the constant stream of audio produced by our environment. We memorize what kind of sound a certain thing, action, animal, or machine is making. Having heard these events enough times gives us the ability to make an educated guess about what is causing the current noise we hear, no matter that the new type of sound is slightly different. Rather, we define it as belonging to a similar "sound class". For example: even if we don't know what kind of bird is singing outside the window, by experience and having heard all kinds of bird songs before, we can say that the sound coming from outside is definitely from *some* kind of bird.

This is referred to as Auditory Scene Analysis (ASA) [6, 17] and is the basis for so-called Sound Event Detection (SED) or sometimes Acoustic Event Detection (AED).

It has many applications including, speech recognition, surveillance, equipment-, medical-, and wildlife monitoring, multimedia indexing, context recognition, and video keyword tagging [3, 5, 6].

An important distinction in SED is between monophonic and polyphonic SED. Monophonic SED makes the assumption that at any moment in time, only one type of sound event is active, thus making the problem much easier. When an event is detected, the predicted sound class is selected by the highest likelihood of current audio data belonging to one out of all classes. Whereas in polyphonic SED, multiple sound sources can be active at the same time, creating a multi-label problem. At each time step we need to make a binary prediction for all possible (known) sound classes, about whether they are currently active or not.

In figure 1.1, the difference between monophonic and polyphonic sound event detection is illustrated.

Cakir et al. [4] state that monophonic event detection results in a loss of information for real-life sound scenes, and emphasize that polyphonic detection systems are needed to handle more complex and realistic system requirements. The additive nature of sound sources makes it difficult to separate active events. This resulted in many attempts and efforts to keep track of the context in the current recording and incorporate the correlations between sounds.

Parascandolo et al. [14] describe in their work, how not knowing the possibly large amount of concurrently emitting sound sources, creates an especially challenging task. This is due to the extracted features

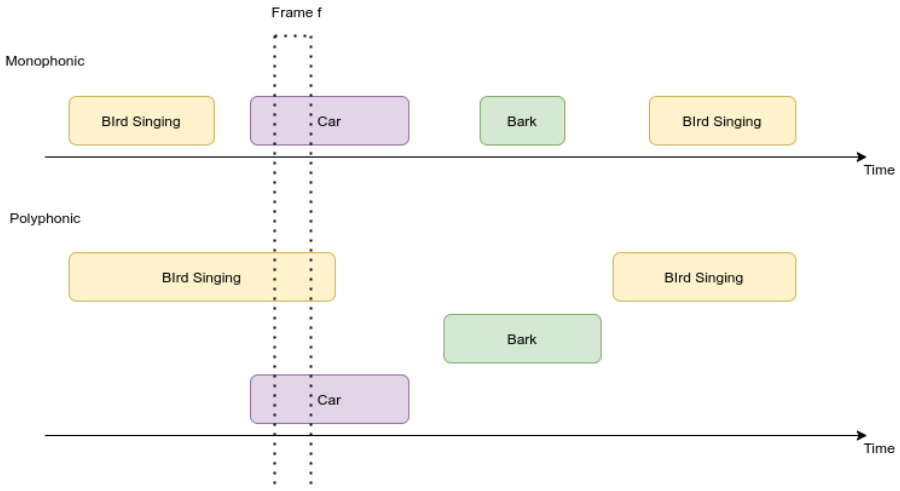


Figure 1.1: Comparison between monophonic and polyphonic SED. (inspired by Figure 2 from [1])

from mixtures of sounds not matching the ones from extraction of isolated sounds.

Chan et al. [6] gives an overview over the early concepts for SED, including making use of Hidden Markov Models (HMMs) [15] together with Gaussian Mixture Models (GMM), Non-negative Matrix Factorization (NMF) [12], and a Generalized Hough Transformation (GHT) voting systems [7]. Recent advancements in machine learning (ML) have also introduced a wide variety of Neural Networks (NN), which learn abstract feature representations from raw input data. For time series data, they can keep information in so-called hidden states, which provide the consecutive states with an accumulated representation of the (recent) past. This can help to give the prediction system more context for its final output.

Cakir et al. [4] further state that Deep Neural Networks (DNN) can outperform conventional mixture methods like HMMs, by using separate sets of its hidden units to model multiple simultaneous events in a given time instance. They further describe that this makes them superior to conventional mixture models.

Many techniques from Automatic Speech Recognition (ASR) and Image/Video processing are applicable to the SED task. As in SED, one of the most important tasks of ASR is to find onsets and offsets of speech, for recognizing when a word or sentence has begun and ended. Image and video processing systems do lots of convolutions with special-purpose kernels for smoothing, sharpening, finding edges and gradients, many of which are similarly applied to spectrograms [21].

The annual Detection and Classification of Acoustic Scenes and Events (DCASE) challenge was launched to tackle all kinds of auditory challenges like scene classification, SED, Direction of Arrival (DOA)

estimation, and sound tagging. Here, scientists and practitioners from all around the world try to surpass each other, causing research in this rather unexplored area. Many known techniques from other Machine Learning (ML) domains were adapted to fit the sound-based challenges and have shown promising results over the years.

One such Task was "Sound event detection in real-life audio" from the DCASE2016 challenge. This work and the applied methods in the implemented framework are focused on this specific task.

The challenge of polyphonic SED is to detect sound events and their onset and offset times in an audio stream. In figure 1.1, we illustrate how multiple sound sources of the same type become active and inactive at different times, with the possibility of overlap. One way to detect events in polyphonic audio is to separate the audio into different sound source streams and detect onsets and offsets separately using a detection function. Alternatively, we can look at overlapping events as a whole and train a model that can predict all events in a sound snippet.

In the context of this work, we compare different input feature types and their respective performance, using a simple CNN together with FNN output layers. We use the TUT-SED 2016 dataset [13] which consists of two separate sound scenes, each with its own distinct sound event classes (see section 4). We also wanted to find out which of the chosen feature types fit especially well for which sound scene. Another research question we wanted to answer was how a single network for each sound scene combined compares to the performance of one network for predictions in both scenes. Lastly, we investigate good pre-, and post-processing steps and data augmentation techniques to improve F-score and reduce the error rate of our network.

The concrete and formal set of research questions we tried to answer with our experiments is listed below. Most of them were answered by analysing and comparing metrics gained from running different scenarios.

- How can (CNN-based) Sound Event Detection (SED) systems based on the TUT-SED 2016 data set achieve high performance in terms of high F1-score and low error rate?
- What are the advantages/disadvantages of treating SED as a combined single-scene classification problem compared to treating it as a multi-scene classification problem? How well do they perform in comparison?
- What are the most effective post-processing steps for SED systems to achieve high performance in SED?

- Which data augmentation techniques significantly improve generalization in SED?
- Does perceptual weighting, in particular, logarithmic weighting and MEL-Band filtering, have a significant influence on system performance?
- In single-scene classification: what combination of pre-/post-processing is best suited for which particular sound scene?

The inspiration for the combined single-scene versus multi-scene comparison came from Cakir et al. [3], where they train a separate network for each sound class. Their results have shown that that combined single-label training is slightly less effective. In our work, we would like to investigate whether a network can successfully specialize on one whole sound scene.

For answers to these questions see the results and discussion in section 5.

This paper's further structure is as follows: in section 2 we give an overview of different system types used in the SED literature and refer to our main sources. In section 3 a short summary of popular feature types used for SED is given. Section 4 describes the methods and techniques used in our framework. Section 5 shows the results from the experiments performed within the framework, followed by an analysis of observations made and a discussion of our findings. Lastly our conclusions and a summary can be found in section 6.

2 TAXONOMY OF SED SYSTEMS

There are many approaches to handle the problem of polyphonic SED. Chan et al. [6] structure the different kinds of SED systems roughly into:

- Non-Neural Network Based Systems
- Neural Network Based Systems

The category of Neural Network Based Systems is then further divided into the sub-categories of:

- Non-Hybrid
- Hybrid
- Weakly Labeled

Figures 2.1 visualizes this classification:

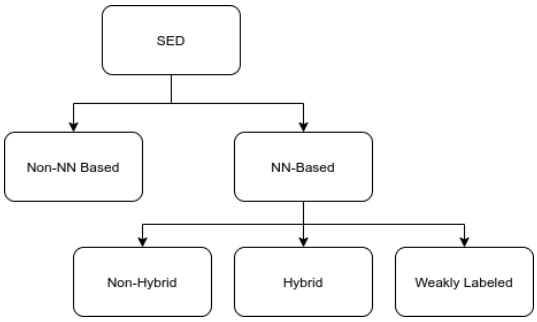


Figure 2.1: SED categories. (based on Figure 3 from [6])

2.1 NON-NEURAL NETWORK BASED SED

These methods were very popular earlier and are still used widely across all domains. They are based on likelihood, statistics, and algebra. We will give a quick overview of two of the most popular techniques and their advantages and caveats.

NON-NEGATIVE MATRIX FACTORIZATION is the process of decomposing a non-negative matrix S of size $L \times N$ into two non-negative matrices W and H. These have the shape $L \times K$ and $K \times N$, respectively. K represents the number of components, W serves as a dictionary matrix, and H as an activation matrix. For the result, $S \approx W \cdot H$ should hold (original matrix approximated well), with $\forall S_{ij}, W_{kl}, H_{mn} > 0$.

This can be formulated as a minimization problem:

$$\underset{W,H}{\operatorname{argmin}} ||S - WH||_2^2 \tag{2.1}$$

According to Chan et al. [6], W is extracted from isolated events to form a dictionary, which is then applied to the test data to derive H. By post-processing H, we get the activations of events for each timeframe.

While NMF is a powerful and easy to apply method, hugely popular in sound source separation, it has problems detecting polyphonic events.

GMM-HMM is very popular for ASR and also found its way into SED. It is based on a statistical model with hidden states and uses the Expectation-Maximization (EM) algorithm to categorize each HMM state by a GMM.

Chan et al. [6] describe that HMMs are models for a sequence of variables, where not every state is directly observable. While the inputs are observable in an HMM, the states are "hidden" and abstract. Using EM, we can categorize each of the states by a GMM, modeling the states' observations by an acoustic vector. They state that this technique is well suited for the SED task and easily scaleable to huge amounts of concurrent classes. They further write that although it seems like a fitting approach for SED, observations have shown that the achieved F1-scores and Error Rates (ER) for SED are still inferior to more modern techniques. They explain that this might be due to such systems trying to detect mainly the most prominent events, which is not directly applicable to real-life audio scenes.

While still very relevant, these approaches are more dated and have shown to achieve worse results than state-of-the-art Deep Neural Network (DNN) methods. In this paper, we try to focus on NN based methods for SED.

2.2 NEURAL NETWORK BASED SED

With the recent progress in computer hardware and advancements in ML algorithms, training DNNs has become more feasible. DNNs learn abstract representations of the input data within their many hidden layers and use these to make more sophisticated output predictions than shallow networks.

2.2.1 *Neural Networks*

Neural networks are parallel processing systems inspired by the human brain, with small computing units (neurons). In supervised learn-

ing techniques, they learn an arbitrary big matrix of weights based on input values and expected output values.

FULLY CONNECTED FEED-FORWARD NEURAL NETWORKS or Multilayer Perceptrons (MLPs) are universal function approximators [11], meaning that they can learn arbitrary complex decision functions. They feed input only forward and every neuron of each layer is connected to every neuron of the next layer.

We used these types of network for the final layers of our framework's model 4.2.

Figure 2.2 visualizes how a simple FNN looks like. x_i denote the input neurons, h_i the hidden units, o_i the output neuron(s) and w_{ij} the weigths from unit j to unit i.

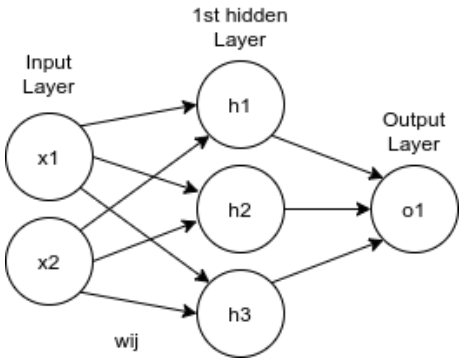


Figure 2.2: A simple FNN

RECURRENT NEURAL NETWORKS in their simplest form, are similar to FNNs, but neurons can additionally have connections to themselves or to units of the same layer. This helps to retain information from previous input steps by accumulating (numeric) information over time.

They are the basis for the LSTMs used in [14].

Figure 2.3 shows such a fully connected RNN. It is similar to figure 2.2, but has added feedback loops and intra-layer connections.

CONVOLUTIONAL NEURAL NETWORKS are a little different from previously mentioned networks. They share the same weight matrix, also called *kernel*, for each input position and are thus very cheap in terms of parameter space. The same kernel is applied to every input index using the convolution operation (actually cross-correlation) and results in a new output "pixel".

Formally the cross-correlation operation is defined as following:

$$(h * k)(a, b) = \sum_{i=0}^{R-1} \sum_{j=0}^{R-1} h(a + i, b + j) \cdot k(i, j) \tag{2.2}$$

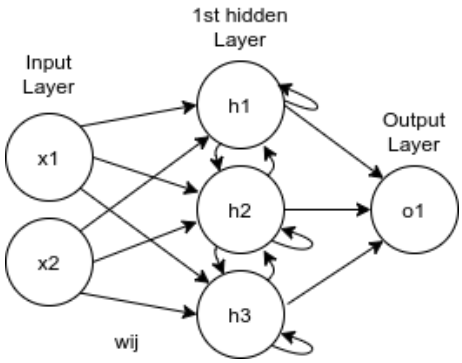


Figure 2.3: A simple RNN

Where h is the input matrix and k the kernel matrix of size R.

Figure 2.4 sketches how CNNs work. For not decreasing the feature matrix size we would have to f.e. pad zeros around the input matrix.

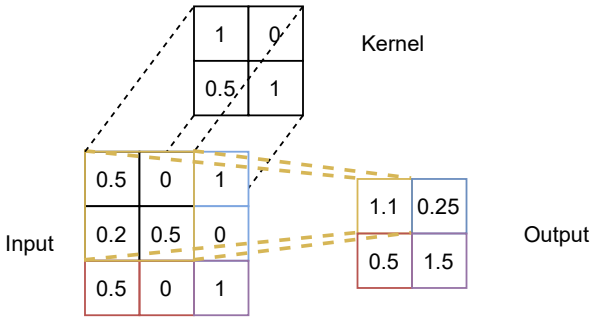


Figure 2.4: Illustration how CNNs work

The CNN architecture are actively used in our framework on SIFs. For further details see section 4.2.

FORWARD PASS is the process of feeding one input vector as the input layer's activation into the network, followed by passing these activations on the next layers, until reaching the output layer. Each unit has a pre-activation s_i which is the sum of all incoming activations a_i times the weight w_{ij} associated with the connection.

Formally:

$$s_i = \sum_{j=0}^{Q} w_{ij} \cdot a_j \tag{2.3}$$

Where Q is the total number of units.

The unit's activation is then computed by applying a non-linear activation function f to the pre-activation:

$$a_i = f(s_i) \tag{2.4}$$

The activations of the output layer correspond to the network's final output y.

BACK-PROPAGATION is how the network learns from annotation labels for each input vector. First, the network's final output is compared to the expected value (true label) by a loss function. Then, from this loss, the gradients are computed, and each contributing weight is updated by a small part of its gradient. This is referred to as "Stochastic Gradient Descend" (SGD). As this is out of the scope of this paper, we will not go into detail about these calculations.

2.2.2 *Non-Hybrid models*

Non-Hybrid models use only one layer type, as opposed to combining complementary types into one ensemble. For example only convolutional layers or only recurrent layers, not both.

Cakir et al. [4] suggested in their work to use so-call Maxout Networks [9] for polyphonic SED. These are basically Fully-connected Neural Networks (FNNs) using the maxout activation function. The idea is to use dropout and an activation function, which is not bounded, easy to optimize and does not suffer from the vanishing gradient problem. The formula is as followings:

$$f(x) = max(x \cdot W + b) \tag{2.5}$$

x is the input data, W corresponds to the learned weight matrix, and b is the additive bias factor, where x is $\in \mathbb{R}^d$, $W \in \mathbb{R}^{d \times m \times k}$ and $b \in \mathbb{R}^{m \times k}$ respectively [9]. The use of maxout networks can speed up training due to the max-pooling unit and has become a popular technique to improve model training speed. According to [6], the max-function only propagates the gradients to the unit with the largest activation and can therefore be prone to over-fitting. Maxout networks also require twice the amount of training parameters per unit, compared to traditional FNNs.

In a later work, Cakir et al. [3] investigated the idea of using binary single-class classifiers instead of a multi-class system. This allows for a dynamic extension and combination of classifiers within the system, without discarding and re-training the whole model. For each sound class, a single classifier instance is trained and saved. To support further, possible, concurrent classes, we simply have to train another instance on only the new class. They concluded that single-class classifiers are almost equally well-performing but slightly inferior to multi-class classifiers. This might be due to the loss of context information gained from a system with a wider perspective

and shared weighting. Also, the training time and hyper-parameter search effort might be folds higher for separate single-class identifiers.

An illustration of how this was implemented is shown in Figure 2.5:

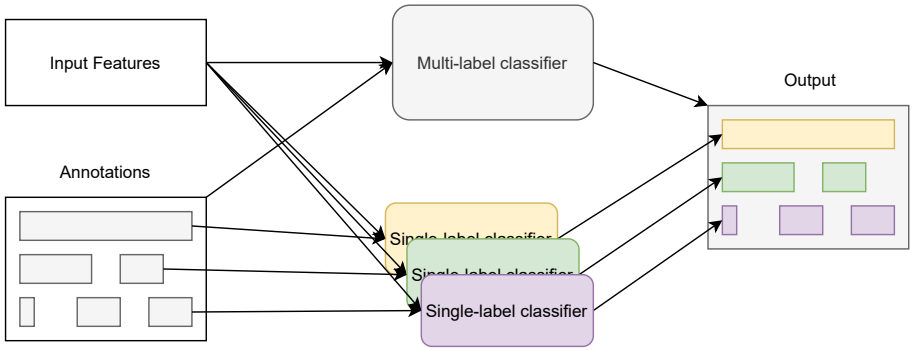


Figure 2.5: Difference between combined single-class and multi-class classification methods (inspired by Fig. 1 from [3]).

Parascandolo et al. [14] attempted to use Bidirectional Long Short-Term Memory ((B-)LSTM) networks for SED. As all Recurrent Neural Networks (RNNs), LSTMs combine the current input with the previous output of the same layer by addition. Bidirectional ones allow processing data in both directions by utilizing two separate hidden layers. The main advantage of RNNs is that they acquire a larger time-dependent context of information, incorporating each previous step into the final output. This can be very beneficial for SED as sound events can spread over a long time, thus affirming the same event's likelihood still being present in the new time step. The main disadvantages of LSTMs are that it is hard to find ideal hyper-parameters and that they prohibit parallelization.

Xia et al. [20] proposed to use Convolutional Neural Networks (CNNs) combined with a confidence based regression function and parabolic post-processing. The parabola spans from the onset till the offset of every event. The closer the current frame is to the center of the manually labeled acoustic event, the higher the confidence will be. Thus they determine the likelihood of an event actually being present by analysing the parabola's peak strength. This can mitigate wrongly annotated event times by an error-prone human annotator.

Many other promising approaches to non-hybrid SED exist, each with its own benefits and disadvantages. Combining the best attributes of each can lead to especially good model performance.

2.2.3 *Hybrid Models*

Hybrid Models are ensembles of different network types, making use of the advantages of each contained type, while mitigating their respective caveats. Most state-of-the-art systems nowadays are based on such hybrid models.

One of the most popular and commonly used one for SED, is the Convolutional Recurrent Neural Network (CRNN/RCNN). It combines the long-term time-context information gained from RNNs with the abstract representation layers and pattern recognition ability in the frequency domain from CNNs, to achieve a combined, supplementary perspective on input data.

Figure 2.6 shows the simple approach to CRNNs by stacking CNN layers on top of RNNs, feeding the feature maps from the convolutional layers to the recurrent blocks, followed by an FNN output layer.

Figure 2.6: Flowchart of a CRNN model

In recent years, CRNNs were the most popular architectures for almost all SED tasks in the annual DCASE challenges. For example, in task 3 of the 2020 DCASE challenge about "Sound Event Localization and Detection", which is a combination of SED with DOA estimation, Adavanne et al. [2] provided the baseline-system using a CRNN. Also, nine out of the final ten best-performing systems for this task were RCNNs. In their work, Adavanne et al. also compare their SELDnet to other architecture types, only with respect to SED metrics, and show that CRNNs are performing significantly better than non-hybrid models.

As described in Chan et al. [6], many more combinations of network types for SED training were suggested by various authors. These include Gated Recurrent Units (GRUs), LSTMs, CBLSTMs, GMMs, and Auxiliary Classifier Generative Adversarial Network (AC-GAN). Most of these are either special types of CRNNs or less commonly used approaches. We will not go into detail here as it is less relevant for the context of this paper.

2.2.4 *Models For Weakly Labeled Data*

All above-mentioned concepts have relied on strongly annotated datasets. This means that they contain annotations including onset and offset times for each event. Such datasets are sparse and usually small

in size. However, there exist a lot of weakly labeled sets, meaning that they are annotated with occurring sounds inside the recording but without exact time positions. These types of data require slightly different approaches and tricks to train models for detecting them effectively.

Chan et al. [6] state that such challenges are usually approached by using an ensemble of two different models where one is focused on detecting the right label for the recording, while the other one finds the onset-offset bounds. In total, previous research has shown that weakly labeled systems are not on par yet with human-annotated strongly labeled counterparts.

3 AUDIO FEATURES

The effectiveness of training highly depends on the dataset, the chosen architecture, and the type of extracted features. Depending on the task, purpose, and available data, many different kinds of features have been chosen for SED. By far the most common, popular, and proven to be effective feature type is logarithmic Mel-band scaled energies. Other popular ones are Mel-Frequency Cepstral Coefficients (MFCC), raw spectrogramm images and Generalized Cross Correlation with phase transform (GCC-PHAT). Simple features like raw waveform amplitudes are rarely used and less effective.

3.1 RAW AUDIO

In nature all sound is analogue and continuous. For storing a representative form of audio signals digitally, the signal must be sampled in constant, finite time intervals. The result is a sequence of numbers which correspond to the waveform's amplitudes at these discrete time points. This process is displayed in figure 3.1:

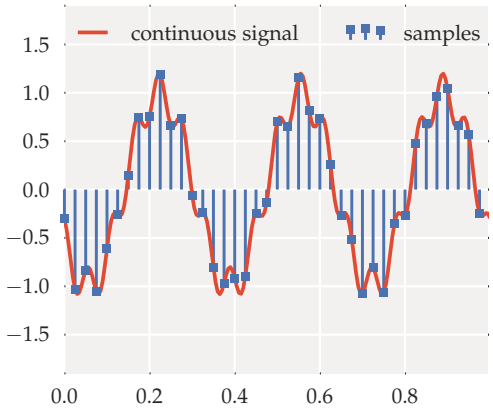


Figure 3.1: Waveform sampling (Figure 3.1a from [16], used with permission)

This "raw" sequence of amplitudes is the basis for the popular `.wav` audio files, used in many datasets. We could use this directly for SED, but it has many limitations. Some of them are:

1. (Depending on the sample rate) the amount of data points per second is huge and would require a large network with many parameters. Looking at only a small portion of samples would give too little information to infer which sound events occured.

2. It is hard to digest for our network. SIFs already contain useful pre-processesed information for deriving the sound class and present this in an interpretable format. With raw audio, the system first has to learn how to extract these essential details from the waveform sequence.

3. Amplitudes themselves give little indication about the difference between sound events and noise.

In practice, raw audio data is not used for SED directly, but is the basis for all kinds of other feature representations that serve as fitting input feature types.

3.2 SPECTROGRAMS

After parsing and sampling raw audio signals into a discrete, finite waveform amplitude sequence, the windowed Short-Term Fourier Transform (STFT) is applied, using usually a Hamming window and 50% overlap, resulting in the linear frequency spectrum.

As described in section 3 of Schlüter [16], the STFT is basically applying the Discrete Fourier Transformation (DFT) on overlapping excerpts of the signal and summing up the results. We take an excerpt of length T from the waveform sequence and multiply the time-domain signal with all cosine and sine functions, with a whole number of oscillations, over this excerpt. Summing up these products over time results in the so-called *spectrum*. Due to the *Nyquist theorem* we receive only half the amount of bins of the sequence length, because any oscillation faster than the *Nyquist rate* would again be interpreted as a lower frequency when sampled.
The formula is as shown in equation 3.1 (taken from [16]):

$$X_{t,k} = \sum_{n=0}^{T-1} s_{Ht+n}\omega_n \left(\cos(\frac{n}{T}2\pi k) - i\ \sin(\frac{n}{T}2\pi k)\right), \quad k \in (0,1,...T/2) \tag{3.1}$$

Where each time-window frame $X_{t,k}$ gives magnitudes and phases of sinusoids with frequency k/T, starting at position Ht for each excerpt. ω is the window frame (vector of length T).

An example of a two seconds excerpt in a residential environment, where a bird is singing, results in the spectrum displayed in figure 3.2. The frame length was chosen as 1024 and the hop size as 512. This results in 513 (T/2+1) spectral bins.
On the y-axis are the frequency bins, on the x-axis the time frames. The absolute magnitudes (in dB) of the time-frequency bins are shown

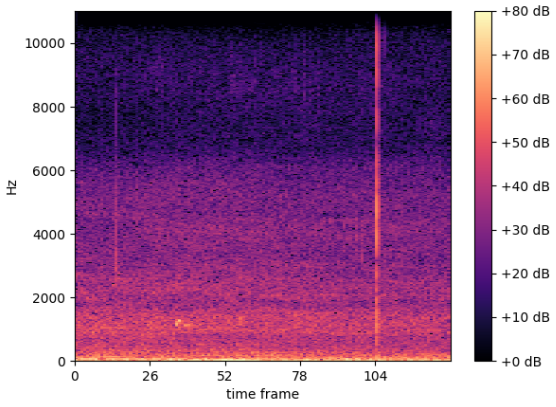


Figure 3.2: spectrogram (generated with `librosa` python library)

in the color channel. The bright line through all frequencies at time frame 104 is an indicator of a new sound event onset, due to large magnitudes across most of the frequency range.

This example emphasizes how useful such spectral images can be for training a CNN to detect events and using the adjacent context information about frequencies and magnitudes, to determine the event's class.

Zhang et al. [21] proposed spectrogram image features (SIFs) that can be used effectively in combination with Convolutional Neural Networks (CNNs). Sources of different sound class usually produce audio in different frequencies. Looking at images of the frequency spectrum can give strong indications about what combination of sounds make up the spectrum, by analysing the patterns of the SIFs. They point out how in image processing CNNs are especially often used for this purpose.

3.3 LOG MEL ENERGIES

Logarithmic Mel energies currently count as one of the most used types of features for audio and music processing. Its popularity and well performing models in ML challenges can be explained by the perceptual weighting done on the audio data, which correlates to the annotations usually done by human annotators.

PERCEPTUAL WEIGHTING is then applied to the powers of the spectrum by mapping them to the Mel scale. By using a filter bank of commonly 40 unequally distributed triangular windows [13] on the spectrum, the signal gets divided into bins which are non-linearly laid

out over the frequency dimension. The formula for this transformation is shown in Equation 3.2:

$$m = 2595 \cdot log_{10}(1 + \frac{f}{700}) \tag{3.2}$$

The result corresponds to roughly human-like perception by approximating the response of the human auditory system to frequency changes, as opposed to the linearly-spaced frequency bands used in the usual spectrogram. The equation was derived by Stevens et al. [18] in 1937 by empirical studies. Subjects were asked to judge the perceived distance between pitch levels. The results then were then averaged and used to derive the numerical constants in the formula. For high frequencies, humans perceive changes in frequency as less salient than for low ones.

Figure 3.3 and figure 3.4 from Jan Schlüter [16], visualize how the Mel scale correlates to increasing frequency and how the applied triangular windows are distributed.

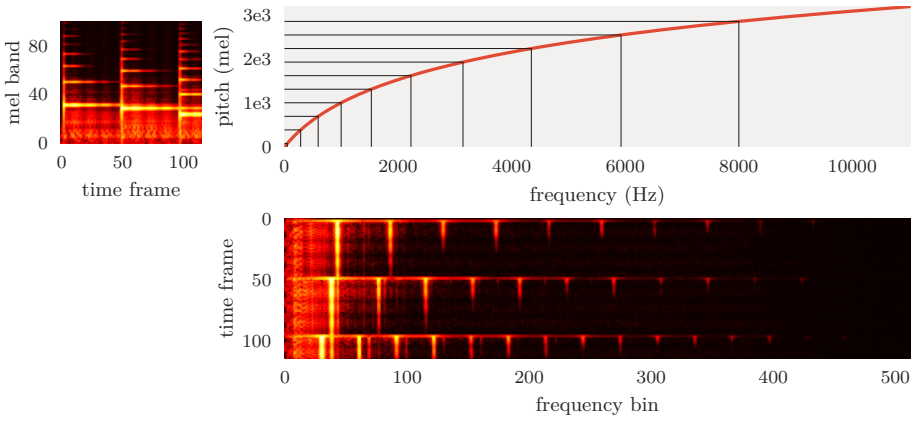


Figure 3.3: The Mel spectrogram (Figure 3.5 from [16], used with permission)

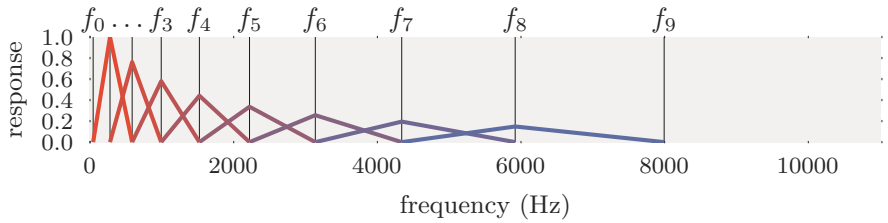


Figure 3.4: The Mel filterbank (Figure 3.6 from [16], used with permission)

3.4 MEL-FREQUENCY CEPSTRAL COEFFICIENTS

MFCCs are based on these perceptually weighted inputs and further apply the Discrete Cosine Transformation (DCT) on the list of log Mel powers. The amplitudes of the resulting spectrum are the MFCC.

The most prominent oth coefficient (first and top-left-most one) is often removed from the result data to improve neural network training by eliminating an extreme outlier [1], thus controlling gradient strength and preserving more meaningful feature values after input normalization.

According to Schlüter [16] the DCT is similar to the DFT for spectrograms, but it's using only cosines as templates and frequencies of k and $k + 0.5$ cycles to still form a complete basis. The coefficients provide a description of the spectrum in terms of periodicities over the frequency bins. He further writes that it can be seen as a decorrelation and compression of the spectrum. Nowadays it's another popular input feature type for SED.

4 EXPERIMENTS

In the context of this thesis, a framework was implemented to evaluate the effectiveness of different input feature types for multi-label prediction of polyphonic SED, with strongly labeled data. We further compared multi-scene versus single-scene performance. As an architecture, a CNN was chosen. The task also required a fitting dataset with strong annotations.

4.1 STRONGLY ANNOTATED DATASET FOR SED

There are many publicly available datasets for SED, but only few of them come with full human-annotated labels. Chan et al. [6] mentioned a few datasets for SED in their work. For the full list see appendix-section A.1.

The chosen dataset was *TUT-SED 2016*. It contains:

- strong labels
- real-life recordings
- two different sound scenes
- 18 sound classes, distinct by scene

The dataset was used as a baseline in the DCASE2016 challenge. According to Mesaros et al. [13], it features a total of 18 sound classes, divided into two very different sound scenes. For the challenge, the recorded data were split 70/30 into development set[1] and evaluation set[2]. The development set's "home" recordings consist of 11 different sound classes totaling about 36 minutes, while the "residential area" recordings sum up to a length of 42 minutes and contain only 7 distinct sound classes. In the evaluation split, another 18 minutes of each scene is available. The equipment used for capturing the audio data was a binaural Soundman OKM II Klassik/studio A3 electret in-ear microphones and Roland Edirol R09 wave recorder using 44.1 kHz sampling rate and 24-bit resolution.

The annotated sound classes used for the DCASE task back in 2016 can be found in Table 4.1.

These classes and scenes are the basis for our framework. The model architecture is built dynamically, to predict different amounts of classes based on the chosen scene.

1 https://zenodo.org/record/45759
2 https://zenodo.org/record/996424

Residential area		**Home**	
event class	instances	event class	instances
(object) banging	23	(object) rustling	60
bird singing	271	(object) snapping	57
car passing by	108	cupboard	40
children shouting	31	cutlery	76
people speaking	52	dishes	151
people walking	44	drawer	51
wind blowing	30	glass jingling	36
		object impact	250
		people walking	54
		washing dishes	84
		water tap running	47

Table 4.1: TUT-SED 2016 dataset sound classes and their number of instances [13]

For comparing which class of input features fits best for given data and would achieve the highest scores, we chose to support these three types for our framework:

- Logarithmic STFT-spectrograms
- Log Mel energies
- MFCC

The framework was implemented in Python using `Python 3.8.6` and `PyTorch 1.6.0`. The metrics implementation from the DCASE2016 baseline system[3] was integrated into our framework for comparable scores. Many additional python packages were used to serve special needs in processing operations and visualizing data efficiently.

4.2 ARCHITECTURE

The system's architecture uses a simple non-hybrid convolutional neural network with few layers, followed by separate fully connected output layers for each possible event class. The architecture was kept rather simple to get fast results, comparing which combinations of input representation are especially effective for which scene and how post-processing steps improve our final predictions.

It is reasonable that this comparison should also scale to larger networks considering that:

3 https://github.com/TUT-ARG/DCASE2016-baseline-system-python/blob/master/src/evaluation.py

- additional layers and parameters will just push into over-fitting due to the small dataset
- the difference in effectiveness between individual feature types (e.g. their class-correlation) will not change by the depth of the network
- Our pre-, and post-processing techniques used are independent of the network complexity

A sketch of the system architecture is shown in figure 4.1:

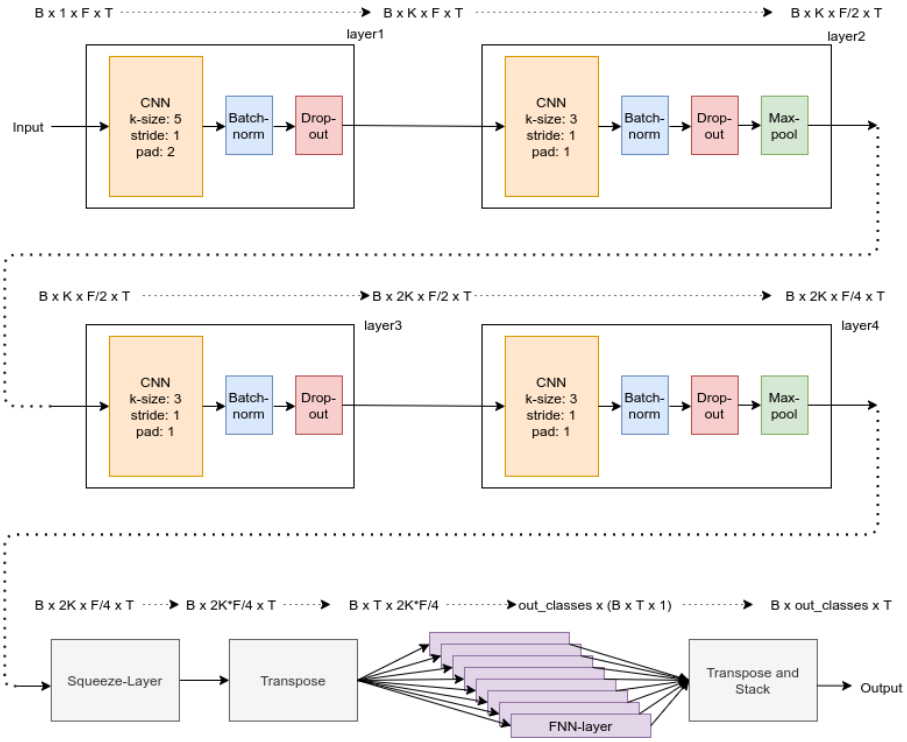


Figure 4.1: The network architecture

The input of our network has a size of (B, 1, F, T). B stands for the batch size, meaning how many samples are fed to the network at the same time. 1 (or later K) represents the number of input channels, defined by the number of kernels used in the previous layer. F is the feature dimension size, which is depending on the type of input features used. T is the number of time-steps (or amount of frames in one excerpt). The dataset consists of recordings with a duration between three and five minutes. Because these recordings are too large to input into the network whole, input data is fed to the network in excerpts of configurable length.

In the first layer (dubbed "layer1"), the channel size dimension of the input tensor increases by the number of kernels specified. In the second layer, the feature dimension is reduced in size by half due to the (1, 2) sized max-pooling layer. The third layer doubles the number

of channels again, and the fourth reduces the feature dimension by another half. The output of these four CNN layers is then put into a "squeeze layer" which combines the K and F axes into a single one, leaving us with a three-dimensional tensor. The transposed data are then fed to separate FNNs, one for each of the sound classes, resulting in one-dimensional, binary vectors of predictions for each sound class. All vectors are then merged together to a single tensor of size (B, <number of classes to predict>, T), which is the total prediction for all excerpts in the mini-batch, each shaped (<amount of classes> x <excerpt-length>). The batch and time dimensions are never reduced. As a non-linear activation function, we added the commonly used Rectified Linear Unit (ReLU) after each CNN block.

The binary-cross-entropy loss function combined with the Adam optimizer is used for estimating loss and performing stochastic gradient descent (SGD) to optimize our model. The default window size is 512 frames; the sampling rate is extracted from the audio files by `librosa`, the learning rate is 1^{-5} and weight decay is 1^{-6}. Other noteworthy default hyper-parameters are excerpt_size=384, batch_size=16, and an excerpt overlap factor of 4 (meaning the inter-excerpt distance is actually 384/4 during training). The standard dropout rate is set to 30%, and the number of initial kernels for the CNN is 64.

Figure 4.2 shows an example plot of an excerpt of a residential area recording with a length of 512 frames, illustrating the targets gathered from provided annotations, the network predictions, and the error resulting from taking the difference of both.

We observe that there is still much noise and small gaps between predictions or insignificant lengths for detected active events. Reducing these errors might lead to a better score, which leads us to the framework's post-processing methods.

4.3 POST-PROCESSING

We applied several post-processing steps to the network prediction output. Some of which are taken from related literature, one of them from the DCASE2016 baseline system.

Cakir et al.[4] suggested to use a 10-frame windowing function to smoothen the output signal and reduce noise. We adapted this method in our implementation. For each frame, the post-processed output is obtained by taking the median of the binary outputs in a 10-frame window as

$$\hat{z}_t = \begin{cases} 1, & \text{if median}(z_{(t-9):t}) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

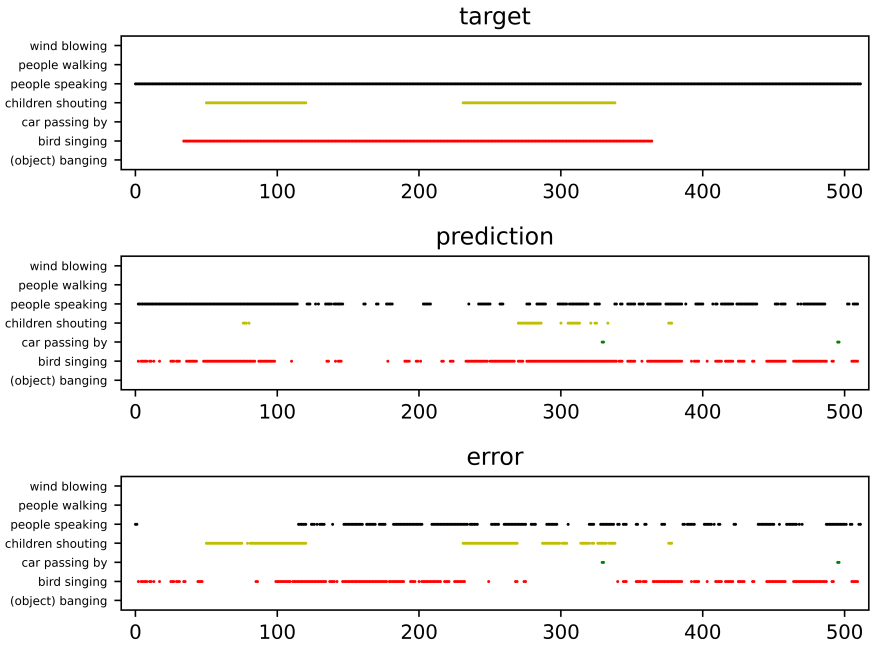


Figure 4.2: An example plot showing how targets, predictions and resulting error looks like.

This led to non-significant improvements according to the metrics calculated, so further methods where investigated.

Normalizing the event segments to a minimum event length and a minimum gap between events, as implemented in the baseline by Mesarosa et al. [13], has brought upon the biggest improvement in metric scores. First segments of length smaller than 0.1 seconds are removed, followed by merging events with a gap of fewer than 0.1 seconds between them. The snippet below shows how this looks in python code.

Listing 4.1: post-processing of segments inspired by [13]

```
def normalize_segments(array: np.ndarray, min_event_len=0.1, min_gap=0.1,
      hop_size=512, sr=22050) -> np.ndarray:
    threshold = 0.5
    min_length_indices = int(min_event_len * sr / hop_size)
    min_gap_indices = int(min_gap * sr / hop_size)

    def remove_short_segments(bool_array, min_seg_length, replacement):
        # Find the changes in the bool_array
        change_indices = np.diff(bool_array).nonzero()[0]
        # Shift change_index by one, focus on frame after the change
        change_indices += 1
        if bool_array[0]:
            # If first element of bool_array is True add 0 at beginning
            change_indices = np.r_[0, change_indices]
        if bool_array[-1]:
```

```
            # If last element of bool_array is True, add array-length
            change_indices = np.r_[change_indices, bool_array.size]
        # Reshape the result into two columns
        segments = change_indices.reshape((-1, 2))

        # remove short segments
        for seg in segments:
            if seg[1] - seg[0] < min_seg_length:
                array[b, c, seg[0]:seg[1] + 1] = replacement

    for b, batch in enumerate(array):
        for c, class_values in enumerate(batch):
            # no changes if no events predicted
            if np.all(class_values == 0):
                continue
            event_segments = class_values >= threshold
            remove_short_segments(event_segments, min_length_indices, 0)
            gap_segments = class_values < threshold
            remove_short_segments(gap_segments, min_gap_indices, 1)

    return array
```

One more idea for post-processing was gained from [19] and [8]. By using an additional "silence" (or "background" in our framework) class, we could try to let the network predict the non-existence of any active sound source and as following remove all other detected events at this frame instance, when a certain threshold for "silence" was detected. This method brought upon another minor bump in metric scores.

4.4 DATA AUGMENTATION

Lastly, we would like to mention some data augmentation techniques used in our framework to artificially enlarge our rather small dataset and improve generalization.

RANDOM GAUSSIAN NOISE is one of the most known and popular techniques to add some variation to the input data. In our case, it adds random values to the spectrum using a Gaussian distribution of 0.1 variance and zero mean. This causes the exact same spectrum image excerpt to always differ slightly, without changing the data too much and teaching the network to generalize better.

Gorin et al. [10] used the same augmentation technique in their CNN based approach for the same DCASE task.

RANDOM SPECTRUM STRETCHING is another method to introduce altered excerpts to the network, which are transformed in their pitch intensity by $\pm$ 30%. We used `scipy` and its `affine_transform` transformation for this augmentation.

RANDOM DB ALTERATION was also applied to each excerpt to add up to ± 30% decibel to the whole spectrum.

FILTERING is the final applied augmentation. We choose a random part of the excerpt and add or remove another ± 10 decibel in only this section.

In total, all these augmentations brought upon a significant increase in scores and made training more stable. Parameters were found out by experimenting with values and comparing results.
For details about the effectiveness of our data augmentation methods we refer to section 5 and table 5.2.

For implementation details and instructions on how to use the framework see the appendix-section A.2.
Tensorboard was used to tune parameters and log results. For more details about it's usage see appendix-section A.3.

5 RESULTS & DISCUSSION

5.1 RESULTS

After training on each scene with all different feature types, each for about ten thousand mini-batch updates, we received some insight into the success of every type of combination. The scores after post-processing are also evaluated and logged separately. The main scoring metric used in the baseline is segment-based F1-score and Error Rate (ER) on the evaluation set. Mesaros et al. [13] write, that F1-score is based on the total count of true positives (TP), false positives (FP) and false negatives (FN). From these, precision, recall and F-score are calculated as following:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F = \frac{2 \cdot P \cdot R}{P + R} \tag{5.1}$$

Error rate is calculated similarly as the Levenshtein distance. We sum up all Substitutions (S), Insertions (I) and Deletions (D) per 1-second segment needed, to get to the ground-truth annotation. The equation for ER is defined as:

$$ER = \frac{\sum_{k=1}^{K} S(k) + \sum_{k=1}^{K} D(k) + \sum_{k}^{K} = 1I(k)}{\sum_{k=1}^{K} N(k)} \tag{5.2}$$

with K being the total amount of segments, and $N(k)$ the number of active ground truth events in segment k. Substitution means the system predicted an event but gave it the wrong label, deletion that an event was recognized though there was none, and insertion means that a ground truth event wasn't detected at all.

Table 5.1 shows the metrics gathered from training according to different setups. Each row corresponds to a different scene type.

	Baseline		Spectrum		Spectrum (pp)		Mel energies		Mel energies (pp)		MFCC		MFCC (pp)	
Scene	**ER**	**F [%]**	**ER**	**F [%]**	**ER**	**F [%]**	**ER**	**F [%]**	**ER**	**F [%]**	**ER**	**F [%]**	**ER**	**F [%]**
Indoor	0.95	18.1	1.32	14.6	1.07	6.4	1.31	21.8	1.04	6.8	1.21	8.3	1.01	0.8
Outdoor	0.83	35.2	1.06	49.7	0.91	48.3	1.03	51.9	0.83	54.4	1.21	48.6	0.87	52.2
Avg single scene	0.89	26.65	1.19	32.15	0.99	27.35	1.17	36.85	0.935	30.6	1.21	28.45	0.94	26.5
Multi-scene			1.21	22.8	1	2.8	1.35	30.2	1.01	29.5	1.17	18.3	1.01	2.6

Table 5.1: Metrics gained from training runs. "Indoor" are the metrics for the dataset's home scene, "Outdoor" for the residential area scene. "Avg single scene" corresponds to $(Indoor + Outdoor)/2$. "Multi-scene" is the classifier that predicts events for both scenes simultaneously. "(pp)" stands for post-processed scores (all mentioned methods applied).

Metrics were gathered by training our network first for each scene separately, then in the combined mode where all labels from both scenes are predicted on every recording's excerpt. For each type of scene we trained three different networks. One for every input feature type. This resulted in a total of 9 models. As post-processing can be dynamically added, we gain both original and post-processed score in one training run. Baseline scores were taken from Mesaros et al. [13].

For investigating the effectiveness of our data augmentation techniques, two additional runs were made with Mel energies (our best feature type) but without using data augmentation.

Table 5.2 shows the difference between each scene's results without data augmentation compared to its counterpart using these techniques.

	ER (no DA)	F (no DA) [%]	ER (with DA)	F (with DA) [%]
Indoor	1.51	15.4	1.31	21.8
Outdoor	1.05	51.4	1.03	51.9
Avg single scene	1.28	33.4	1.17	36.85
Multi-scene	1.37	29.2	1.35	30.2

Table 5.2: Comparison of metrics for *Mel energies* after using data augmentation and those without using it. As in table 5.1 "Avg single scene" is again the average value of Indoor and Outdoor metrics, while "Multi-scene" corresponds to the model predicting events in both scenes. "DA" stands for data augmentation.

5.2 FINDINGS

Analysing the data in table 5.1 has shown that our simple approach to the task with only 4 CNN blocks combined with FNNs for each label class has given scores that come very close to the GMM baseline system [13]. Back in 2016, only one system has actually beaten baseline [1], so we were satisfied with the results.

1 http://dcase.community/challenge2016/task-sound-event-detection-in-real-life-audio-results

While in our results, F1-score was usually higher than in baseline metrics, error rate scores were slightly worse in comparison.

The applied post-processing seemed to have a very positive effect on the ER metric. In terms of F-score, only outdoor results have improved or stayed similarly good, while in the indoor scene, it actually had a significant negative effect. For example, in the Mel energies section, the indoor F-score has decreased from 21.8 to a mere 6.8, but for the outdoor scene, it actually increased from 51.9 to 54. We explain this by how events behave in the recordings from the indoor scene. They are usually sound bursts of very short length, like closing a cupboard or opening a drawer, while the outdoor scene features longer-lasting events like a bird singing or people walking. The baseline's post-processing of removing short segments could easily remove such if short enough. Also, the median filter suggested by Cakir et al. [4] or the "silence" class from [8, 19] could similarly eliminate such events accidentally. So for the maximum score in combined single-scene detection, we should take the non-post-processed output from indoor combined with the post-processed one from outdoor. Another viable option would be to explore different post-processing methods, used only, or especially for the indoor scene.

Further, we observe a trend in input feature performance. We compared feature classes according to pure performance, without post-processing, for not incorporating bad effects for the indoor scene into our judgment. Log Mel energies seem to be the most effective ones, followed by logarithmic spectrogram images, and MFCC slightly behind in the last place. For combined single scene mode, the non-post-processed results with log Mel energies, compared to the spectrum images, perform about 15% better in average F-score and 2% better in the ER department. The MFCC approach in our system, compared to Mel energies, shows to be about 29.5% worse in terms of F-score and 3.5% in the ER metric. This leads us to two conclusions. That perceptual weighting indeed is a superior way to approach sound-based tasks with human annotators and that MFCC *does* lead to a slight loss of information, as stated in the work of Chan et al. [6] and Cakir et al. [4], because it performs worse than raw logarithmic spectrograms.

The data gathered also show that combined single-scene SED seems to be superior to multi-scene detection in both ER and F1-score. For the log Mel energies input feature, the difference is 22%, for MFCC 55%, and for log spectrograms 41%. This might be due to the separate models, each specializing in their respective acoustic environment and adapting better to ignoring specific background noises and unannotated sound sources. Also, limiting one model's amount of classes to detect, leads to a lower false prediction rate.

Table 5.2 shows how data augmentation clearly had a positive effect on the final system metrics. All scores have improved by using these

techniques. Especially in the indoor scenes, we observe a significant bump in F-score and a decrease in error rates. This is probably due to the scarcity and short duration of these events in the data. Data augmentation can help here, for the network to see more instances of each event.

5.3 RESEARCH QUESTIONS

We can now answer the research questions from the introduction section 1, created before implementation of the framework, to some degree. Based on our findings, we conclude the following:

1. How can (CNN-based) Sound Event Detection (SED) systems based on the TUT-SED 2016 data set achieve high performance in terms of high F1-score and low error rate?
    - Combined-single scene classification using log Mel frequency features was the most successful variant in terms of overall F-score and ER.
    - The mentioned data augmentation techniques from section 4.4 helped further improve generalization to unseen data.
    - The baseline's segment-based and other post-processing methods have greatly reduced ER and further boosted F-score.
2. What are the advantages/disadvantages of treating SED as a combined single-scene classification problem compared to treating it as a multi-scene classification problem? How well do they perform in comparison?
    - By creating a separate network specialized for each scene and their distinct event classes, the overall system performance has increased.
    - Our research has shown that multi-scene can be between 22% and 55% more effective, based on the feature type used.
    - Training and using two (or more) networks requires twice the amount of training time, parameters, and memory/CPU resources.
    - The system can be extended easily by an additional sound scene by training only one more sub-network. In contrast, for the multi-scene classifier, we would have to re-train the whole system and discard the previous model.
    - New (unknown) audio samples need an indication of which network to use for prediction. Either supervised or an additional network that predicts just the sound scene.

3. What are the most effective post-processing steps for SED systems to achieve high performance in SED?
    - Normalizing the event segments to a minimum event length and a minimum gap between events, as implemented in the baseline by Mesarosa et al. [13], turned out the be the most effective method. This is partly because the evaluated metrics stem from the baseline itself.
    - The 10-frame windowing approach to smoothen the output, suggested by [4], did not have a significantly beneficial effect according to the metrics.
    - Adding a "silence" class and attempting to filter out false positives during silent periods (suggested by [8, 19]), has also removed many true positives and mostly canceled out its positive effects. This is especially true for the indoor scene with mostly short sound events.
4. Which data augmentation techniques significantly improve generalization in SED?
    - All the applied methods combined brought a positive increase in the final performance. Especially random spectrum stretching and adding a random amount of dB seemed to be effective.
5. Does perceptual weighting, in particular, logarithmic weighting and Mel-Band filtering, have a significant influence on system performance?
    - Yes. The log-Mel-frequency feature type has outperformed the log-frequency variant significantly.
6. In single-scene classification: what combination of pre-/post-processing is best suited for which particular sound scene?
    - The minimum event length and gap width approach was well suited for both sound scenes.
    - The windowing and silence-class approach have slightly helped in outdoor but had negative effects on the indoor scene.

6 CONCLUSION AND SUMMARY

In summary, we can say that the task of polyphonic SED is far from solved, and there is still much room for improvement. Machine learning algorithms have made it easier to achieve these tasks to some degree, but there is no one solution for all kinds of data. Also pre-, and post-processing methods can be explored further. For image and video processing there exist many approaches and algorithms to improve picture quality, but in the audio domains there is still much to discover and work to be done. Especially polyphony is making the task hard. While we are already able to easily detect multiple objects in one picture, detecting and classifying multiple sound sources is still hard.

In general, our CNN based approach was quite successful, and our framework makes for a good basis for further hybrid variants of models based on it. Especially incorporating recurrent neural networks into our existing system would probably greatly improve performance. In terms of input features, we conclude that log Mel energies is still one of the best types and confirmed one more time, that it is popular for a reason.

The relatively low scores from both baseline and our implementation goes back to the dataset being just too small to effectively train for each individual sound class.

6.1 FUTURE WORK

As a next step, we would look into improving our network and create a deeper architecture or compare different model classes. For example, creating a CRNN instead of a simple CNN, would probably, greatly improve the metrics. We also would like to explore more pre-/post processing methods like filtering out noise before the STFT or the parabolic confidence based post-processing from [20]. Especially the post-processing for indoor recordings needs further work. In terms of data augmentation, there is also more work to be done. Parascandolo et al. [14] suggested to use block mixing to increase the size of the limited dataset by combining recordings and filling out silent parts using data from another file.

Adding a Cross Validation (CV) implementation to the framework might also further improve the estimation of the performance. It would allow us to use all sound files for both training and validation instead of splitting off one-fourth for the validation part.

Another variant we would like to look into for improving the system's performance is to add additional context information for each position in the input layer and feed them to the network as additional channels for each sample. The idea is, that the more (diverse) information given, the better the quality of the prediction would be.

Lastly, we would like to test our framework on further datasets to evaluate how well our implementation works on different kinds of data.

APPENDIX

A.1 DATASETS FOR SED

The implemented framework required a strongly annotated dataset. Chan et al. [6] mentioned a few datasets for SED. Their most relevant attributes were:

- TUT-SED 2016
  - strong labels
  - real-life recordings
  - two different sound scenes
  - 18 sound classes, distinct by scene
- TUT-SED 2017
  - strong labels
  - real-life recordings
  - two sound scenes
  - 7 common sound classes for both scenes
- DCASE 2017 - weak labels
- DCASE 2018 - weak labels

The basis of our framework implementation was the *TUT-SED 2016* dataset.

A.2 IMPLEMENTATION DETAILS

The framework is called by invoking python on the `main.py` file. The three additional parameters required are: <feature type>, <scene> <config>.

- feature type - the input feature type to be used. Has to be one of:
  - mels - for log Mel energies
  - mfcc - for Mel frequency cepstral coefficients
  - spec - for logarithmic weighted spectrograms
- scene - the part of dataset used, depending on the acoustic environment it was recorded in:
  - indoor - the "home" part of the dataset

  - outdoor - the "residential area" part of the dataset
  - all - the combination of both "home" and "residentail area" (multi-scene)

- config - a json-file containing common hyper parameters for the STFT, the network, logging, excerpt length, feature size, etc.

If non-existent on the hard drive, the TUT-SED 2016 datasets will be downloaded automatically. As a next step, STFT spectrograms of all recordings in each scene, as well as the respective target arrays according to the annotations provided, will be pre-computed and saved in a `pickle` file. Then, in a background thread, excerpts of those spectrograms will be prepared, applying all the previously mentioned random data augmentation methods.

3/4 of the development set was used as our training set, and 1/4 was used as a validation set. Splits were done based on the information provided in the dataset's evaluation-setup. The currently known best network model is only updated if the new validation loss is less than the previous one.

After training is finished, the final evaluation is initiated. All data from the development set and evaluation set are fed to the network. Metrics are calculated on file level as well as dataset level. Additionally, targets, predictions, and errors are plotted for each file separately. The results are stored on disk and automatically zipped after each run.

A.3 TENSORBOARD

We used the `tensorboard` python module to tune our parameters. Losses, gradients, and metrics during training are logged in tensorboard, combined with the hyper-parameters used. This allowed us to visualize the effects of different parameter values in combination with the resulting score. Using the "parallel coordinates view", we were able to manually inspect the result of different parameter combinations and optimize their performance.

A sample of the parallel coordinates view after a small parameter sweep, is shown in figure A.1:

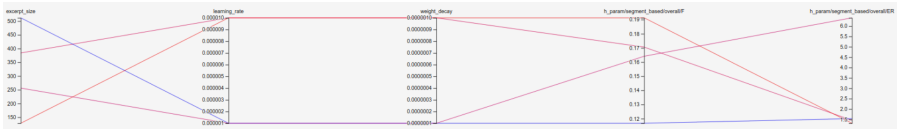


Figure A.1: Tensorboard's parallel coordinates view

The corresponding values for the parallel coordinates view are also provided in a readable format in table A.1:

The tensorboard interactive interface then allows inspection of each run by mouse-over and detailed tooltips.

excerpt_size	learning_rate	weight_decay	segment_based/F	segment_based/ER
384	$1 \cdot 10^{-5}$	$1 \cdot 10^{-6}$	0.171	1.46
512	$1 \cdot 10^{-6}$	$1 \cdot 10^{-7}$	0.117	1.54
256	$1 \cdot 10^{-6}$	$1 \cdot 10^{-7}$	0.164	6.4
128	$1 \cdot 10^{-5}$	$1 \cdot 10^{-6}$	0.191	1.31

Table A.1: Values of the parallel coordinates view

Figure A.2 shows examples of how scalars like F-score and error rate development from multiple runs, with different parameters, are displayed.

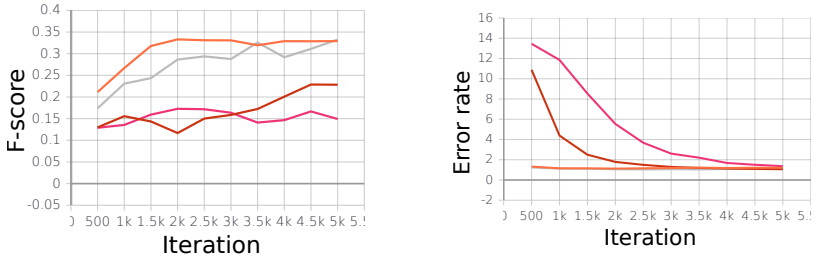


Figure A.2: Development of F-score and error rate scalars (4 different runs)

BIBLIOGRAPHY

[1] Sharath Adavanne, Giambattista Parascandolo, Pasi Pertilä, Toni Heittola, and Tuomas Virtanen. "Sound event detection in multichannel audio using spatial and harmonic features." In: *arXiv preprint arXiv:1706.02293* (2017).

[2] Sharath Adavanne, Archontis Politis, Joonas Nikunen, and Tuomas Virtanen. "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks." In: *IEEE Journal of Selected Topics in Signal Processing* 13.1 (2018), pp. 34–48.

[3] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. "Multi-label vs. combined single-label sound event detection with deep neural networks." In: *2015 23rd European signal processing conference (EUSIPCO)*. IEEE. 2015, pp. 2551–2555.

[4] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. "Polyphonic sound event detection using multi label deep neural networks." In: *2015 international joint conference on neural networks (IJCNN)*. IEEE. 2015, pp. 1–7.

[5] Léo Cances, Patrice Guyot, and Thomas Pellegrini. "Evaluation of post-processing algorithms for polyphonic sound event detection." In: *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2019, pp. 318–322.

[6] TK Chan and Cheng Siong Chin. "A Comprehensive Review of Polyphonic Sound Event Detection." In: *IEEE Access* 8 (2020), pp. 103339–103373.

[7] Jonathan Dennis, Huy Dat Tran, and Eng Siong Chng. "Overlapping sound event recognition using local spectrogram features and the generalised hough transform." In: *Pattern Recognition Letters* 34.9 (2013), pp. 1085–1093.

[8] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. "Audio set: An ontology and human-labeled dataset for audio events." In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 776–780.

[9] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. "Maxout networks." In: *International conference on machine learning*. 2013, pp. 1319–1327.

[10] Arseniy Gorin, Nurtas Makhazhanov, and Nickolay Shmyrev. "DCASE 2016 sound event detection system based on convolutional neural network." In: *IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events* (2016).

[11] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks." In: *Neural networks* 4.2 (1991), pp. 251–257.

[12] Daniel D Lee and H Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization." In: *Nature* 401.6755 (1999), pp. 788–791.

[13] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. "TUT database for acoustic scene classification and sound event detection." In: *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE. 2016, pp. 1128–1132.

[14] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. "Recurrent neural networks for polyphonic sound event detection in real life recordings." In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 6440–6444.

[15] Lawrence R Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition." In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[16] Jan Schlüter. "Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals." PhD thesis. Austria: Johannes Kepler University Linz, July 2017.

[17] Larry Squire, Darwin Berg, Floyd E Bloom, Sascha Du Lac, Anirvan Ghosh, and Nicholas C Spitzer. *Fundamental neuroscience*. Academic Press, 2012.

[18] Stanley Smith Stevens, John Volkmann, and Edwin B Newman. "A scale for the measurement of the psychological magnitude pitch." In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190.

[19] Qing Wang, Huaxin Wu, Zijun Jing, Feng Ma, Yi Fang, Yuxuan Wang, Tairan Chen, Jia Pan, Jun Du, and Chin-Hui Lee. *The USTC-iFlytek system for sound event localization and detection of DCASE2020 challenge*. Tech. rep. DCASE2020 Challenge, Tech. Rep, 2020.

[20] Xianjun Xia, Roberto Togneri, Ferdous Sohel, and David Huang. "Confidence based acoustic event detection." In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 306–310.

[21] Haomin Zhang, Ian McLoughlin, and Yan Song. "Robust sound event recognition using convolutional neural networks." In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 559–563.