Tampere University

EMRE CAKIR

# Deep Neural Networks for Sound Event Detection

Tampere University Dissertations 12

Tampere University Dissertations 12

EMRE CAKIR

# Deep Neural Networks for Sound Event Detection

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty Council of Computing and Electrical Engineering
of Tampere University of Technology,
for public discussion in the auditorium TB207
of the Tietotalo building, Korkeakoulunkatu 1, Tampere,
on 22.01.2019, at 12 o'clock.

ACADEMIC DISSERTATION
Tampere University, Faculty of Information Technology and Communication Sciences
Finland

| | | |
|---|---|---|
| *Responsible supervisor and Custos* | Tuomas Virtanen<br>Tampere University<br>Finland | |
| *Pre-examiners* | Dan Stowell<br>Queen Mary University of London<br>United Kingdom | Justin Salamon<br>New York University<br>United States |
| *Opponents* | Tomi Kinnunen<br>University of Eastern Finland<br>Finland | Dan Stowell<br>Queen Mary University of London<br>United Kingdom |

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

Copyright ©2019 author

Cover design: Roihu Inc.

ISBN 978-952-03-0961-9 (print)
ISBN 978-952-03-0962-6 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-0962-6

PunaMusta Oy – Yliopistopaino
Tampere 2019

# Abstract

The objective of this thesis is to develop novel classification and feature learning techniques for the task of sound event detection (SED) in real-world environments. Throughout their lives, humans experience a consistent learning process on how to assign meanings to sounds. Thanks to this, most of the humans can easily recognize the sound of a thunder, dog bark, door bell, bird singing etc. In this work, we aim to develop systems that can *automatically* detect the sound events commonly present in our daily lives. Such systems can be utilized in *e.g.* context-aware devices, acoustic surveillance, bio-acoustical and healthcare monitoring, and smart-home cities.

In this thesis, we propose to apply the modern machine learning methods called deep learning for SED. The relationship between the commonly used time-frequency representations for SED (such as mel spectrogram and magnitude spectrogram) and the target sound event labels are highly complex. Deep learning methods such as deep neural networks (DNN) utilize a layered structure of units to extract features from the given sound representation input with increased abstraction at each layer. This increases the network's capacity to efficiently learn the highly complex relationship between the sound representation and the target sound event labels. We found that the proposed DNN approach performs significantly better than the established classifier techniques for SED such as Gaussian mixture models.

In a time-frequency representation of an audio recording, a sound event can often be recognized as a distinct pattern that may exhibit shifts in both dimensions. The intra-class variability of the sound events may cause to small shifts in the frequency domain content, and the time domain shift results from the fact that a sound event can occur at any time for a given audio recording. We found that convolutional neural networks (CNN) are useful to learn shift-invariant filters that are essential for robust modeling of sound events. In addition, we show that recurrent neural networks (RNN) are effective in modeling the long-term temporal characteristics of the sound events. Finally, we combine the convolutional and recurrent layers in a single classifier called convolutional recurrent neural networks (CRNN), which emphasizes the benefits of both and provides state-of-the-art results in multiple SED benchmark datasets.

Aside from learning the mappings between the time-frequency representations and the sound event labels, we show that deep learning methods can also be utilized to learn a direct mapping between the the target labels and a lower level representation such as the magnitude spectrogram or even the raw audio signals. In this thesis, the feature learning capabilities of the deep learning methods and the empirical knowledge on the human auditory perception are proposed to be integrated through the means of layer weight initialization with filterbank coefficients. This results with an optimal, ad-hoc filterbank that is obtained through gradient based optimization of the original coefficients to improve the SED performance.

# Preface

This study was carried out at Tampere University of Technology (TUT), Finland between 2015 and 2018.

First and foremost, I would like to express my sincere gratitude to my supervisor Tuomas Virtanen, for accepting me into Audio Research Group (ARG) as a research assistant in 2014, and supporting, guiding and encouraging me in my academic career since then.

I would like to thank all of my co-authors, Toni Heittola, Giambattista Parascandolo, Ezgi Ozan, Konstantinos Drossos, Sharath Adavanne, and Heikki Huttunen. This thesis would not be possible without your deeply valuable contributions.

I would like to thank the pre-examiners of this thesis, Dan Stowell and Justin Salamon, and the opponents of the public defense of this thesis, Tomi Kinnunen and Dan Stowell.

I would like to thank all the members of ARG for creating such a friendly and motivating research atmosphere. I would like to especially thank my office mates Gaurav Naithani, Tuomo Tuunanen, and Shuyang Zhao.

I would like to thank Anssi Klapuri for offering me a part-time job in his company and giving me the chance to integrate and explore my research findings in a real-life scenario.

Living in a country with such different culture is not always easy. I would like to thank Joonas Riikonen, Veera Repo, Pyry Urhonen, Visa Savolainen and Ville Laine for their great friendship and making Finland feel like home.

Finally, I would like to thank my mother, Fatma, my father, Ünal and my sister, Bilge for supporting me in this academic journey abroad.

# Contents

# Acronyms

| | |
|---|---|
| AIR | audio information retrieval |
| ANN | artificial neural network |
| ASR | automatic speech recognition |
| AUC | area under curve |
| CNN | convolutional neural network |
| DCT | discrete cosine transform |
| DNN | deep neural network |
| EER | equal error rate |
| ER | error rate |
| ERB | equivalent regular bandwidth |
| FNN | feed-forward neural network |
| FPR | false positive rate |
| GAN | Generative adversarial networks |
| GMM | Gaussian mixture model |
| GRU | gated recurrent unit |
| HMM | hidden Markov model |
| HOG | histogram of oriented gradients |
| KNN | K-nearest neighbor |
| MFCC | mel frequency cepstral coefficient |
| MP | matching pursuit |
| NLP | natural language processing |
| NMF | non-negative matrix factorization |
| ReLU | rectified linear unit |
| RNN | recurrent neural network |
| ROC | receiver operating characteristic |
| SEC | sound event classification |
| SED | sound event detection |
| SGD | stochastic gradient descent |
| SIF | spectrogram image feature |
| SNR | signal-to-noise ratio |
| STFT | short-time Fourier transform |
| TPR | true positive rate |
| VAE | variational autoencoders |

# Nomenclature

### Latin alphabet

| | |
|---|---|
| $X$ | scalar |
| $\mathbf{x}$ | vector with entries $x_i$ |
| $\mathbf{x}_t$ | column vector representing the entries of column $t$ for the matrix $\mathbf{X}$ |
| $\mathbf{X}$ | matrix with entries $X_{i,j}$ |
| $\mathcal{X}$ | tensor/array with three or more dimensions |
| $\mathbf{X}_t$ | matrix representing the entries for the tensor $\mathcal{X}_{:,:,t}$ |

# List of Publications

This thesis contains the following peer-reviewed publications, preceded by an introduction to the background and main points for each publication. Original publications are reproduced with permission of the respective copyright holders. The included publications are referenced in the text as [I] to [VII].

I Emre Çakır, Toni Heittola, Heikki Huttunen, Tuomas Virtanen, "Polyphonic Sound Event Detection Using Multi Label Deep Neural Networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN).*, Killarney, Ireland, July 2015, pp. 1-7.

II Emre Çakır, Toni Heittola, Heikki Huttunen, Tuomas Virtanen, "Multi-label vs. Combined Single-label Sound Event Detection with Deep Neural Networks," in *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO).* Nice, France, September 2015, pp. 2551-2555.

III Emre Çakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, Tuomas Virtanen, "Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection," in *IEEE/ACM Transactions on Audio, Speech and Language Processing*, volume 25, issue 6, pp. 1291-1303., 2017.

IV Emre Çakır, Sharath Adavanne, Giambattista Parascandolo, Konstantinos Drossos, Tuomas Virtanen, "Convolutional Recurrent Neural Networks for Bird Audio Detection," in *Proceedings of the 25th European Signal Processing Conference (EUSIPCO).* Kos, Greece, September 2017, pp. 1744-1748.

V Emre Çakır, Tuomas Virtanen, "Convolutional Recurrent Neural Networks for Rare Sound Event Detection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017).* Munich, Germany, November 2017, pp. 27-31.

VI Emre Çakır, Ezgi Ozan, Tuomas Virtanen, "Filterbank Learning for Deep Neural Network Based Polyphonic Sound Event Detection," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN).* Vancouver, Canada, July 2016, pp. 3399-3406.

VII Emre Çakır, Tuomas Virtanen, "End-to-End Polyphonic Sound Event Detection Using Convolutional Recurrent Neural Networks with Learned Time-Frequency Representation Input," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN).* Rio de Janeiro, Brazil, July 2018, pp. 2412-2418.

# Author's contribution to the publications

Emre Çakır is the primary author of all publications in this thesis. The work leading to each publication has been supervised by Tuomas Virtanen, in the means of regular discussions for developing the ideas and giving feedback during the experiment and documentation process. All the work mentioned in this thesis has been done at the Laboratory of Signal Processing, Tampere University of Technology, Finland.

The initial idea for the proposed methods in [I – III] has been proposed by Tuomas Virtanen, and for [IV – VII] by Emre Çakır. Ezgi Ozan has contributed for developing the idea in [VI]. The software code and the experiments for all publications have been conducted by Emre Çakır. Toni Heittola has assisted for the code in [III – V]. Giambattista Parascandolo has assisted for the code and the experiments in [III], and he is also the equal main author for that work. Heikki Huttunen has been the co-supervisor for the work done in [I – III].

# 1 Introduction

The world witnessed a rapid development in the consumer electronic devices in the last few decades. This affected the interaction of humans with the electronic devices in a major way. For instance, computers are not giant, bulky machines that are only used by the scientists for specific purposes anymore. They are smaller, more powerful, easier to use, and the vast amount of the world's population are using them daily for leisure or work purposes. The introduction of smartphones, pocket-size computers with phone capabilities, has led to an even tighter bond between the humans and the computers.

With the increased familiarity and the bond with the humans and the electronic devices, the technology industry has been looking to finding ways of integrating better the computers to the daily lives of the humans. In this direction, one of the goals is to make devices that can recognize and understand the things and events happening around them without any user input, and then perform some operations based on their understanding. Such property is called context awareness [93], and in fact, it has already been introduced to a certain degree in some of the commonly used and well-known electronic devices. For instance, certain smartphone apps provide *e.g.* restaurant recommendations based on the geographical location of the users, extracted automatically using the GPS property. Some smartphones also have the function to adjust the ringtone volume automatically based on the ambient noise level (low volume in quiet settings such as an office meeting, high volume in *e.g.* a noisy street). In addition, certain surveillance cameras alert their users automatically when there is an unexpected movement detected. Finally, automatic emergency braking has been introduced in recent cars, which makes the vehicle stop without the user input when an object/human is automatically detected as dangerously close.

Designing more advanced context-aware devices is a challenging task. Currently, most of these devices can understand some aspects of their environment (*e.g.* movement detection in surveillance video, or loudness detection in audio), but they cannot understand what is the actual source of the input, which would have opened up a whole lot of doors for context-awareness. For instance, a device that can understand the commonly known sounds such as a doorbell, a baby cry, or a car horn can be well used by hearing impaired people as a personal assistant

device. Another example would be the autonomous vehicles, which continuously recognize the road lanes, other vehicles, road signs, vehicle horns etc. to drive the vehicle automatically.

In this thesis, we focus on the automatic detection of sound events in real-world environments. The sound events define a significant portion of the characteristics of a physical context. Therefore, one of the senses that we use most while interacting within our physical context is hearing. Humans are very good at interpreting and assigning meanings to the sounds. On the other hand, computers still cannot offer reliable accuracy for this task. In this thesis, we propose and develop methods that can be used for automatic sound event detection (SED). This task differs from another well-studied audio information retrieval task called automatic speech recognition (ASR), in that the aim is not to map the speech audio into words/phonemes, but to map non-speech audio to their corresponding semantic labels.

## 1.1 Sound Event Detection

Sound event is defined as "an audio segment that can be labeled as a distinctive concept in an audio signal" [43][VI]. In our daily lives, we frequently encounter various sound events such as door bell, car engine, footsteps, keyboard sounds etc. In addition, music and speech can also be considered broadly as sound events, independent of their content such as genre, notes, words etc. Sound event detection (SED) involves determining the start and end times of the sound events in an audio signal and associating them with their corresponding textual labels. The main goal of SED is to correctly detect the sound events present in the audio signal. Estimating the exact start and end time is of secondary importance and the errors on this regard are often tolerated for around $\pm 250$ ms [70].

SED can be considered in two main categories as *monophonic* and *polyphonic* [68]. Monophonic SED systems can detect maximum one (often the most prominent) sound event at any time regardless of the actual number of sound events present at that given time. The limit on the number of detectable sound events is a disadvantage for the real-life applicability of such systems, because in real-life the sound events often occur simultaneously. For instance, an audio signal recorded in a busy street may contain car horn, sirens, humans speaking loudly, all occurring simultaneously. On the other hand, in *polyphonic* SED, the goal is to detect multiple sound events simultaneously present at any given time instance, which suits better for real-life applications. In this case, the number of sound events to be detected can be varying between the time instances.

In the literature the terminology on SED varies between authors; with the terms such as sound event *detection*, *tagging* and *classification* being used interchangeably. Sound events are defined with pre-determined labels called sound event *classes*. In this thesis, sound event classification (SEC) (or sound event tagging) refers

to labeling the whole audio recording with the sound event class(es) present, regardless of the start and end time. On the other hand, an SED task includes the onset/offset detection for each occurrence of the class(es) present in the recording. This is often done by first dividing the recording into equal-length segments, then performing classification within each segment and finally combining the classification outputs for the consecutive segments. The segment length determines the *temporal resolution* of the SED system. Since both SED and SEC aim to solve very similar problems, the methods that are proposed for these tasks are often motivated from each other. Some of the SEC datasets even consist of 4-10 second chunks of real-life audio recordings, and this task can be considered as SED with low temporal resolution. For this reason, some of the methods proposed for SEC are also covered in this thesis. Finally, we use the term *machine hearing* to generally refer to the sound related machine learning tasks, such as ASR, SED, SEC, music transcription etc.

## 1.2 Objectives of the Thesis

The main objective of this thesis is to study and utilize the recently proposed advanced machine learning techniques in the context of SED. These techniques include deep neural networks (DNN), convolutional neural networks (CNN) and convolutional recurrent neural networks (CRNN). While utilizing these techniques for various SED tasks, we aim to develop an understanding of the working principles of the neural networks for a specific SED problem. The wide range of SED tasks that are tackled in this thesis provide a clear idea on the scalability and the robustness of these machine learning techniques. Moreover, we aim to propose novel methods in terms of sound representation for neural networks and the post-processing of the network outputs for more realistic and smooth estimation of the onset/offset times. Lastly, we aim to investigate the *end-to-end* SED and propose a novel, ad-hoc sound representation method which is obtained through the hidden layer outputs of a neural network.

In this thesis, the main research questions that we ask can be listed as follows. We investigate whether the modern machine learning techniques such as deep learning methods can be used to develop SED systems with robust performance in real-life conditions. We test the multi-label learning capabilities of the deep learning methods for SED in various polyphony levels. We investigate on which deep learning methods perform the best for a given SED task, and how should the model architecture be adjusted for the optimal performance. We question the effectiveness of the established sound representation techniques, and we search for ways to make deep learning methods model the sound events directly from raw audio signals in a polyphonic setting.

## 1.3 Main Results of the Thesis

The main results and the contributions of our publications included in this thesis are listed as follows.

**Publication 1: Polyphonic Sound Event Detection Using Multi Label Deep Neural Networks**

In publication [I], deep neural networks (DNN) with mel band energy input has been proposed for polyphonic SED. DNN offers a considerable 42% relative performance improvement over the conventional GMM-HMM classifiers for the task of polyphonic SED in real-life environments. In addition, median filtering based post-processing provides an efficient solution to smoothing the noisy, frame level predictions of the DNNs for SED, where the noise is often the result of the coarse time resolution of the reference annotations.

**Publication 2: Multi-label vs. Combined Single-label Sound Event Detection with Deep Neural Networks**

In publication [II], the validity of the claim that class correlation information is essential for multi-label classification is investigated for polyphonic SED using DNNs. The performance drop is very limited with combined single label DNN classifier versus a multi-label classifier for polyphonic SED. Combined single label DNNs provide a different option for polyphonic SED with the benefit of usage case flexibility in exchange for minimal to no decrease in detection performance.

**Publication 3: Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection**

In the proposed convolutional recurrent neural network (CRNN) method in publication [III], the capability of CNNs to learn spectral and temporal shift invariant filters and the capability of the RNNs to model long term temporal dependencies are combined, resulting with a more powerful classifier than both CNN and RNN for SED. CRNN provides the state-of-the-art results for various SED datasets, beating the previous DNN state-of-the-art by a considerable margin.

**Publication 4: Convolutional Recurrent Neural Networks for Bird Audio Detection**

In publication [IV], CRNN is proposed for bird audio detection as a part of Bird Audio Detection 2017 challenge. CRNN is especially suitable for the task of bird audio detection due to the acoustic characteristics of the bird sounds, and this is also evident in the performance of the method for the Bird Audio Detection challenge 2017, where the proposed CRNN method came second.

**Publication 5: Convolutional Recurrent Neural Networks for Rare Sound Event Detection**

In publication [V], CRNN is proposed for rare SED as a part of DCASE 2017 challenge. The proposed method suits the rare SED task due to its capability of learning high level features for the rare sound events and reflecting the sudden changes between the consecutive frame features. This theoretical advantage is also supported by the evaluation results, where the proposed CRNN method comes second in the challenge.

**Publication 6: Filterbank Learning for Deep Neural Network Based Polyphonic Sound Event Detection**

The proposed method in [VI] is a first attempt to learn an ad-hoc filterbank for SED using the feature learning capabilities of deep learning methods and the empirical knowledge about the human auditory perception. Initializing the first convolutional layer filter weights of a CNN with the mel filterbank magnitude response provides a modest increase in performance over a CNN whose weights are initialized randomly.

**Publication 7: End-to-End Polyphonic Sound Event Detection Using Convolutional Recurrent Neural Networks with Learned Time-Frequency Representation Input**

In publication [VII], polyphonic SED is proposed to be done using directly raw audio data as input to a single deep learning classifier. The classifier includes a feature extraction block composed of feed-forward layers whose weights are initialized to get the spectrogram at the block output, and a convolutional recurrent layer block. To the author's knowledge, the proposed method is the first to integrate the the domain knowledge of the perception based sound representations into the parameters of a deep learning classifier to conduct end-to-end SED. While the proposed method does not outperform a CRNN classifier with mel spectrogram as input, it offers a data-driven alternative to designing audio filterbanks for SED by learning optimized filterbank parameters with the gradient descent algorithm to minimize the SED loss.

## 1.4 Outline and Structure of the Thesis

The organization of the remainder of this thesis is as follows.

The background information about SED, and the topics of machine learning, feature learning, artificial neural networks, and evaluation methods in the context of SED are presented in Chapter 2.

Chapter 3 presents two DNN based approaches for polyphonic SED, multi-label DNNs and combined single label DNNs. The advantages of both methods in terms of accuracy, computational cost, and usage case flexibility are investigated.

In Chapter 4, CRNNs have been proposed for three separate SED tasks: polyphonic SED, bird audio detection, and rare SED. The theoretical explanation and the empirical evidence for why CRNN suits these tasks are presented.

Filterbank learning approaches for polyphonic SED are studied in Chapter 5 over two novel methods proposed in [VI] and [VII]. Both these methods aim to combine the feature learning capabilities of deep learning classifiers with the established acoustic feature extraction methods in spectral domain.

Conclusions of this thesis and discussions for the current and future research on SED are provided in Chapter 6.

# 2 Background

## 2.1 Problem Formulation

The goal of sound event detection (SED) is to automatically estimate the start and end times of the sound events present for a given collection of audio recordings, and then to associate a textual label to each of these sound events. These textual labels are often called *classes*. SED can be formulated in two stages: sound representation and classification. In the sound representation stage, acoustic features are extracted for each short time frame $t$ in the audio recording to obtain a feature vector $\mathbf{x}_t \in \mathbb{R}^M$, where $M$ is the number of features per frame. In the classification stage, the task is to learn an acoustic model that would estimate the event presence probabilities $p(\mathbf{y}_t|\mathbf{x}_t, \theta) \in [0, 1]^t$ for each pre-defined sound event class, where $\theta$ represents the acoustic model parameters of the classifier. In the usage case, the event presence probabilities are binarized by e.g. constant thresholding to obtain the event presence predictions $\mathbf{y}_t \in [0, 1]^t$. By combining the presence predictions for consecutive time frames, one can determine the start and times of the sound event classes.

In the scope of this thesis, acoustic model parameters $\theta$ are optimized using supervised learning (Section 2.5). There are also other learning methods to optimize the model parameters, such as unsupervised learning (often used when target outputs are unavailable/unused) and semi-supervised learning (used when the target outputs are available for only a portion of the data). In supervised learning for SED, the binary target outputs $\mathbf{y}_t$ for each frame $t$ are obtained from the reference annotations, which include the start and end time of each event in the audio recordings. If the recordings are obtained from the real-life environment, then the reference annotations are often collected manually, *i.e.* by a human listening through the recordings and labeling the start and end times of the sound events that they could notice. Supervised learning for SED is formulated in such a way that the sound event classes of interest are defined beforehand. This makes the SED task concrete, and helps the human annotator in omitting the irrelevant sound events and grouping different sound events under a certain class (*e.g.* different kinds of doorbells under a single doorbell class). An audio recording annotated with the sound events is visualized in Figure 2.1.

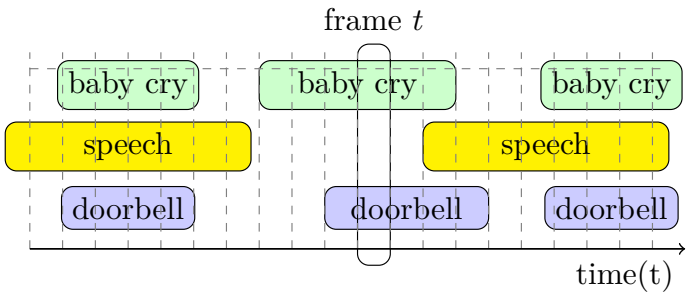


Figure 2.1: A sound recording labeled with the sound event classes in the time domain. During frame $t$, baby cry and cat meow sound events are present.

## 2.2 Applications

Human perception of the outside world is mostly driven by the audiovisual cues collected from the environment. On the audio side, detecting sound events represent an important part of the auditory human perception. For instance, the acoustic characteristics of a busy street can be defined by the sound events such as car engine sound, car horn, footsteps, speech etc. Therefore, automatic sound event detection can be utilized for the task of creating context-aware devices [10, 110]. In addition, according to World Health Organization [107], around 5% of the world's population suffer from disabling hearing loss. Creating assistant devices that can perform automatic SED would help the hearing impaired people significantly in their daily lives. SED can also be utilized in computational bioacoustics [100], *e.g.* remote wildlife monitoring of bird species which involves the detection of bird sounds [95], and in healthcare for automatic detection of *e.g.* coughing sounds of patients [31, 80].

Recently, SED systems have been commercialized into smart-home products [58, 105]. Some smart-home systems utilize SED for the purpose of surveillance by detecting sound events such as glass breaking, gun shot, smoke alarms etc [85]. The benefit of using SED instead of image-based surveillance methods is that audio signals are not affected by illumination, and microphones can be effectively used to cover a wider area than the cameras. In addition, SED has been proposed for urban sound analysis in smart cities, for tasks such as audio surveillance and noise pollution monitoring [4, 75, 92].

Online multimedia sharing has increased exponentially, which resulted in millions of hours of multimedia data in content sharing sites such as YouTube. While the users provide certain descriptors for each video, there is often a large amount of non-annotated information, which can be extracted with the techniques such as SED. In order to facilitate SED research on this direction, AudioSet [25] dataset has been released. It is composed of around 5000 hours of YouTube video material annotated by human annotators with sound events within 10 second clips. While

the time resolution of the annotations are rather coarse than what is often expected from a real-world SED system, AudioSet provides a benchmark dataset for SED research, and the SED acoustic models learned from this vast amount of data can be then utilized in real-world SED tasks.

## 2.3 Challenges

It can be claimed that the research progress on SED has been stagnant until the recent years. One of the reasons is that there are several challenges for a robust SED system that can operate in real-life conditions. The challenges for SED systems can be listed as, but not limited to, intra-class variability, overlapping sound events, environmental noise, lack of structure, and definitive ambiguity. It should be noted that some of the mentioned challenges are not specific to SED and also apply for other machine hearing tasks such as ASR, music transcription and musical genre classification.

#### Intra-class Variability

Sound event classes for SED are often defined broadly such as phone ringing, doorbell etc., and this presents a challenge for SED methods in the form of intra-class variability. For instance, doorbell class can be used to represent all types of doorbells, whose acoustic characteristics can vary significantly among the examples of this class. Therefore, in order to claim that an SED system can robustly detect doorbells, it should be able to do so on a wide variety of doorbells. This requires the SED method to be able to detect or extract the acoustic features that are found in common among different examples of the same class.

#### Overlapping Events

The earlier research on SED has been focused on the detection of individual sound events recorded in isolated environments [26]. However, in the real world, sound events often occur simultaneously. For instance, a recording from a children's park may include children shouting, adults speaking, footsteps and birds singing; all happening at the same time. Since the audio signals are additive, the resulting audio recording includes a mixture of all of these sound events. Therefore, the SED system should be able to distinguish the acoustic characteristics of each individual sound event among this mixture.

#### Environmental Noise and Recording Conditions

The scope of the SED task is defined by the pre-determined sound event classes. The sound events that are not in the scope of the given SED task can be essentially deemed as the background noise in the recording. For instance, the wind is very commonly present in the real-world recordings, and it can significantly decrease the

signal-to-noise ratio (SNR). In addition, the variations in the recording conditions such as the distance of the recording device to the sound source and the type of the used recording may present an additional challenge for the SED systems.

#### Lack of Structure

Certain audio signals such as speech and music contain some structure that can be used to extract informative sound representations from the signal. For instance, speech can be divided into certain phonemes and the characteristics of each phoneme can be investigated separately. This makes it easier to find a mapping between the phonemes and the language representations, then combine the representations to recognize the whole speech (which is the task of ASR). The same property can be claimed for music and its subdivision, the notes. On the other hand, it is not possible to find a common definition of subdivision for the sound events, which makes the task of SED challenging.

#### Definitive Ambiguity

It may not always be possible to exactly determine the onset and offset of a sound event. Certain sound events such as *car passing by* have relatively long rise and fall times, and the labeling of onset and offset for this event is often left to the subjective decision of the annotator, or automated based on a certain signal energy threshold. Besides, the interpretation of some of the repetitive sound events can be a source of ambiguity. For instance, while the annotators would most likely label a 10-second audio segment of human footsteps as a single sound event, a machine learning system with access to high time resolution features may aim to label these events individually, since each footstep is distinctively different from the background. Finally, the annotators often have to make subjective, loudness perception based decisions on whether a sound event can be recognized as a distinctive event in the clip or it should be considered as a part of the background. Factors such as SNR level and distance to the source play an important role on these subjective decisions. Definitive ambiguity can also be a challenge for tasks such as ASR (*e.g.* muttering, whispering) and musical genre classification.

## 2.4 Sound Representation

Audio signals for SED are obtained by digitally recording the sound events in a real-life environment or in a studio. The time domain representation of a sound event is considered as the lowest level representation, since the signal is not much processed before using it as the representation of a sound event. On the other hand, this representation is quite redundant for a classifier to learn which sound event it belongs to. For this reason, audio signals for SED are often represented by extracting certain *acoustic features*. The acoustic features are mostly extracted in the frequency domain, as the signals belonging to the same

sound event often share components in the frequency domain. Besides, frequency domain representation is more compact and noise robust than the time domain representation. The number of processing steps over the time domain signal before obtaining the acoustic features determine the level of abstraction of the sound representation. For instance, mel frequency cepstral coefficients (MFCC) and histogram of gradients (HOG) features, both explained below, are considered higher level representations, as the calculation of these features require multiple processing steps in frequency domain, and therefore the representations become more abstract.

### Stages of acoustic feature extraction

There are three main stages of acoustic feature extraction in frequency domain: frame blocking, windowing, and frequency spectrum calculation. In order to obtain the frequency spectrum through Short-time Fourier Transform (STFT), the signal should be assumed to be able to model a sum of stationary sinusoids. Hence, the frequency spectrum of the audio signals are calculated by first dividing the signal into short time frames. This process is called *frame blocking*. There is a trade-off between frequency resolution and time resolution based on the frame length. Frequency resolution increases with increased frame length, resulting in worse time resolution. Therefore the selection of frame length is dependent on the machine hearing task at hand. For SED, frame length is often selected in the range of 20 to 50 ms. An overlap of 25% to 50% of the frame length is often selected between the frames to obtain a smoother representation. Then, each short time frame signal is multiplied with a window function. This process is called *windowing*, and it is done in order to avoid the discontinuities at the borders of the frame, which would corrupt the frequency spectrum estimation. Hamming, Hann and Blackman functions are often used for windowing in SED. Finally, the frequency domain representation of each short time frame signal is obtained by taking the discrete Fourier transform. The stages of acoustic feature extraction has been visualized in Figure 2.2.

### Spectrogram

The time-frequency domain feature matrix that is obtained by concatenating the frequency domain feature vectors for the consecutive time frames of a recording is called the *spectrogram*. The spectrogram of an audio signal often provides the base of sound representation for SED. As the Fourier transform is a complex-valued function, the spectrogram consists of complex values. However, most machine learning methods are designed to work only with real-valued input. Therefore, often in machine hearing, the phase information is discarded as it is deemed less informative [27], and only the magnitude of the spectrogram is used. Fourier

[1]Reprinted by permission from Springer: *Computational Analysis of Sound Scenes and Events* by Virtanen, Plumbley, and Ellis, 2018.

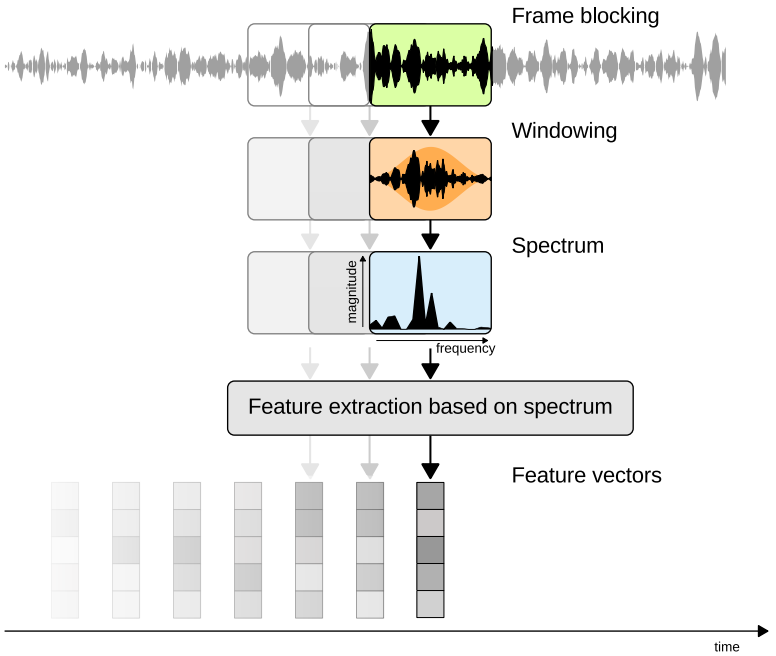


Figure 2.2: The stages of acoustic feature extraction in frequency domain[1].

transform has a linear frequency resolution, and the sound events often have much higher energy in the lower frequency levels, so these lower frequency components dominate as the sound features. The dynamic range of the linear magnitude spectrogram can be compressed by taking the logarithm to obtain log magnitude spectrograms.

Using spectrograms as the sound representation for SED is beneficial in the following ways. Compared to raw audio signal in time domain, spectrogram provides more compact and rich information about the sound events based on the relative distribution of energy in the frequency domain [44]. In addition, similar to images, spectrogram is multi-dimensional, which makes the vast research on machine learning methods developed for image classification based tasks applicable to SED. It can also be claimed that spectrograms are more robust to noisy environments than time domain audio signals, because the environmental noise often tends to be limited to the lower frequencies, and the obtained SED performance is often better than using the raw audio signals.

### Mel spectrogram

There are several spectrogram representation methods that are based on human auditory perception. Empirical results [2] show that humans do not perceive sounds through a linear frequency scale, and we are more sensitive to the changes in the lower frequency range than the higher frequency range. *Mel scale* (illustrated

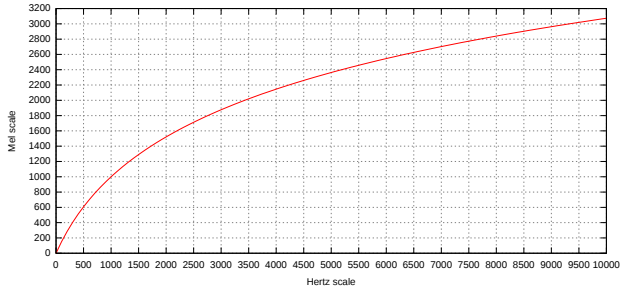


Figure 2.3: Hertz vs. Mel scale.

in Figure 2.3) is a non-linear frequency scale in which the pitches are adjusted by the human listeners to be perceived as equally spaced [98]. Mel scale based sound representations include mel spectrogram and mel frequency cepstral coefficients (MFCCs). Mel spectrogram is a matrix that consists of *mel band energy* feature vectors concatenated for consecutive time frames, and it is obtained by applying the mel filterbank at each time frame over the magnitude spectrogram. Mel filterbank utilizes the mel scale and it consists of triangular filters, whose bandwidths widen with increasing central frequencies for the filters. This results in higher frequency resolution in the lower frequency range, and vice versa. Mel spectrogram is often further processed by taking the logarithm to compress the dynamic range, resulting in the log mel spectrogram. Currently, log mel spectrogram is the most popular sound representation for SED tasks, and have been used in many state-of-the-art methods for polyphonic SED [1][III], rare SED [65], and SED using weakly labeled data [63]. The number of mel filterbanks for SED is often selected in the range of 40 to 80, which is most likely smaller than the number of frequency bins used in STFT. Therefore, mel spectrogram provides a more compact representation than the magnitude spectrogram. Several sound representation methods have been visualized in Figure 2.4.

### Mel Frequency Cepstral Coefficients (MFCC)

MFCCs [14] are obtained by applying Discrete Cosine Transform (DCT) over the log mel spectrogram. Mel filters are overlapping and this results with the correlation between adjacent filterbank outputs, therefore DCT is used to approximately decorrelate the log mel spectrogram features. The higher MFCC coefficients are often discarded as they provide little information. The first coefficient is also typically discarded, as it is simply equal to the average log energy and does not give information about the spectral characteristics. The first 10-16 coefficients are used as the acoustic features for each short time frame. MFCCs are often concatenated with delta-MFCC features, which are calculated from the difference between the MFCC features for consecutive time frames. MFCCs have been extensively used for machine hearing tasks, especially for ASR. MFCCs (and delta-MFCCs) have been combined with a Gaussian Mixture Model (GMM) -

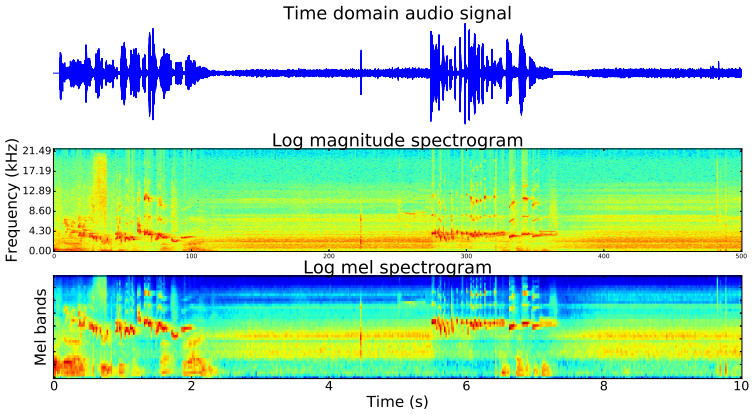


Figure 2.4: Time domain (top), log magnitude spectrogram (middle), log mel spectrogram for a 10 second audio signal, which includes birds singing and humans speaking in a park.

Hidden Markov Model (GMM) based classifier to conduct polyphonic SED in everyday environments in [68]. In that work, MFCCs for each time frame have been modeled as samples from mixtures of Gaussians, and the temporal context has been modeled through a three-state HMM. With recent machine learning techniques such as deep neural networks, it became clear that the classifiers do not necessarily require the decorrelation step over the log mel spectrogram, and they perform better by simply using log mel spectrograms as input. For this reason, MFCCs have recently dropped out of favour as the sound representation method for SED. Finally, unlike magnitude spectrogram and mel spectrogram, the concatenation of MFCC features for consecutive time frames can not be necessarily deemed as a time-frequency representation, as the MFCC coefficients are not ordered in frequency axis.

### Other acoustic feature extraction methods

Apart from mel scale based representations, there are also other sound representation methods that stem from the magnitude spectrograms. For instance, Gammatone spectrogram (or gammatonegram [21]) is another human auditory perception based acoustic feature extraction method, which has been commonly used for machine hearing [104] and recently proposed for rare sound event detection [81]. The central frequencies for Gammatone filters are calculated based on the equivalent regular bandwidth (ERB) scale [28]. In addition, spectrogram image feature (SIF) is an image processing inspired method that extracts a visual cue from the spectrogram of the audio signal, and it has been proposed for SEC in [15, 16]. It consists of quantizing the magnitude spectrogram into an RGB image based on the normalized amplitude values of the spectrogram (red channel represents high amplitude components, and blue low amplitude components, as in the "jet" colormap for matlab and matplotlib). This is analogous to pseudo-

coloration in image processing. Another image processing based feature extraction method is the histogram of oriented gradients (HOG) [13], which utilizes the change of intensity (amplitude) in the sub-blocks of the spectrogram. HOG has been proposed for acoustic scene and event classification in [86].

## 2.5 Machine Learning for Sound Event Detection

Machine learning is a field of computer science that aims to design machines (or software) that are able to learn directly from the given data. Machine learning systems are especially useful for certain tasks for whom the solutions are very challenging to implement as an algorithm by a human engineer. Sound event detection can be given as an example for such tasks. Due to the challenges explained in Section 2.3, it is difficult to come up with an engineered algorithm that could map audio signals with their corresponding sound events with satisfactory accuracy to be utilized in real-world applications. By studying how machine learning systems for sound event detection process audio data, it may even be possible in the future to gain more insight about how humans perceive sound events.

According to [73], "A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$". In this thesis, the task $T$ is generally defined as sound event detection, and in some cases more specifically as polyphonic sound event detection in real-life environments, bird audio detection, rare sound event detection etc.

In the case of SED, the experience $E$ often corresponds to the acoustic features and the sound event labels for a collection of audio recordings. Acoustic features can be represented as an input matrix $\mathbf{X} \in \mathbb{R}^{M \times T}$ which includes the $M$ acoustic features extracted from each frame in each audio recording, where the total number of time frames is $T$. The labels are binary encoded as the target output matrix $\mathbf{Y} \in 0, 1^{C \times T}$, where $C$ is the number of predefined sound event labels such as doorbell, baby cry, dog barking etc. The target outputs are obtained from the available reference annotations (Section 2.1). If the $i^{th}$ sound event is present in the $j^{th}$ frame, then $Y_{i,j}$ is set to 1, and 0 vice versa. If the reference annotations for the sound events are available (which is the case in this thesis), the task of sound event detection is a form of *supervised* machine learning. For the performance measure $P$, various metrics such as F1 score and error rate [70] are used (Section 2.8.1).

The aim of machine learning for SED is to learn a function (*i.e.* model) $f$ that can map the given input data $\mathbf{X}$ to the target outputs $\mathbf{Y}$. In the case of SED, for a given acoustic feature vector $\mathbf{x}$ for a single time frame, $f(\mathbf{x})$ outputs the probability vector $\hat{\mathbf{y}} \in [0, 1]^C$ for the sound events being present in the given frame. During the learning stage, the parameters of $f$ are updated to minimize

a certain error function between the estimated output $\hat{\mathbf{Y}}$ and the target output $\mathbf{Y}$. During the usage stage, where the target outputs are either not available or only available for evaluation, the utilization of the estimated outputs depend on the type of the SED task. For monophonic SED, the sound feature vector $\mathbf{x}$ for a single frame is mapped to the sound event label with the highest probability (*i.e.* $\arg\max(\hat{\mathbf{y}})$) for the given frame. For polyphonic SED, $\hat{\mathbf{y}}$ is often binarized using a certain threshold (mostly 0.5 as it is an unbiased threshold), and $\mathbf{x}$ is mapped to the label(s) that have the probability above the threshold.

In supervised machine learning for SED, the input and the target output representations depend on the selected machine learning method. For some methods such as Gaussian mixture models (GMM) and feed-forward neural networks (Section 2.7.1), a pair $(\mathbf{x}, \mathbf{y})$ of acoustic feature vector $\mathbf{x}$ and the target output vector $\mathbf{y}$ represents a single *learning example*. Other methods such as RNNs (Section 2.7.4) and CNNs (Section 2.7.5) utilize the temporal information from a sequence of frames to calculate the estimated output at each frame. For these methods, the input $\mathbf{X}$ and the target output $\mathbf{Y}$ matrices are typically divided into $T'$ non-overlapping sequences of $N$ frames, resulting with the input tensor $\mathcal{X} \in \mathbb{R}^{M \times N \times T'}$ and the target output tensor $\mathcal{Y} \in \mathbb{R}^{C \times N \times T'}$, where $T = N \times T'$. In this case, a pair $(\mathbf{X}_t, \mathbf{Y}_t)$ represents a single learning example.

The earlier work on machine learning for SED mainly involved utilizing the techniques that were proposed earlier for other machine hearing tasks such as music transcription and ASR. In [42, 68], MFCCs were used as the acoustic features and GMM-HMM classifiers for modeling the sound events through these features. However, the performance of these systems were not satisfactory to be deployed in a real-world application. In [11], several time and frequency domain audio features such as zero-crossing rate, short-time energy, MFCCs, band energy ratio and spectral flux were concatenated in a single acoustic feature vector. The performance for these features were compared both separately and jointly with the time-frequency domain features extracted using the matching pursuit (MP) algorithm [67]. The task involved classifying a combination of sound events and acoustic scenes in 4-second length segments using the aforementioned features with K-nearest neighbor (KNN) [23] and GMM classifiers. The experimental results with joint MP-MFCC features provided close performance to human listeners on the same task. In [15], linear support vector machine (SVM) [6] classifiers were proposed to be trained with pseudo-colour quantized SIF features for an SEC task, which resulted with superior performance compared to an MFCC-HMM based method.

## 2.6 Feature Learning for Sound Event Detection

Machine learning for SED is usually performed by using the acoustic features extracted from the sound recordings as input. The acoustic features used in

machine hearing often do not utilize any information about the given classification task, *i.e.*, MFCCs are calculated in the same manner whether the task is *e.g.* ASR or SED. Recently, most of the proposed SED methods simply pick one of the well-known acoustic feature extraction methods for machine hearing, and focus on developing a novel and better performing acoustic model through machine learning. Although acoustic features such as log mel spectrograms perform satisfactorily for SED, it can be argued that sound representations that can adapt better to the given task would be beneficial. Besides, human auditory perception based representations such as mel spectrogram inherently assume that the way humans process the audio signals before mapping them to their corresponding sound events is similar to the way a machine does the same task. However, it is possible that the machines would benefit from a very different kind of sound representation for SED. The problem is that since the relationship with the audio signals and their corresponding sound events is highly complex (for the reasons mentioned in Section 2.3), it is hard to come up with an engineered sound representation method that would be optimal for SED.

Apart from the acoustic feature extraction methods explained in Section 2.4, machine learning can also be used to learn discriminative features from low-level representations for SED. Low-level representations may refer to either time domain signal, or the magnitude spectrogram. The method of using machine learning for the acoustic feature extraction is often called *feature learning* (or representation learning).

There are several feature learning methods that have been proposed for SED. In [19], non-negative matrix factorization (NMF) [62] has been proposed for SED. In this work, coupled NMF is used to decompose the magnitude spectrogram into an audio dictionary matrix and an excitation matrix. The same excitation matrix is used to decompose the target output matrix into an annotation dictionary matrix and an excitation matrix. In the usage case, the learned annotation dictionary and the excitation matrices are used to obtain the estimated output matrix for the unseen data. This method effectively uses the reference annotations of the sound events to learn a better representation through the audio dictionary matrix, therefore it is an example of supervised learning. In [43], unsupervised NMF is proposed to separate the magnitude spectrogram of the overlapping sound events into different streams containing (ideally) homogeneous spectral content, *i.e.* the spectral content of each sound event would be distributed into separate streams. This would make it easier to overcome the problem of overlapping sound events in the sound representation for SED. Since the reference annotations are not utilized in the feature learning stage, it is an example of unsupervised feature learning. Another example of unsupervised feature learning for SEC has been proposed in [90]. In this work, an over-complete codebook has been learned based on the whitened log mel spectrograms using *spherical k-means algorithm* [12]. The learned codebook of $k$ vectors (means) is then used to encode the log mel spectrograms into a different feature space. Recently in [113], time-frequency

domain filterbanks that operate over the magnitude spectrogram and are learned by neural networks have been proposed for audio source separation.

Feature learning methods for SED often operate over the spectrograms of audio signals. The benefits of using spectrograms over time domain audio signals have been listed in Section 2.4. Recently, the idea of combining the acoustic feature extraction and the acoustic model training stages of SED in a single machine learning system has been gaining traction. With this method, a machine learning system is trained to map directly the time domain audio signal to the sound events that are present in the signal. This method is often called *end-to-end learning* for SED. The benefit of end-to-end learning is that, there is no initial assumptions involved about the optimal sound representation, *i.e.* the machine learning system extracts the relevant and application specific acoustic features from the time domain signal by using the target output information in SED. Studies have showed that for some machine hearing tasks such as ASR, the learned representations of end-to-end systems in fact converge to some well-known sound representation methods like Gammatone spectrogram [89]. However, in the case of SED, spectrogram input representations have still been performing better than the time domain signals as input (more details in Section 5.2). Developing better machine learning methods to conduct end-to-end learning is an ongoing area of research.

## 2.7 Artificial Neural Networks

An artificial neural network (ANN) is a machine learning method that is loosely based on the information processing inside the human brain. The human brain is composed of 15-20 billion inter-connected nodes called neurons, which are stimulated by various electrochemical signals [79]. Different signals (such as sensory, audio, visual etc.) stimulate different paths of neurons, and the signal information is processed through a collective set of neuron stimulation inside the brain. From the birth of the human being, the neurons specialize in processing certain signals and continuously improve their abilities to create a mapping between the given input signal and its cognitive representation. For instance, human speech, a special kind of audio signal essentially used for communication, is first transformed into electrochemical signals inside the ear and brain. Then, through neuron stimulations, these signals are mapped to a certain set of phonemes, which have a shared cognitive representation for human communication. Most intensively during the infancy, the neurons inside the brain update their structure to be able to better map these signals into their cognitive representation, which leads to the human making sense of speech.

The working principles of ANNs are somehow similar to information processing methods of the brain. ANNs are composed of stacks of inter-connected artificial neuron blocks (often called *layers*) that aim to find a mapping between the given

input signal and the target output signal. Each neuron has certain parameters (such as the *weights* and *biases*) which are iteratively updated, often through gradient optimization, to minimize a certain error function between the desired target output and the estimated output. The layer that is receiving the input signal is called the input layer, the last layer which determines the network outputs is called the output layer, and the layers in between these layers are called the hidden layers. Each network has a certain set of *hyper-parameters* that determine the network architecture (number of hidden units in each layer, number of layers etc.) and the network training procedure (optimization method parameters, regularization parameters etc.).

The ANNs that include more than one hidden layer are often grouped under the name *deep learning*, or *deep neural network* (DNN). In this thesis, we reserve the term DNN for a feed-forward neural network with multiple hidden layers. In the case of other neural network techniques such as convolutional and recurrent neural networks, the *deep* prefix is often omitted although these networks also most often utilize multiple hidden layers. In this thesis, we use the term deep learning to refer to all ANN methods that utilize multiple hidden layers. With the introduction of advanced training techniques, large datasets and increased computational power, deep learning has been recently dominating the machine learning field in many tasks. Consequently, the current state-of-the-art methods for sound event detection also utilize deep learning [63, 65][III].

### 2.7.1 Feed-forward Neural Networks

Feed-forward neural network (FNN) consists of sequential layers of fully connected neurons, *i.e.* the output of each neuron in a single layer is given as input for all the neurons in the next layer. The layer outputs are calculated through a forward pass, and there is no feedback connection for the neuron outputs. When the FNNs are extended to have feedback connections for the neuron outputs, they are called recurrent neural networks (RNN), which are explained in Section 2.7.4.

For FNNs, the neuron output is calculated in two steps. First, the weighted sum $\mathbf{z}$ of the neuron outputs of the previous layer is calculated, with an additional neuron bias $\mathbf{b}$ as

$$z_j^{(i)} = \sum_k W_{j,k}^{(i)} h_k^{(i-1)} + b_j \tag{2.1}$$

where $z_j^{(i)}$ is the weighted sum for the neuron $j$ in layer $i$, $\mathbf{W}$ represents the weights between the layers $i$ and $i-1$, and $h_k^{(i-1)}$ is the output for the neuron $k$ in layer $i-1$. In order to increase the FNN's capacity to learn the highly nonlinear relationship of the input and the target output; a continuous, (sub-)differentiable, non-linear (or piece-wise linear) activation function (covered in Section 2.7.2) is applied to $z_j$ to get the neuron output $h_j$ as

$$h_j^{(i)} = \sigma(z_j^{(i)}) \tag{2.2}$$

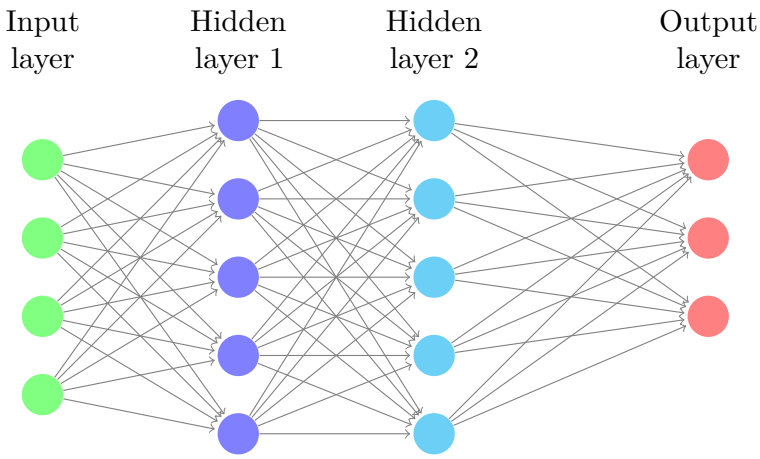


Figure 2.5: Feed-forward neural network with two hidden layers and a single output neuron. ©2015 IEEE.

where $\sigma$ is the activation function.

When the FNN is used to learn an acoustic model for SED, the FNN input is often a feature vector $\mathbf{x}$ that includes the acoustic features extracted from a single time frame. The input layer simply sets $\mathbf{x}$ as the output of the first layer, and no activation function is applied ($\mathbf{h}^{(0)} = \mathbf{x}$). Defining such an input layer supports the notation of Eq. 2.1 since $\mathbf{W}^{(1)}$ represents the weights between the input and the first hidden layer. The layer outputs are then calculated in a chain form through the input layer, the hidden layers and finally the output layer to obtain the network output

$$\hat{\mathbf{y}} = f(\mathbf{x}, \theta) \tag{2.3}$$

where $f$ represents the neural network based acoustic model, and $\theta$ represents the parameters of the network. The number of the neurons in the output layer is determined by the number of pre-defined sound events in the given SED task. A feed-forward neural network with its neuron connections is illustrated in Figure 2.5. For instance, this illustrated network can be used to learn an acoustic model which takes four acoustic features per time frame as input, and predicts if any of the three pre-defined sound events is present for a given frame. If the target output vector $\mathbf{y}$ is binary encoded (which is often the case for SED), then the weighted sum of each output layer neuron is passed through an activation function bounded between 0 and 1, so that the output of the network $\hat{\mathbf{y}}$ can be treated as the estimated probabilities of the sound events being present in the given frame.

The FNNs are often utilized with multiple hidden layers, *i.e.* in the form of DNNs, for machine hearing and detection tasks. DNNs offer superior expression capability for modeling the complex input - target output relationships by learning the high level representations in several layers of abstraction. One of the earlier examples of DNNs proposed for SEC is [26]. In this work, MFCCs and mel

spectrogram features have been used as input for training an FNN with two to five hidden layers in different experiments. It has been shown that FNN's sound event classification accuracy is superior to a GMM-HMM based classifier model by 5.5% absolute points. This method also includes the unsupervised pre-training of the DNN, which is based on minimizing a certain energy function defined by the output of a layer, and the pre-training is conducted layer by layer [48]. Lastly, an example of DNNs proposed for SED is [I], where DNNs with maxout activation functions [33] are proposed for polyphonic SED. This work is explained in more detail in Section 3.1.

### 2.7.2 Network training

The output of a neural network is calculated from its input and its parameters such as weights and biases. In order to get the desired target outputs as the network output for a given input, the network is trained using a certain optimization algorithm. Since the input cannot be updated by the network, the network training involves the optimization of the network parameters. The network optimization techniques mentioned in this subsection are utilized not only in feed-forward neural networks, but also in other types of ANNs which are explained later in the chapter.

The network parameters $\theta$ are often initialized with small, random values sampled from *e.g.* normal distribution. The network input $\mathbf{x}$ is passed through this initial network, and the network output $\hat{\mathbf{y}} = f(\mathbf{x}, \theta)$ is obtained. This process is called *forward propagation*. In order to estimate the proximity of the estimated output $\hat{\mathbf{y}}$ to the target output $\mathbf{y}$, the scalar loss $l(\hat{\mathbf{y}}, \mathbf{y})$ is calculated using a loss function. The common choices for the loss function in SED are the mean squared error and the cross entropy. An important point about the loss function is that it should be non-negative and differentiable everywhere, since the gradient based optimization algorithms are used to update the network parameters.

#### Gradient Descent Based Optimization Algorithms

During the network training, the parameters $\theta$ are updated based on the *gradient descent* algorithm in order to minimize the loss. For the gradient descent, the gradient $\nabla l$ with respect to the network parameters is defined as

$$\nabla l \equiv \left( \frac{\partial l}{\partial \theta} \right) \tag{2.4}$$

where $\theta$ includes the weights and biases in the network. The gradient $\nabla l$ is calculated efficiently using the *backpropagation* algorithm [87], where the gradient for each parameter is calculated using the chain rule of calculus starting from the output layer and moving towards the input layer. The change $\Delta l$ in the loss $l$ can be approximated as

$$\Delta l \approx \nabla l \cdot \Delta \theta \tag{2.5}$$

where $\Delta l$ represents the change in $l$ and $\Delta\theta$ represents the change in $\theta$. The idea of gradient descent is to update $\delta$ based on $\Delta\theta$, so that $l$ would decrease, which means $\Delta l$ would be negative. For this reason, we choose

$$\Delta\theta = -\eta\nabla l \tag{2.6}$$

where $\eta > 0$ is the learning rate. Equation (2.6) gives $\Delta l \approx -\eta||\nabla l||^2$, $\Delta l \leq 0$ and therefore $l$ will decrease. The weights and biases are updated as

$$\theta \leftarrow \theta + \Delta\theta = \theta - \eta\nabla l \tag{2.7}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}} \tag{2.8}$$

$$\mathbf{b} \leftarrow \mathbf{b} - \eta\frac{\partial l}{\partial \mathbf{b}} \tag{2.9}$$

**Stochastic Gradient Descent**

There are several different approaches involving the frequency of the network parameter update. One option is to calculate the loss for each training input $\mathbf{x}$, and then calculate the average loss and do the network update based on this average loss. This method is called *batch gradient descent*, and it requires all the training inputs to be forward propagated before any change is made on the network parameters. This is undesirable, especially in the beginning of the network training, when the acoustic model is very inaccurate on mapping the input to their target outputs, and the network requires frequent updates. Another (and more common) option is to use *stochastic gradient descent* (SGD), where a pre-defined amount of randomly selected training inputs are used to calculate an average loss, then the network parameters are updated based on this loss, and this is repeated until there are no training inputs left that have not been used in the training earlier. When all the training inputs are used for training, it is defined as an *epoch* of network training is complete. In that case, the inputs are again divided into randomly selected groups and the parameters are updated based on these new groups. The idea is that the randomly selected group of training inputs would provide a sufficiently good estimate of the true average loss, and the network parameters will converge faster through frequent updates.

While the earlier versions of SGD set the pre-defined amount of training inputs in a group to 1, it has been noticed that it is computationally more efficient to use multiple training inputs in a group. This method is called *mini-batch SGD*, and it is widely used especially the computational efficiency is of importance, for instance the machine learning tasks that include very large datasets. Besides, in the case of machine hearing, the input $\mathbf{x}$ is extracted from a single short time frame and updating the network parameters based on the loss and the gradients from a single frame may cause to noise in the parameter values. On the other

hand, using mini-batch SGD would lead to more stable parameter updates as the average loss over multiple time frames is utilized.

With the SGD optimization, the network is trained for several epochs (typically 100 to 500 for SED tasks). As each epoch would result in different network parameters due to the updates, the network would produce a different loss value for the same input. After many epochs, the updates on the parameters become very small, the network parameters *converge* and the training is stopped.

#### Momentum

With the introduction of DNNs and large datasets, the efficiency and the speed of SGD based network training has started to become a problem. DNNs typically have much more parameters (often in the million range) compared to the earlier neural networks where SGD has been used. Calculating the gradients and doing the parameter update therefore takes a significant amount of time. In addition, for a fixed learning rate $\eta$, the amount of update is considerably smaller for the parameters in lower layers than the higher layers due to the *vanishing gradient problem* [49]. Besides, large datasets, which are beneficial to train a complex network with many parameters such as deep neural networks, may take a long time to process. These factors encouraged neural network researchers to come up with more efficient optimization techniques, *i.e.*, techniques that lead to the network converging in a shorter amount of time (or smaller amount of epochs). One of these techniques is called *momentum* [84], which is actually an older method that has been resurrected in the deep learning era.

Momentum is a technique for accelerating the gradient descent that accumulates a velocity vector of parameter updates towards the consistent reduction of the loss [102]:

$$\begin{aligned} v_t &\leftarrow \mu v_{t-1} - \eta \nabla l_t \\ \theta_t &\leftarrow \theta_{t-1} + v_t \end{aligned} \tag{2.10}$$

where $\mathbf{v}$ is the velocity vector for parameter $\theta$, $t$ is the epoch index, and $\mu$ is the momentum coefficient. The advantage of momentum is that it speeds up the learning process. For instance, if $\nabla l_t$ is positive, it suggests that $\theta_t$ should be decreased to minimize the loss (Eqn. 2.7). With the momentum technique, unlike SGD, the update of $-\eta \nabla l_t$ is not directly applied to $\theta$, but accumulated in a velocity vector. If the gradient for the earlier epochs also have the same sign as $\nabla l_t$ (positive for this example), then the amplitude of the update will be larger than $\eta \nabla l_t$ due to the accumulated velocity, and this means a bigger step towards minimizing the loss. The disadvantage of the momentum is that, it introduces an additional hyper-parameter $\mu$ which controls the amount of momentum ($\mu = 0$ simply reduces to SGD). This adds the burden of finding an optimal initial value for this hyper-parameter $\mu$ (more about hyper-parameter fine tuning in Section 2.7.2).

### Adam

Learning rate $\eta$ is one of the most important hyper-parameters for neural network optimization, since it directly affects the amount of update for every network parameter. For SGD, the learning rate is fixed throughout the training, which makes the optimization process highly dependent of the selected value of the learning rate. On the other hand, it is more desirable to have larger updates in the first few epochs of the training when the network is not familiar with the task and often makes large mistakes on the output estimation, and smaller updates towards the final epochs when the network requires only small fine-tuning of parameters before the convergence. In this direction, optimization algorithms with adaptive learning rates have been recently proposed and widely used in deep learning. One of the most popular among these techniques is Adam [56]. The algorithm for Adam is given in Algorithm 1. While momentum utilizes only the first moment of the gradient, Adam utilizes both first and second moments; and it applies bias correction to avoid relatively high bias early in training, especially for the second moments [32]. The empirical robustness of Adam algorithm to the hyper-parameters of the deep networks has recently made it a popular choice for deep learning researchers.

**Algorithm 1** Adam algorithm.

**Require:** learning rate $\eta$, exponential decay rates $\rho_1$ and $\rho_2$, and small constant $\epsilon$ for numerical stability
$s \leftarrow 0$ ▷ Initialize first moment
$r \leftarrow 0$ ▷ Initialize second moment
$t \leftarrow 0$ ▷ Initialize timesteps
**while** $\theta$ **not** converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \left(\frac{\partial l}{\partial \theta_{t-1}}\right)$ ▷ update gradient of the loss $l$ w.r.t. the parameter $\theta$
  $s \leftarrow \rho_1 s + (1 - \rho_1) g_t$ ▷ update biased first moment estimate
  $r \leftarrow \rho_2 r + (1 - \rho_2) {g_t}^2$ ▷ update biased second moment estimate
  $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$ ▷ compute bias-corrected first moment estimate
  $\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$ ▷ compute bias-corrected first moment estimate
  $\theta_t \leftarrow \theta_{t-1} - \eta \frac{\hat{s}}{(\sqrt{\hat{r}}+\epsilon)}$ ▷ Update parameter
**end while**
Return $\theta$

### Activation Functions

For the neuron $j$ in layer $i$, the hidden unit output $h_j^{(i)}$ is calculated by applying an activation function over the weighted sum $z_j^{(i)}$ of the neuron outputs for the layer $i-1$. Since the activation functions are applied element-wise for the weighted

sum $z$ for each hidden unit in a layer, the layer and neuron indices are omitted below.

*Logistic sigmoid activation function* is evaluated as

$$h = \frac{1}{1 + \exp(-z)} \tag{2.11}$$

and it is illustrated in Figure 2.6. It is a continuous and differentiable function which is desirable for gradient based network optimization algorithms. Logistic sigmoid function saturates to 1 when the weighted sum $z$ is a large positive value, and saturates to 0 when $z$ is a large negative value, which limits the speed of gradient-based learning for those cases. In the case of SED, logistic sigmoid is often used as the activation function not only for the hidden layer neurons but also for the output layer neurons, since their outputs (bounded between $[0, 1]$) can be treated as the event presence probabilities.

*Rectified linear units* (often shortened as ReLUs) utilize the activation function

$$h = \max\{0, z\}. \tag{2.12}$$

ReLU is a piecewise-linear and sub-differentiable function. The gradient of ReLU is equal to 1 when the weighted sum $z$ is positive, and 0 when $z$ is negative. Even though theoretically the gradient of ReLU is undefined at $z = 0$, empirically it does not create problems. The ReLU input $z$ is hardly ever equal to exactly 0, and the software implementations of neural network training often return either 0 or 1 as the gradient if that is the case, and any number between [0,1] is a sub-gradient of ReLU at $z = 0$. The advantage of ReLUs over logistic sigmoid is that ReLU provides larger gradients than logistic sigmoid whenever the weighted sum is positive (ReLU gradient is 1 while logistic sigmoid gradient is bounded between $[0, 0.25]$(see Figure 2.6)). The disadvantage of ReLU is that their parameters do not get updated when the weighted sum is negative, since the gradient is equal to 0. In the case of SED, ReLUs are often used as the hidden layer activations, especially for the convolutional layers.

*Maxout units* [33] provide a generalized version of ReLUs by applying the max operation over the groups of $k$ weighted sums:

$$h_i = \max_{j \in \mathbb{G}^{(i)}} z_j \tag{2.13}$$

where $\mathbb{G}^{(i)}$ is the set of indices into the inputs for group $i$, $\{(i-1)k + 1, ..., ik\}$. The activation function of each group can be treated as a piece-wise linear, convex function with $k$ pieces, which is learned through the neural network training [32]. In the case of supervised learning for SED, maxout units are used only as the hidden layer activation functions, since the output layer activations must be bounded in order to match with the target outputs.

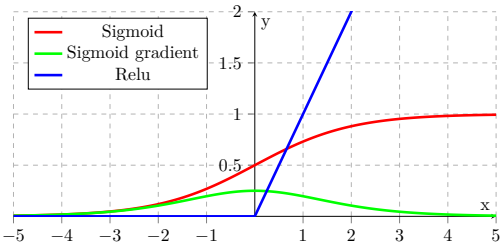


Figure 2.6: Logistic sigmoid and ReLU activation functions.

#### Hyper-parameter Fine Tuning

There is a certain degree of mismatch between the network training and evaluation procedures for SED. The network is trained to minimize a certain, differentiable loss value, however it is evaluated against a non-differentiable performance metric which may not be in positive correlation with the loss value for every epoch. This is often the case for SED, where cross-entropy is mostly used as the loss function and F1 score is used as the evaluation metric. Besides, the network's accuracy on the training set may not always translate to the test set due to overfitting [39]. Therefore, in order to determine the optimal network parameters based on the training, the network is evaluated on the *validation set* after each epoch and the parameters from the epoch with the best validation set performance are used as the final model parameters. Validation set is often a small portion of the training set (around 10-20%) that is not used for training the network, but only for the validation of the network performance during training.

While determining the values for certain hyper-parameters such as the number of hidden units and the number of layers, grid search is often used. With grid search, a certain range of values are selected for each hyper-parameter, and each combination of the values are utilized in different experiments. The best performing set of hyper-parameters are determined based on the validation set performance. By not using the test set performance for the hyper-parameter selection, the possibility of overfitting the hyper-parameters to the given test set is avoided.

### 2.7.3 Network Regularization

There are several techniques proposed to improve the accuracy, generalization over the unseen examples and the speed of convergence for neural network training. These techniques are grouped under *network regularization* techniques. Two of these techniques that are used for some of the publications in this thesis are explained below.

### Dropout

With the dropout algorithm [96], the activations of the hidden units are dropped out (*i.e.* set to 0) with a certain probability (often around 0.1 to 0.25) and therefore these units do not have any effect on the output of the network. During training, the probability of dropping out each hidden unit activation is randomly sampled for every epoch. This reduces the unit co-adaptation (*i.e.* the output of a unit being dominated by a single or a few of its input connections), and training with dropout efficiently approximates training several networks with different initial parameters and averaging their outputs to be used as the final output of the system.

### Batch Normalization

Due to shrinking (or, less commonly, exploding) gradients problem, the distribution of the activations of each hidden layer becomes very diverse for deeper networks, which slows down the learning as each layer is updated with the same learning rate. Batch normalization [52] essentially involves the normalization of the hidden unit activations at each layer to zero mean and unit standard deviation. For mini-batch SGD, the mean and the standard deviation of each hidden unit activation is calculated as the average of the activations of the mini-batch examples. During training, the activation matrix $\mathbf{H}$ for a mini-batch examples is normalized to $\mathbf{H}'$ as

$$\begin{aligned} \boldsymbol{\mu} &= \frac{1}{m}\sum_i \mathbf{h}_i \\ \boldsymbol{\sigma} &= \sqrt{\delta + \frac{1}{m}\sum_i (\mathbf{H}-\boldsymbol{\mu})_i^2} \\ \mathbf{H}' &= \frac{\mathbf{H}-\boldsymbol{\mu}}{\boldsymbol{\sigma}} \end{aligned} \tag{2.14}$$

where $\mu$ and $\sigma$ are the average mean and standard deviation for the mini-batch examples for a given hidden unit, and $\delta$ is a small positive constant to avoid divide-by-zero errors when $\sigma = 0$. The normalized $\mathbf{H}'$ can be further scaled as $\mathbf{H}'' = \gamma\mathbf{H}' + \beta$, where $\gamma$ and $\beta$ are the learned network parameters. This reparametrization can be useful to increase the expressive power of the network by letting the network learn the optimal mean and the standard deviation for the hidden unit activations at each layer. While using the trained network, $\mu$ and $\sigma$ are replaced with their averages over the whole set of examples instead of mini-batches.

## 2.7.4 Recurrent Neural Networks

Recurrent neural networks (RNN) [87] are a specialized family of ANNs that are beneficial to use with sequential data. While DNNs use only the current input $\mathbf{x}_t$ at time frame $t$ and the network parameters $\theta$ to calculate the activation $h_t$,

RNNs also utilize the activations from the earlier time frames $h_1, h_2, ...h_{t-1}$. In the case of SED, the input features from the consecutive time frames are naturally correlated, as the time frames are on the order of 20-60 ms length and the sound events are typically present for many consecutive frames. Therefore, being able to utilize the activations from the earlier frames makes RNNs a more suitable machine learning algorithm for SED than DNNs.

RNNs are used to map a sequence of input vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]$ to a sequence of target output vectors $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T]$ with the same sequence length $T$. While there are other types of RNNs that are used to map input - target output pairs with different sequence length (*e.g.* in machine translation [8]), these methods are out of scope of this thesis. In the case of SED, the sequence length $T$ is often selected in the range of 2 to 10 seconds, and this is based on the average time a human takes to identify sounds in real life [11, 92].

For RNN, the $i^{\text{th}}$ hidden layer output $\mathbf{h}_t$ for the input at timestep $t$ can be written in vector multiplication form as

$$\begin{aligned} \mathbf{z}_t^{(i)} &= \mathbf{W}^{(i)}\mathbf{h}_t^{(i-1)} + \mathbf{W}^{*(i)}\mathbf{h}_{t-1}^{(i)} + \mathbf{b} \\ \mathbf{h}_t^{(i)} &= \sigma(\mathbf{z}_t^{(i)}) \end{aligned} \tag{2.15}$$

where $\mathbf{W}^{(i)}$ is the weight matrix between layers $i-1$ and $i$, $\mathbf{W}^{*(i)}$ is the feedback weight matrix and $\sigma(\cdot)$ is the activation function. Comparing Eq. 2.15 with the DNN hidden layer output calculation in Eq. 2.1 and 2.2 would show that recurrent layers have an additional weight matrix which is used to determine the contribution of the output at the previous timestep $t-1$ on the output for the current timestep $t$.

### Bidirectional Recurrent Neural Networks

RNNs process the input sequence in a single direction, *i.e.* they utilize the information from the previous timesteps to calculate the output for the current timestep. On the other hand, in some cases, it may be also beneficial to process the input sequence in the opposite direction. An example for such a case occurs in ASR, where the articulation of the current phoneme may be dependent on the next phonemes as a result of linguistic characteristics of some languages. For SED, there may be certain sound events that share similar acoustic characteristics with another sound event class in the beginning of the event. Using the outputs from the future timesteps as a feedback would be helpful to distinguish these sound event classes at the beginning of the event.

Bidirectional RNNs [94] are a special type of RNNs whose hidden units are split into two groups to process the information in both directions. The hidden layer output of a bidirectional RNN layer thus consists of both the forward output $\mathbf{h}_t$, and the backward output $\mathbf{g}_t$, which is calculated similarly to Eq. 2.15 with the feedback from the frame $t+1$ instead of $t-1$. The forward $\mathbf{h}_t$ and the backward

output $\mathbf{g}_t$ are then combined by simply concatenating to obtain a single output vector at each timestep $t$.

The disadvantage of bidirectional RNNs is their non-causality, *i.e.* the inputs from the future timesteps are assumed to be available. While this may create problems in real-time systems, the typical sequence lengths used in most SED methods (a few seconds) can be tolerated by introducing a short time delay between the input and the system output.

### Gated Recurrent Neural Networks

The most important problem with the RNNs as the way they were initially introduced is that, although in theory there is no limit to the length of the temporal context that can be modeled with an RNN layer, in practice it is challenging to model the long-term dependencies with the RNNs. The reason is that the contribution of the outputs from the past timesteps decay exponentially over time due to vanishing gradients problem [49].

As a solution, gated recurrent layer methods such as gated recurrent units (GRU) [7] and long-short term memory networks (LSTM) [50] were introduced. The units for both methods are called *cells* whose output is calculated as a combination of multiple gate activations. These gates are called external input, forget, and output gates for LSTM, and update and reset gates for GRU. The gates are composed of weights and an activation function. Each cell includes a cell state, which consists of the accumulated information from the previous timesteps. During training, the gate weights learn by which proportions to combine the cell state and the input for the current timestep to produce the gated unit output for the current timestep.

The long term temporal modeling of the sound events can be optionally handled through the GRU layers. GRU layers control the information flow by the gated unit structure including the reset gate $r_t$ and the update gate $u_t$. For frame $t$, the total output of the GRU layer is a linear interpolation of the output for the previous frame $h_{t-1}$ and the candidate output for the present frame $\hat{h}_t$ as

$$h_t = u_t \cdot h_{t-1} + (1 - u_t) \cdot \hat{h}_t. \tag{2.16}$$

Candidate output $\hat{h}_t$ is calculated through $h_{t-1}$, the layer input $x_t$ and the reset gate $r_t$. When reset gate is closed ($r_t = 0$), the contribution from the previous frame output $h_{t-1}$ is discarded. This implies that the the higher level representations for the current (which is used as input $x_t$ to GRU layer) are significantly different from the earlier frames, and the cell state needs to be updated to reflect this.

When the GRU layer outputs are fed as input to a feed-forward output layer, the contributions of GRU's previous and candidate outputs to the network outputs,

$c_{t-1}$ and $\hat{c}_t$ respectively, can be computed as

$$\begin{aligned} c_{t-1} &= w \odot (u_t \cdot h_{t-1}) \\ \hat{c}_t &= w \odot ((1 - u_t) \cdot \hat{h}_t) \end{aligned} \tag{2.17}$$

where $w$ is the weight vector that connects GRU layer and the feed-forward layer, and $\odot$ denotes element-wise multiplication. The total output $o_t$ of the feed-forward layer (which is also the output of the whole network) is calculated as $o_t = c_{t-1} + \hat{c}_t$. The output is then dominated by the candidate output at the beginning and the end of the target sound event, and both candidate and previous outputs are as small as possible when the target sound event is not present.

The idea behind the LSTM and GRU is very similar, which is to allow the relevant information from the previous timesteps to be stored in the cell state, and control the cell state through the gates that learn which information is relevant for the given task. The main difference is that GRUs combine the forget and external input gates of the LSTM in a single gate called update gate, hence has less parameters compared to LSTM. Currently, both methods are widely used in classification tasks with sequential inputs, such as ASR [34], music onset detection [22], and machine translation [7]. In [77], bidirectional LSTMs improved the state-of-the-art DNN method for polyphonic SED by 5.6% and 1.6% absolute points in frame-wise and one-second block-wise F1 score, respectively.

### 2.7.5 Convolutional Neural Networks

For the traditional ANN methods such as FNNs, every input feature is connected to every unit of the hidden layer through network weights. This may be deemed inefficient especially for the cases when the input features contain a spatial structure, because the features that represent far apart regions in the spectral domain are treated the same way with this method. Convolutional neural networks (CNN) [60, 61] address this problem by estimating their outputs from a local region of the input through convolution with a kernel.

CNNs are a special type of ANNs that are able to exploit the spatial structure information in the input. CNNs offer to improve the modeling efficiency of the traditional FNNs through three main properties: local representations, parameter sharing, and pooling.

Given the input matrix $\mathbf{X}$ and a two-dimensional kernel $\mathbf{K}$, the convolution kernel output $\mathbf{H}$ is calculated as

$$\mathbf{H}_{i,j} = \sigma(b^{(K)} + \sum_m \sum_n \mathbf{X}_{i-m,j-n}\mathbf{K}_{m,n}) \tag{2.18}$$

where $\sigma$ is the activation function (often selected as ReLU) and $b^{(K)}$ is the bias for the kernel $\mathbf{K}$. The kernel size is often much smaller than the input size, and the kernel is slided over the input to extract local representations. Therefore,

rather than connecting each input feature to a hidden unit in the next layer, convolutional layer parameters are *shared* among the input features, and the kernel parameters are trained to learn local patterns that can be found in any part of the given input. This is especially favorable in the case of SED, where a sound event should be detected from *e.g.* a given spectrogram regardless of the temporal position of the event in the spectrogram. In addition, even though convolutional layers typically consist of tens or hundreds of kernels, this parameter sharing approach is often far more memory-efficient than the fully-connected FNN method due to less number of weights.

Convolutional layers are often followed by a max-pooling layer, which divides each convolution kernel output into equal regions of pre-defined size, and outputs the maximum value in each region. This provides a downsampled representation of the kernel output, and it is especially useful when the spectral position of a pattern specific to a class may exhibit small shifts. This is often the case in SED, where the sound event classes are rather broadly defined. For instance, the examples of the same class (*e.g.* dog bark) may have slightly shifted representations in the frequency domain (chihuahua bark vs. pitbull bark), while the underlying spectral pattern is similar. If the frame-level detection is required, max-pooling in the CNNs should be only applied in the frequency axis, because max-pooling in time domain would reduce the temporal resolution of the network outputs.

The outputs of each convolution kernel (often called *feature map*) are stacked in a separate dimension, so the output of a convolutional layer is three dimensional. Convolutional layers are often used as high-level feature extractors in ANNs, and the network output is obtained from a different layer that would take the convolutional layer output as its input. If the convolutional layer output is fed to a different ANN layer that requires its input to be vectors (such as RNNs), the features extracted by each feature map are concatenated in the frequency axis, so that a single feature vector is obtained at each timestep.

In the case of SED, CNNs are suitable as the classifiers since the input representation is often two dimensional, *e.g.* mel spectrogram. On the other hand, the limited temporal modeling capabilities have been a drawback for CNNs, and the initial interest on CNNs has quickly switched into methods that combine CNNs and the algorithms with long term temporal modeling, such as CRNNs with GRU/LSTM recurrent layers. On the other hand, CNNs are still a useful method for the tasks that do not require very high temporal resolution, such as SEC. In [37, 82, 91, 103], CNNs with log mel spectrogram input and dropout regularization were proposed for single label SEC over 4 to 12 second chunks of real-life environmental recordings. While the framework is similar for all three methods (convolutional layers followed by max pooling and feed-forward layers with ReLU activation on top and softmax output layer), each proposed method offers various extensions such as delta input features as a separate channel [82], data augmentation through methods such as time stretching, pitch shifting, dy-

namic range compression [91] and equalized mixture data augmentation [103], and multiple instance learning [103]. The methods are evaluated over established SEC datasets such as ESC [83] and UrbanSound8K [92] and are shown to outperform methods such as unsupervised feature learning through dictionary learning [90] and DNN-HMM.

## 2.8 Evaluation of Sound Event Detection Methods

In order to effectively measure the improvements offered by a proposed scientific method, systematic evaluation is crucial. In the case of SED, systematic evaluation involves measuring the performance of the proposed method based on the performance metrics that are selected to reflect the real-life applicability of such methods. During the evaluation, using a set of commonly used, benchmark performance metrics makes it easier to interpret and measure the progress that is offered with the proposed methods.

In addition, the dataset that has been selected to train and evaluate the SED method should include a diverse set of examples for the given SED task, so that the performance evaluated on such a dataset would reflect the performance of the method in real-life conditions. The SED dataset can be either collected in real-life environments and annotated manually, or it can be synthesized using individual sound events and mixing them to sufficiently resemble the acoustic characteristics of a real-life environment. The advantage of creating synthetic SED datasets instead of recording from real-life environments is that the onset and offset annotations are much more accurate, which makes the evaluation more fair and the machine learning method to be trained with more accurate target outputs. The disadvantage is that it is challenging to create sound event mixtures in a controlled way that the mixture would resemble a recording from a real-life environment.

The performance metrics and the datasets used in the publications in this thesis are explained below.

### 2.8.1 Performance Metrics

Evaluation of SED methods is done by computing a performance metric based on the binary detection outputs and the target outputs for the given test (or evaluation) set. The most commonly used performance metrics for SED can be listed as precision, recall (also called true positive rate), F1 score, error rate and the area under receiver operating characteristic (ROC) curve. A more detailed study on these metrics can be found in [70].

The SED performance metrics utilize a set of common intermediate statistics, and the final performance is calculated based on the metric and these intermediate statistics. These statistics include the count of true positives (TP), true

Table 2.1: Definitions of TP, TN, FP and FN based on the detection and target outputs for a given segment or event.

| Active in the... | TP | TN | FP | FN |
|---|---|---|---|---|
| Detection output | ✓ | | ✓ | |
| Target output | ✓ | | | ✓ |

negatives (TN), false positives (FP) and false negatives (FN), which are obtained based on Table 2.1. In terms of time resolution, the SED metrics can be calculated either segment-wise or event-wise. For the segment-wise metrics, the intermediate statistics are calculated among all the frames inside the given segment. For instance, if a class is detected as present at any frame in the given one second segment, and the target output for that class is 1 at any frame inside the segment, then the detection is counted as TP, regardless of whether the detection and the target output is active at the same time frame. The intermediate statistics are accumulated over all the segments and the sound event classes for the given test set. The performance metric can be then calculated for each class separately from these statistics, or it can be calculated globally by accumulating the statistics over the classes.

**Precision**

Precision shows the ratio of the correctly detected examples among all the detections made by the system. It is calculated as

$$P = \frac{TP}{TP + FP} \tag{2.19}$$

**Recall**

Precision shows the ratio of the correctly detected examples among all the examples. It is calculated as

$$R = \frac{TP}{TP + FN} \tag{2.20}$$

**F1 Score**

F1 score is the harmonic mean of precision and recall, and it is calculated as

$$F1 = \frac{2 \cdot P \cdot R}{P + R}. \tag{2.21}$$

Currently, F1 score is the most commonly used performance metric for SED, and it has been used as the official metric for DCASE 2016 [72] and 2017 [71] SED challenges. The advantages of the F1 score are that it can be applied on both single label and multi-label classification tasks, and it takes into account equally the precision and the recall of the system. The disadvantage is that true negatives are not used in the calculation of the F1 score.

### ROC Curve Based Metrics

Unlike the previously mentioned metrics, ROC curve is not obtained by using a single detection threshold value for binarizing the detection probabilities, but it is obtained by calculating the true positive rate (TPR) against the false positive rate (TPR) at various threshold values. TPR is equal to recall, and FPR is calculated as

$$FPR = \frac{FP}{FP + TN} \tag{2.22}$$

The *area under ROC curve* is indicative of the system performance, since a high performance method should have greater TPR than FPR at each threshold value.

Another metric based on the ROC curve is the *equal error rate* (EER). EER simply has the value of the curve where TPR is equal to 1-FPR. Therefore, unlike the area under ROC curve, it is dependent on only a single point of the curve.

### Error Rate

Error rate ($ER$) is calculated through three values of intermediate statistics that measures the mistakes made with the detections: insertions ($I$), deletions ($D$), and substitutions ($S$). While $I$ and $D$ simply correspond to $FP$ and $FN$, respectively, $S$ is an inter-class measure that occurs when the wrong class is detected instead of another class for a given segment. Therefore the mistake is not counted twice (as $I$ for one class and $D$ for another class), and it is counted as a single $S$. The error rate is then calculated as

$$ER = \frac{S + I + D}{N} \tag{2.23}$$

where $N$ is the number of positively labeled examples.

ER can be calculated both segment-wise and event-wise. For segment-wise ER, the intermediate statistics are accumulated over each segment and the ER is calculated over these accumulated values. For event-wise ER, the statistics are obtained for each event separately, and then accumulated over all the events. While calculating the event-wise ER, the time alignment at the onset and the offset between the detection and the target output is often tolerated within a collar, which is often selected around 100-500 ms depending on the desired time resolution for the given event-based metric and considering the inexact labeling of the data [70].

## 2.8.2 Datasets

The datasets used in the experiments for this thesis are explained below in detail and summarized in Table 2.2.

Table 2.2: SED datasets used for experiments in this thesis.

| Dataset | Environment | Annotation level | Total length (mins) | #classes |
|---|---|---|---|---|
| TUT-SED 2009 | real-life | frame | 1133 | 61 |
| TUT-SED synthetic 2016 | synthetic | frame | 566 | 16 |
| TUT-SED 2016 | real-life | frame | 78 | 18 |
| CHiME-Home | real-life | clip (4 s) | 186 | 7 |
| BAD 2017 | real-life | clip (10 s) | 4052 | 1 |
| DCASE 2017 rare SED | synthetic | frame | user-defined | 3 |

**TUT-SED 2009**

TUT-SED 2009 dataset includes 8-14 recordings per 10 real-life acoustic scenes, with a total of 1133 minutes. The recorded real-life environments are: basketball game, beach, inside a bus, inside a car, hallway, office, restaurant, shop, street and stadium with track and field events. A total of 61 classes (cat meowing, applause, traffic, yelling etc., full list of classes can be found in [I]) were defined, and one extra class is reserved for unknown (or very rare) events. The average polyphony in a given time frame is 2.53. The sound events in TUT-SED 2009 were annotated manually. TUT-SED 2009 dataset was first introduced in [41]. Currently, the dataset is not publicly available due to copyright issues. A demo using the samples from the dataset and the predicted outputs with various SED methods are available at [2].

**TUT-SED synthetic 2016**

TUT-SED synthetic 2016 dataset includes synthetic, polyphonic sound event mixtures created by mixing isolated sound event samples from 16 everyday classes. 994 individual sound event samples are used to create 100 mixtures, each with length around 5-7 minutes, and a total of 566 minutes. Mixtures were created by randomly selecting a segment of length 3-15 seconds from an individual sample, pre-processing the segment by start-end silence trimming and peak normalization, and mixing the sound event segment with the mixture at a randomly selected onset. The average polyphony level in one-second blocks is above 1 for over 48% of the dataset. TUT-SED synthetic 2016 dataset was first introduced in [III]. A more detailed explanation on the dataset creation procedure can be found in the supporting website of this article [3]. The dataset is publicly available through the supporting website.

**TUT-SED 2016**

TUT-SED 2016 is a real-life SED dataset with recordings from residential area and home environments [69]. Each recording is captured in a different location (*i.e.*,

[2]http://arg.cs.tut.fi/demo/CASAbrowser/

[3]http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/tut-sed-synthetic-2016

different streets, different homes) to increase the variability of the sound event classes present between the recordings. A 3-5 minute audio recording is provided for each location, and the total amount of audio is around 78 minutes. The number of annotated sound event classes are seven for residential area recordings and 11 for home recordings. TUT-SED 2016 dataset is publicly available online [4].

### CHiME-Home

CHiME-Home dataset [24] consists of 4-second chunks of recordings from home environments. The pre-determined sound event classes are child speech, adult male speech, adult female speech, video game / TV, percussive sounds, broadband noise and other identifiable sounds. The annotations for CHiME-Home include which sound events are present at any point in each four second chunk. This differs from other SED datasets, where the annotations include the onset and offset time for each sound event in a given audio recordings. However, since the audio chunks in CHiME-Home are the non-overlapping parts of continuous recordings, it can be regarded as an SED dataset where the maximum time resolution of the detections is implicitly set as four seconds. CHiME-Home has been used as the official dataset for the audio tagging task in DCASE2016 challenge [72], and it is publicly available online [5].

### Bird Audio Detection 2017

The Bird Audio Detection (BAD) challenge 2017 dataset [101] consists of 10 second-long audio recordings collected from real-life environments by e.g. crowd-sourcing from a mobile app for bird audio classification, and individual recordings that are freely available online through content sharing websites. The recordings are collected from a wide range of indoor and outdoor locations with varying environmental conditions. The recordings are annotated for whether a bird sound is present at any given time during the 10-second recording. While the challenge is specified as "detection", it can be regarded as "classification" within the terminology of this thesis, since the task is to label the whole 10-second recording instead of detecting the onset and offset. The total length of recordings is around 68 hours. The dataset is publicly available online [6].

### DCASE 2017 challenge, rare SED

DCASE 2017 challenge rare SED dataset consists of 30-second recordings of real-life acoustic scenes as the background, and each recording may or may not include one of the three target sound events (baby cry, glass break, gun shot). The target sound events are mixed with the background at random onset times

[4] https://zenodo.org/record/45759
[5] https://archive.org/details/chime-home
[6] http://machine-listening.eecs.qmul.ac.uk/bird-audio-detection-challenge/#downloads

at various SNR levels such as -6, 0 and 6 dB. The onset and offset times for each target sound event is labeled. Each recording can include maximum one target sound event, and the mean duration for target sound events is 2.25 seconds. The dataset is publicly available online [7].

[7] https://zenodo.org/record/401395

# 3 Deep Neural Networks for Polyphonic Sound Event Detection

In [I] and [II], deep neural networks (DNN) have been proposed for polyphonic sound event detection on realistic everyday environments. Sound events often occur simultaneously in real-world environments. Therefore, polyphonic detection is essential to get a robust SED system that would provide high performance in complex auditory scenes such as real-world environments.

## 3.1 Polyphonic Sound Event Detection Using Multi-label Deep Neural Networks

Since the introduction of Deep Belief Networks [48] in 2006, deep learning based methods have provided state-of-the-art results in many classification tasks such as image recognition [40, 57], speech recognition [34, 88] and machine translation [8]. However, at the time of writing [I], DNNs had still not been fully explored for SED tasks.

The main motivation of [I] has been the findings of [26], where DNNs with unsupervised pre-training have outperformed a conventional GMM-HMM based classifier for single label SEC on isolated sound events. Encouraged by this result, we experimented with DNNs for polyphonic SED on realistic everyday environments. The capability of DNNs to utilize their neurons in separate groups to distinguish different sound events make them a viable option as a classifier for polyphonic SED [46].

### 3.1.1 Method

The proposed method is composed of two stages: sound representation and multi-label classification. In the sound representation stage, the audio recordings from everyday environments are divided into 50 ms time frames that overlap 50%, and log mel spectrogram features are calculated. Mel spectrogram is a matrix that

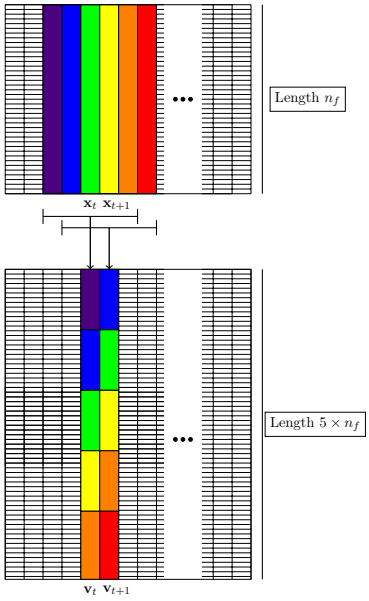


Figure 3.1: Context window $\tilde{\mathbf{x}}_t$ of $\mathbf{x}_t$ with its 2 adjacent frames on both sides.

consists of log mel band energy feature vectors for consecutive time frames. Since DNNs can take only single dimensional features as input, log mel spectrogram cannot be directly utilized as the network input. Instead, the log mel band energies $\mathbf{x}_t$ for the time frame $t$ can be used as a single training instance for the network, however this would mean that the temporal information that can be utilized by the network is limited to a single frame.

In order to extract the time dynamic property information of the signal, context windowing method is used. The log mel band energies for the consecutive time frames are concatenated in single dimension. The resulting feature vector $\tilde{\mathbf{x}}$ has a dimension of $(2 \times N_{\text{adj}} + 1) \times N_{\text{f}}$ where $2 \times N_{\text{adj}}$ is the total number of adjacent (past and future) frames concatenated with $\mathbf{x}_t$ and $N_{\text{f}}$ is the number of features extracted from the short time frame. Context windowing has also been used in other studies on sound classification [26, 51] and it is visualized in Figure 3.1.

The target output $\mathbf{y}_t$ for each frame $t$ is a binary vector whose entries encode the presence of the sound event classes. Each class present in the frame is labeled with 1 and 0 otherwise. The input - target output pair $[\tilde{\mathbf{x}}_t, \mathbf{y}_t]$ represents a single training instance for the DNN.

In order to learn a complex function $f$ that can map the input features $\mathbf{x}_t$ and the target output $\mathbf{y}_t$, a DNN is trained. Due to their hierarchical layer structure and high expression power, DNNs can model highly nonlinear relationships by learning a higher level representation at each layer. This is especially useful for the cases when the input-output relationship is hard to express algorithmically, such as for polyphonic SED.

A DNN with two hidden layers of 800 units is utilized in [I]. Each hidden unit has maxout activation (see Section 2.7.2). The output layer neurons have logistic sigmoid activation (bounded between 0 and 1), and the number of the neurons in the output layer is equal to the number of sound event classes. The network output $\hat{\mathbf{y}}_t$ is regarded as the class presence probability vector. The network is trained using mini-batch stochastic gradient descent (SGD) to minimize the cross-entropy loss $l(\mathbf{y}_t, \hat{\mathbf{y}}_t)$ between the estimated output $\hat{\mathbf{y}}_t$ and the target output $\mathbf{y}_t$, which is formulated as

$$l(\mathbf{y}_t, \hat{\mathbf{y}}_t) = -[\mathbf{y}_t \cdot \log \hat{\mathbf{y}}_t + (1 - \mathbf{y}_t) \cdot \log (1 - \hat{\mathbf{y}}_t)]. \tag{3.1}$$

The network is trained iteratively until the loss value for the validation set converges.

In the usage stage, the class presence probabilities are binarized using the unbiased, constant threshold of 0.5 to get the binary prediction vector $\mathbf{z}_t$. Sound events typically take minimum 1-2 seconds, which corresponds to 40-80 time frames with the given frame length and the overlap settings. By combining the binary predictions for the consecutive time frames, the onset and the offset for each sound event can be determined.

The initial experiments in [I] have shown that there can be some abrupt changes between predictions for consecutive time frames. The reason for this can be explained with the difference between the time resolution for the feature extraction (high) and the manual annotation of the sound event labels (low). This causes some of the frames to be labeled with sound events that are not present. In order to smooth the network predictions in the usage stage, a median filtering based post-processing method is proposed in [I]. For each frame, the post-processed predictions $\hat{\mathbf{z}}_t$ are obtained by taking the median of the binary predictions in a 10-frame window as

$$\hat{\mathbf{z}}_t = \text{median}(\mathbf{z}_{(t-9):t}). \tag{3.2}$$

The sliding window is applied continuously over the binary predictions for each frame with step 1. This post-processing approach is illustrated in Figure 3.2.

### 3.1.2 Evaluation

The method in [I] is evaluated using TUT-SED 2009, a dataset with around 20 hours of real-life audio recordings collected from 10 different everyday contexts (see Section 2.8.2 for more detail). The sheer size of the dataset makes it a valuable resource the investigate the robustness of the proposed method in the real-life environments. There are 61 pre-determined sound event classes. The proposed method is evaluated using five-fold cross-validation.

The evaluation metric is average F1 score for non-overlapping one second blocks. The proposed method is compared with [43], where the audio recordings are

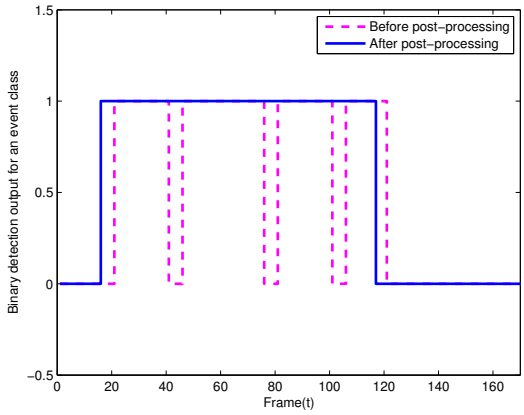


Figure 3.2: Post-processing with a sliding window of median filter. ©2015 IEEE.

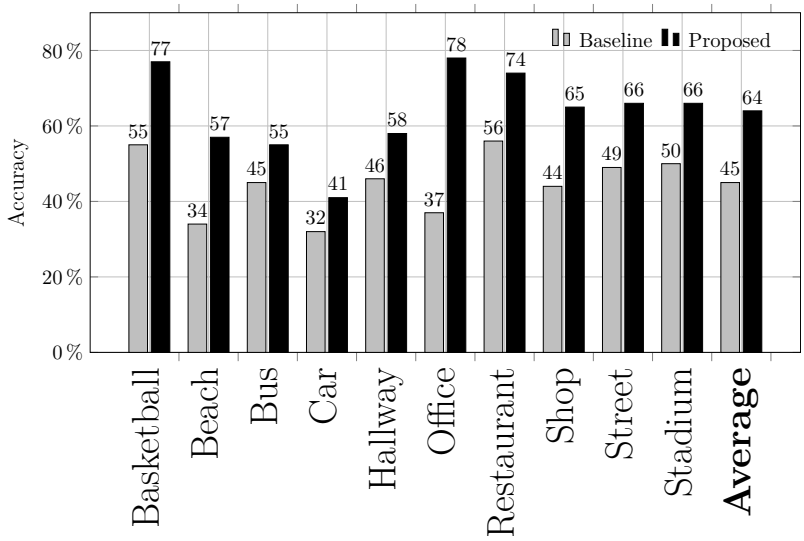


Figure 3.3: Context-wise detection accuracies for proposed system with a comparison to the baseline system. ©2015 IEEE.

first decomposed into four different streams by non-negative matrix factorization (NMF). For each audio stream, SED is performed with an HMM classifier trained using MFCC features.

The F1 score values over ten different contexts are presented in Figure 3.3. The proposed DNN method provides 64% F1 score, which is 19% points higher than the baseline GMM-HMM classifier. This large improvement over the state-of-the-art method at the time can be attributed to the modeling capabilities of DNNs on highly complex relationships. In addition, it is shown that there is a clear positive correlation between the amount of audio data available for a class, and

the F1 score for that class. This can be explained with the following. DNNs have a large number of parameters, so they can model a wide range of functions due to the high degree of freedom. The DNN parameters need to be updated using gradient descent over a large number of training instances that provides a diverse representation of the given sound event class, so that the DNN model can generalize over the unseen instances in the test set. This is especially crucial in SED where the sound event classes are often broadly defined, and sound events with very different acoustic characteristics may end up in the same class.

The performance of DNNs with various sound representations is also investigated in [I]. It is shown that log mel band energies (61.7% F1 score) perform superior to mel band energies (60.4%) and MFCCs (56.8%). This confirms the findings of [26], where this experiment was conducted for single label SEC. It is also shown that median filtering based post-processing provides a valuable boost in performance, especially for lower polyphony levels (10% points increase for polyphony level 1 and 5% for level 2).

### 3.1.3 Contributions and Limitations

To the author's knowledge, [I] is the first work that proposes utilizing DNNs for polyphonic SED. The huge performance improvement of DNNs over the conventional HMM classifiers in this work has contributed to a widespread use of deep learning based methods for SED. The effect of deep learning for SED is very evident in the recent SED challenges such as DCASE2016 [72] and DCASE2017 [71], where over half of the submissions have utilized deep learning methods as classifiers. In addition, the median filtering based post-processing provides an efficient solution to smoothing the noisy predictions of the DNNs for SED, where the noise is often the result of the coarse time resolution of the reference annotations.

The superior performance of DNNs over established GMM-HMM methods were already proven on other machine hearing tasks such as ASR [46] before the publication of [I]. On the other hand, this work also addresses the multi-label multi-class classification capabilities of DNNs, which are harder or often ill-defined for *e.g.* ASR or NLP tasks. In [I] it is shown that the performance of DNN does not decrease even though the polyphony level increases upto 5, which is a challenging case even for human listeners. This makes a strong case for the utilization of these models in rea-life environments.

The main limitation of the DNNs for SED is the inability to model the long term temporal context information. The problem can be remedied to some degree by utilizing a context window as done in [I], however a long input vector makes the number of parameters become quickly very large to train a network in a reasonable amount of time. Besides, context windowing ignores the fact that the same features from consecutive frames carry similar information, and therefore should be connected to the same, shared weights. Besides, the fixed connections between the input and the hidden units of DNNs are not robust for

the small variations in the spectral content of the sound events. For instance, it is common that different dogs would bark at a slightly different frequency. While the spectral shapes for these two sound events are similar, a small offset in the magnitude spectrum would result with DNNs extracting a significantly different representation from these events belonging to the same class. These limitations of the DNNs for polyphonic SED are addressed in the more advanced deep learning methods such as convolutional neural networks (CNN) and recurrent neural networks (RNN).

## 3.2 Multi-label vs. Combined Single Label Sound Event Detection with Deep Neural Networks

### 3.2.1 Context and Motivation

The state-of-the-art results obtained with DNNs for polyphonic SED in [I] has shown that it is worth investigating deep learning methods in various ways for polyphonic SED. Following this idea, two different approaches to utilize DNNs, namely multi-label (ML-DNN) and combined single label DNNs (CSL-DNN), have been compared in [II].

In [111, 112], it is claimed that utilizing the correlation information between the labels is crucial for a classifier to cope with the challenge of the exponential-sized output space due to multi-label target output. This supports the idea that using a multi-label classifier is the optimal choice to exploit the correlation information over the sound events for polyphonic SED. The article [II] tests the validity of these claims for polyphonic SED using DNNs, while considering the practical advantages of using a combined single label classifier over a multi-label classifier. These advantages can be listed as the flexibility of gathering relevant group of sound event classes with various combinations depending on the real-world application, and the ability of dynamical inclusion of new sound event classes without re-training the whole classifier framework.

### 3.2.2 Method

In the sound representation stage, following the improved performance of mel band energies compared to conventional MFCCs for polyphonic SED with DNNs [I], mel band energies have been calculated for each short time frame of length 50 ms with 50% overlap (see Section 3.1.1). The number of filters selected in the mel filterbank is 40, resulting with a 40 dimensional feature vector $\mathbf{x}_t$ for each frame $t$. Similar to [I], context windowing is used to utilize the short term temporal information by concatenating the feature vector with two preceding and two succeeding frame features.

The features extracted from the audio recordings and the target outputs based on the reference annotations are utilized in two separate classifier methods. In

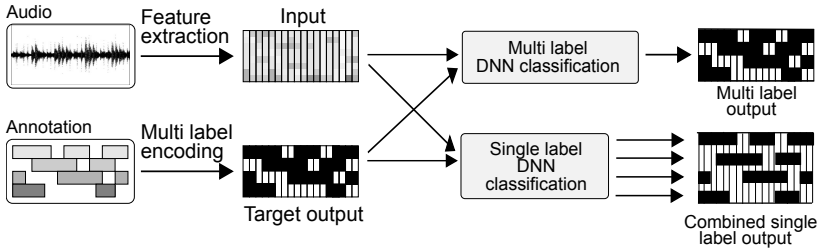


Figure 3.4: Overall framework for multi-label and combined single label classification methods. ©2015 IEEE.

the CSL-DNN method, for each sound event class, a DNN with two hidden layers of 400 units each and an output layer with single unit is trained. The training input is multi-label in content, *i.e.*, it consists of the features extracted from the real-world recordings which may include overlapping sound events. However, each trained network is given the target output of only a single event class. Thus, the network should not only learn the mapping between the features and the given sound event class, but it should also be able to avoid mispredictions when only other sound events are present. The ML-DNN method uses the same classifier as in [I] (see Section 3.1.1), where a DNN is trained in the multi-label setting to predict one or more sound event classes at a given time. The overall framework for both methods is illustrated in Figure 3.4.

The network architecture and the training procedure for both methods is the same, except the number of units in the hidden layers (400 for CSL-DNN vs. 800 ML-DNN) and the output layer (1 vs. $c$, where $c$ is the number of sound event classes). Both networks have two hidden layers with maxout activation functions, and they are optimized using mini-batch SGD to minimize the cross-entropy loss. In the usage stage, the network outputs for both methods are binarized with thresholds 0.2 to 0.9 with 0.1 steps to obtain the presence predictions, which are then post-processed using a median filter based sliding window.

### 3.2.3 Evaluation

The same database of real-life audio recordings with 61 sound event classes used in [I] have also been used for the evaluation of [II] (more details in Section 3.1.2). Average F1 score over one second blocks is used as the evaluation metric.

The test set F1 scores for CSL-DNN and ML-DNN for various binarizing threshold values are presented in Figure 3.5. For the unbiased threshold of 0.5, ML-DNN and CSL-DNN reach to very similar performance (64% vs. 63%), and for other threshold values, the performance difference between the two methods is in the range of $\pm 2\%$ points. This result shows that contrary to the claim made in [111], DNNs can reach to similar performances for polyphonic SED with or without the correlation information between the sound event classes. In addition, the

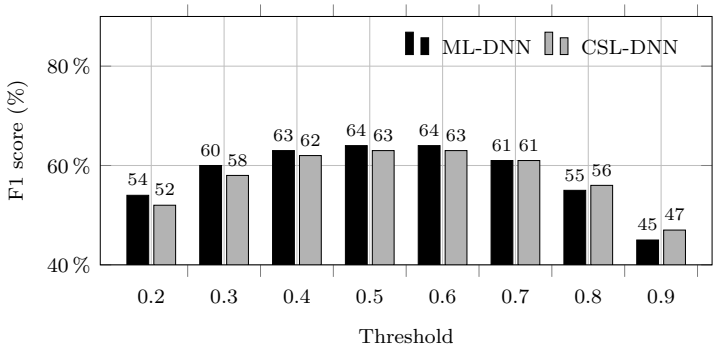


Figure 3.5: Detection performance vs. binarizing threshold for ML-DNN and CSL-DNN. ©2015 IEEE.

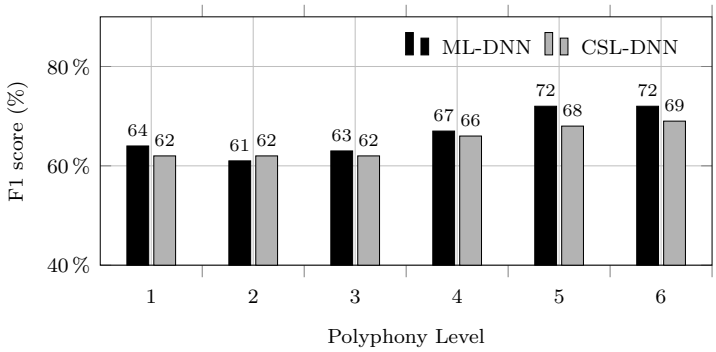


Figure 3.6: Detection performance vs. polyphony level for ML-DNN and CSL-DNN. ©2015 IEEE.

highest F1 score for both methods are obtained around the threshold 0.5, which indicates that the class presence probability distribution over a large number of classes is balanced between 0 and 1 for DNNs. Finally, it can be observed from the Figure 3.5 that ML-DNN performs better than CSL-DNN for lower threshold values, while the opposite happens for higher threshold values.

The effect of the polyphony levels on the performance for CSL-DNN and ML-DNN is presented in Figure 3.6. Unbiased binarizing threshold of 0.5 has been used for these experiments. Considering that the average polyphony level is 2.55 for this dataset, the performance for CSL-DNN and ML-DNN is quite similar over the large portion of the dataset, while ML-DNN gains an edge of $3-4\%$ points for very high polyphony levels mainly due to the class correlation information available. The performance for both methods consistently increase from polyphony levels 2 to 6, which indicates that both methods are robust for polyphonic SED.

### 3.2.4 Contributions and Limitations

In [II], we investigate whether the claim that class correlation information is essential for multi-label classification is also valid for polyphonic SED using DNNs. We observe that the performance drop is very limited in the absence of class correlation information provided to the DNN. We provide a different option in CSL-DNN to utilize DNNs with the benefit of usage case flexibility in exchange to minimal to no decrease in detection performance. The CSL-DNN is most suitable when the sound event classes of interest change frequently for the application.

Similar to [I], the main limitation of CSL-DNN proposed in [II] is the inability to model the long term temporal context information using DNNs. The other limitation is the necessity of training (and storing) a separate model for each of the sound event classes, which can be computationally inefficient when the number of classes is large. Finally, it may have been interesting to experiment the case where the total number of parameters for CSL-DNNs for each class are close or equal to ML-DNN. At the time of writing, this was omitted because there are a total of 61 classes and the aforementioned approach would result with a substantially small CSL-DNN architecture, however the author believes this is worth investigating on a dataset with smaller number of classes.

# 4 Convolutional Recurrent Neural Networks for Sound Event Detection

The methods that are proposed in [III, IV, V] are convolutional recurrent neural networks (CRNN) for various SED tasks such as real-life SED, rare SED, and bird audio detection. While the framework for the methods in these works is the same, there are several variations between the methods mainly due to the task at hand. In this chapter, the motivation for utilizing CRNNs for the given tasks is presented, the main CRNN framework and the differences between the methods are explained, and the findings based on the experiments for each task has been listed.

## 4.1 Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection

Despite the significant performance increase that they offered for SED over the conventional methods, DNNs have been relatively swiftly replaced with more advanced deep learning methods. At the time of writing [III], CRNNs have been starting to replace DNNs as the state-of-the-art methods in a variety of audio classification tasks such as automatic speech recognition [3, 88] and music genre classification [9].

Convolutional recurrent neural networks (CRNN) are one of the more advanced deep learning methods that address the two major shortcomings of DNNs for SED. As also briefly mentioned in Section 3.1.3, these shortcomings are the fixed connections between the input and the hidden units (which prevent modeling small variations among the examples of a sound event class), and the limited ability to model the long term temporal context, which is especially problematic for the typically longer sound events (*e.g.* rain, baby crying, crowd cheering etc.).

Convolutional layers can be used to address the fixed connection limitation by learning filters (*i.e.* weight kernels) that are shared among the input and shifted

in both time and frequency. On the other hand, the temporal context that can be modeled using a convolutional layer is limited. Recurrent layers, especially the ones with a gated structure such as LSTMs and GRUs, can be used to extract long term temporal information among the consecutive time frames by utilizing information from the earlier time frames as a feedback for the calculation of the higher level representation for the current frame. However, RNNs suffer from the same problem with DNNs in terms of fixed connections between the input and the hidden units. Combining convolutional and recurrent layers in a single deep learning architecture helps to combine and emphasize the benefits of both in a single SED method with high performance.

### 4.1.1 Method

The sound representation used as the input for the CRNN is the log mel spectrogram, computed with 40 mel filterbanks in 40 ms time frames with 50% overlap. Therefore, unlike the DNN methods proposed in [I, II], the input of the network is a 2-D representation. Once the log mel spectrogram is calculated, the log energies for each mel band are normalized to zero mean and unit variance. The log mel spectrogram extracted for each recording is then split into sequences of 1024 frames (20.48 seconds), and each sequence represents a single input for the network. The CRNN for SED proposed in [III] consists of four stages: (1) The first stage is composed of stacked convolutional layers with non-overlapping max pooling over the frequency axis; (2) the feature maps for each frame are stacked over the frequency axis and fed to the GRU layers; (3) the output layer is a feed-forward layer with sigmoid activation, and it applies the same set of weights over the higher level representations for each frame; (4) in the usage case, the class presence probabilities obtained from the network output are binarized with a constant threshold 0.5. The overall framework for the proposed CRNN method has been illustrated in Figure 4.1.

Besides the overall framework, there are several regularization techniques employed in order to reduce the overfitting during the training. Dropout [96], a technique that approximates model averaging (see Section 2.7.3 for more details), has been utilized in the convolutional and the recurrent layers. The dropped neurons for the recurrent layers are kept fixed for the entire sequence. In addition to dropout, batch normalization [52], a technique to normalize the layer outputs to remedy the magnitude difference of the gradient updates for different layers (see Section 2.7.3), has been utilized in [III].

### 4.1.2 Evaluation

The CRNN for SED has been evaluated using four different polyphonic SED datasets: TUT-SED synthetic 2016, TUT-SED 2009, TUT-SED 2016 and CHiME-Home (see Section 2.8.2). While the first three mentioned datasets include multiple minute long recordings of polyphonic sound events, CHiME-Home is composed of

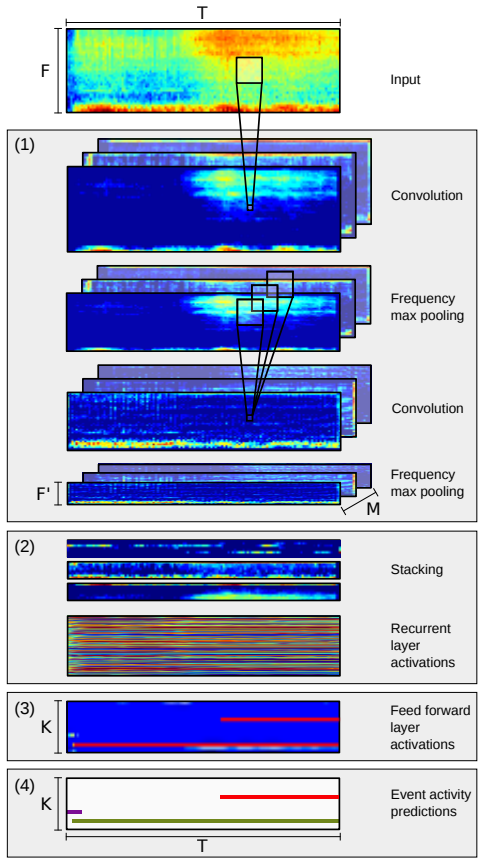


Figure 4.1: Overall framework of CRNN for SED. ©2017 IEEE.

four-second chunks of recordings, and the annotations are available for every four seconds instead of every time frame. Therefore, in [III], a temporal max-pooling layer is added before the output layer of CRNN to get a single presence probability vector for each chunk.

The baseline methods for the evaluation of the proposed CRNN method are GMM, DNN with temporal context input [I], CNN and RNN. The evaluation metrics used in [III] are the F1 score and the segment-based error rate (see Section 2.8.1). Both metrics are calculated both as the average over the time frames ($F1_{frm}$ and $ER_{frm}$) and over the one-second blocks ($F1_{1sec}$ and $ER_{1sec}$). In accordance with DCASE2016 challenge, equal error rate (EER) was used as the evaluation metric for CHiME-Home dataset.

The performances of the methods for the datasets except CHiME-Home is given in Table 4.1. For all the metrics in all the datasets, CRNN perform either the best or within the variance of the best method (variance is based on the average of the experiment results with different random weight initialization schemes, explained in more detail in [III]). Besides, supporting the findings of [I], all deep learning

Table 4.1: Frame-wise and one-second segment-wise F1 score and error rate performance for three datasets. he best performing method for each metric is marked with bold face. ©2017 IEEE.

| | TUT-SED Synthetic 2016 | | | | TUT-SED 2009 | | | | TUT-SED 2016 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $F1_{frm}$ | $ER_{frm}$ | $F1_{1sec}$ | $ER_{1sec}$ | $F1_{frm}$ | $ER_{frm}$ | $F1_{1sec}$ | $ER_{1sec}$ | $F1_{frm}$ | $ER_{frm}$ | $F1_{1sec}$ | $ER_{1sec}$ |
| GMM | 40.5 | 0.78 | 45.3 | 0.72 | 33.0 | 1.34 | 34.1 | 1.60 | 14.1 | 1.12 | 17.9 | 1.13 |
| FNN | 49.2 | 0.68 | 50.2 | 1.1 | 60.9 | 0.56 | 57.1 | 1.1 | 26.7 | **0.99** | **32.5** | 1.32 |
| CNN | 59.8 | 0.56 | 59.9 | 0.78 | 64.8 | 0.50 | 63.2 | 0.75 | 23.0 | 1.02 | 26.4 | 1.09 |
| RNN | 52.8 | 0.6 | 57.1 | 0.64 | 62.4 | 0.52 | 61.8 | 0.55 | **27.6** | 1.04 | 29.7 | 1.10 |
| **CRNN** | **66.4** | **0.48** | **68.7** | **0.47** | **69.7** | **0.45** | **69.3** | **0.48** | **27.5** | **0.98** | 30.3 | **0.95** |

methods outperform the conventional GMM approach by a significant margin.

In order to find the optimal hyper-parameters for the deep learning methods, a grid search is performed including the combinations of some of the hyper-parameters such as the number of layers, number of units in each layer etc. Hence the networks that are used for the results in Table 4.1 do not have the same number of parameters, since each network has different optimal hyper-parameters. The effect of the number of network parameters on the accuracy is visualized in Figure 4.2. The figure is based on the grid search experiment results on TUT-SED synthetic 2016. For the same number of parameters, CRNN outperforms CNN and RNN methods in most cases when the number of parameters is the same between the methods. This supports the idea that combining the CNNs and RNNs into a CRNN classifier is a more efficient and powerful way of utilizing the network parameters compared to CNN and RNN. In addition, the variance of the accuracy for each method indicates that larger and deeper networks do not necessarily perform better, and a grid search should be conducted to find the optimal network architecture. Further experiments on TUT-SED synthetic 2016 on the class-wise performance, the effect of convolutional filter shape and frequency shift invariance can be found in [III].

The effect of sequence length, *i.e.*, the number of frames per input example, is investigated for RNN and CRNN methods on TUT-SED Synthetic 2016 dataset. For these experiments, the number of frames per mini-batch in training is kept fixed, so that the network is trained over the same number of frames in a single update of the parameters. The results are presented in Figure 4.3. Both RNN and CRNN benefit from sequence lengths up to 2.5 seconds. While CRNN mostly benefit from the longer sequence length and gives the best performance at around 20 seconds, the RNN's performance start to decrease for lengths longer than 2.5 seconds.

For TUT-SED Synthetic 2016 dataset, the average duration for the vast majority of the classes are above 5 seconds. This means that the longest temporal context that RNN can model is not sufficient to model the events as a whole in a single sequence. CRNNs can model the whole event in a single sequence, which results in improved performance. This can be explained with the convolutional layer's

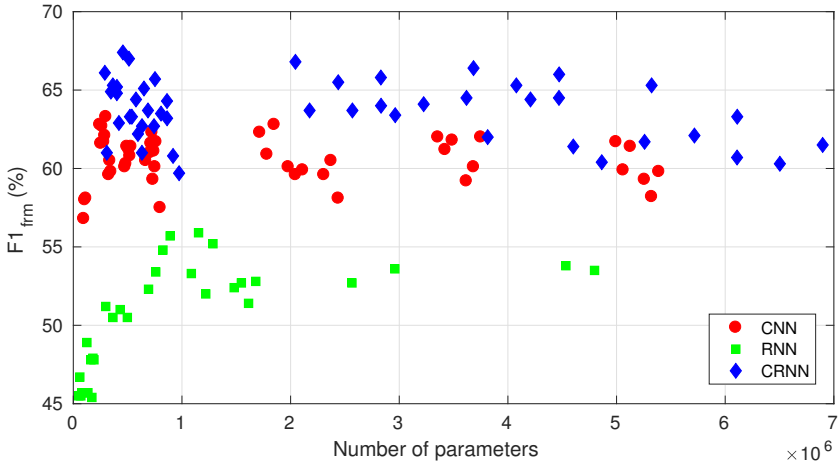


Figure 4.2: Number of parameters vs. F1 score for CNN, RNN and CRNN on TUT-SED synthetic 2016 dataset. ©2017 IEEE.

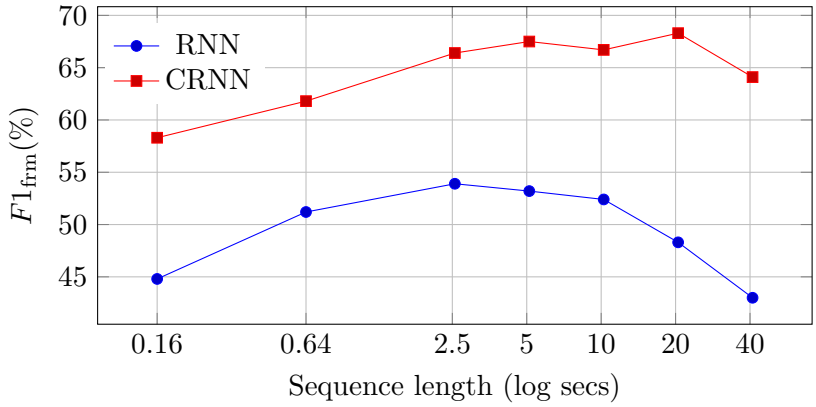


Figure 4.3: Accuracy vs. sequence length (in seconds with logarithmic scale) for RNN and CRNN. ©2017 IEEE.

role in a CRNN architecture. Convolutional layers learn filters that are invariant to short term temporal variations and they effectively pre-process the features to be used in a longer temporal context in the following recurrent layers.

The equal error rate (EER) results for CNN, RNN and CRNN on CHiME-Home dataset are reported in Table 4.2. This dataset has been used for one of the DCASE 2016 challenge subtasks, and therefore we also present the results for the winner [64] of the challenge, which represented the state-of-the-art at the time of writing [III]. For this task, CNN slightly outperforms CRNN while CRNN results are within the variance of CNN results. The difference between the CRNN and CNN architectures is that the GRU layer in the CRNN is replaced by a feed-forward layer followed by batch normalization. To investigate whether CNN

Table 4.2: EER performance for CHiME-Home development and evaluation sets. The best performing method for each metric is marked with bold face. ©2017 IEEE.

| Method | Development EER | Evaluation EER |
|---|---|---|
| Lidy *et al.* [64] | 17.8 | 16.6 |
| CNN | **12.6** | **10.7** |
| RNN | 16.0 | 13.8 |
| CRNN | **13.0** | **11.3** |
| CNN (no batch norm) | 15.1 | 11.9 |

results being better than CRNN was due to the absence of the recurrent layer or to the presence of batch normalization, the experiments for the CNN baseline are repeated after removing the batch normalization after the feed-forward layer. As presented in the last row of Table 4.2, CRNN performs better than CNN when the batch normalization after the feed-forward layer is not involved.

### 4.1.3 Contributions and Limitations

The CRNN method proposed in [III] provided state-of-the-art results for various SED datasets, beating the previous state-of-the-art by a considerable margin. In CRNN, the capability of CNNs to learn spectral and temporal invariant filters and the capability of RNN's to model long term temporal dependencies are combined, resulting in a more powerful classifier than both CNN and RNN for SED. The article [III] conducts a wide-scale investigation on the performance difference between the methods over various angles, while providing a closer look on the network outputs and the learned convolutional filters. After the publication of [III], variants of CRNNs have been widely used for SED tasks, including the winner methods for two of the three SED tasks in the latest SED challenge, DCASE 2017 [71]. Moreover, this article includes experiments on the effects of number of parameters and the sequence length, which are insightful and often a rare sight in the field of SED.

The main limitation of the CRNN is a shared limitation among the deep learning techniques, which is the requirement of large amount of labeled data for training. This becomes the most evident for TUT-SED 2016 results on Table 4.1. The total amount of data for this dataset is only 78 minutes, and there are 18 total sound events, so the total amount of data per class is very limited. This results with low F1 score for all of the deep learning methods. In fact, a similar challenge on this task, *i.e.* real-life multilabel SED, has been introduced in DCASE 2017 [71], where the total amount of data is around two hours while the number of classes is reduced to six. For this task, a CRNN method with log mel-band energy input [1] produces the best results of the challenge, outperforming several CNN and RNN

based methods without utilizing any of the data augmentation and post-processing methods utilized by the latter. The performance improvement in terms of F1 score compared to DCASE 2016 task for the proposed deep learning methods is around 20% points, which highlights the importance of data amount (especially per class) for a real-life multilabel SED task.

## 4.2 Convolutional Recurrent Neural Networks for Bird Audio Detection

### 4.2.1 Context and Motivation

The main motivation for [IV] is to apply the successful CRNN method proposed in [III] on the bird audio detection, as a part of the Bird Audio Detection (BAD) challenge 2017 [101]. The proposed CRNN method is suitable for bird audio detection for the following reasons. Bird audio detection is conducted based on the vocal bird sound such as bird calls and bird songs. Bird calls are often short and serve a particular function such as alarming (high pitch, rapid modulations) or keeping the flock in contact. Bird songs, *e.g.* mating calls, are typically longer and more complex than bird calls, and they often sound melodious to human ears [20]. Vocal bird sounds include distinctive spectral content often including harmonics. Both bird calls and bird songs have distinctive spectral and temporal characteristics, which may be subject to certain shift variations depending on both the bird species and the environmental conditions [17]. Therefore, a bird audio detection method should both be able to capture melodic cues in time domain, and also should be robust to small shifts in spectral content [IV]. In addition, since the BAD challenge is a tagging task, one can argue that there is no benefit on using CRNNs over CNNs because the onset and offset times of the events are not required. On the other hand, bird sounds possess long term temporal characteristics that can be beneficial to model to correctly discriminate them, while the CNN's temporal modeling capabilities are limited in a shorter time context. Finally, the improved performance of CRNNs over CNNs on CHiME-Home dataset (also a tagging problem) in [III] also motivated the authors to choose CRNN as their proposed method for this task.

### 4.2.2 Method

The audio recordings for the challenge [101] are given in 10-second chunks, and the available annotation is file-wise, labeling whether any bird sound is present or not in any portion of a given audio chunk. For sound representation, each 10-second chunk is split into 500 overlapping frames, and log mel spectrogram is obtained with 40 mel filterbanks. The CRNN method utilized in [IV] is composed of three convolutional layers with 96 filters each (followed by max pooling in frequency axis), two GRU layers with 96 hidden units each, a temporal max pooling layer that pools the high level representations over time, and a feed-forward output layer

with a single neuron and logistic sigmoid activation. For network regularization, each convolutional layer is followed by a batch normalization layer and dropout is applied after each convolutional and recurrent layer. The optimal values for the hyper-parameters such as the number of convolutional and recurrent layers, the number of units in each layer and the max pool size after each convolutional layer are obtained based on the grid search over the validation data.

### 4.2.3 Evaluation

The Bird Audio Detection challenge [101] is composed of 68 hours of crowd-sourced field recordings, presented in 10-second chunks (see Section 2.8.2 for more details). Five-fold cross-validation is performed on the development subset, and the optimal neural network configuration is selected based on the average performance over the folds. The proposed method is evaluated with the optimal network configuration on the evaluation subset. The evaluation metric used in [IV] is receiver operating characteristics (ROC) using the area under curve (AUC) measurements (see Section 2.8.1).

As the baseline, a CNN method is utilized where the only difference between the proposed method is that the recurrent layers of CRNN are replaced by the feed-forward layers. The proposed method is also compared against the submissions from two of the top three submitted methods for the challenge (labeled *CNN2* [37] and *CNN3* [78]), which also utilizes CNNs as classifiers. The difference between the baseline CNN method and the other CNN methods are that they use several data augmentation techniques such as frequency and time shifting and pseudo-labeling, and they further apply model ensemble over the networks. These techniques undoubtedly boost the performance of CNNs for the challenge, at the expense of more computational complexity and time consumption for training.

The AUC scores on development and evaluation datasets for the Bird Audio Detection challenge is presented in Table 4.3. The proposed CRNN method ranks second among the submissions for the evaluation dataset of the challenge, behind CNN2 by only 0.2% points. A noteworthy point is that the baseline CNN and the proposed CRNN methods perform similar for the development dataset, while the performance decreases for CNN. However the performance drop for the CRNN is more limited, which indicates CRNN is better at generalizing over the unseen samples in the evaluation dataset, which includes recordings from different environmental and recording conditions than the development dataset.

### 4.2.4 Contributions and Limitations

In the article [IV], we utilize CRNN for bird audio detection. The CRNN classifier is especially suitable for this task due to the acoustic characteristics of the bird sounds, and this is also evident in the performance of the method for the Bird Audio Detection challenge 2017, where the proposed method came second.

Table 4.3: AUC scores on development and evaluation sets.

| Dataset | Method | | | |
|---|---|---|---|---|
| | CNN | CRNN | CNN2 [37] | CNN3 [78] |
| Development | 95.3 | 95.7 | - | - |
| Evaluation | 85.5 | 88.5 | 88.7 | 88.2 |

While the performance of the proposed CRNN method is quite good on the evaluation dataset (88.5% AUC), it is still considerably lower than the development dataset performance (95.7% AUC). This means that CRNN can still be improved for better generalization using *e.g.* regularization and data augmentation techniques. In addition, the performance gap between CNNs and CRNNs in this task is smaller compared to the SED tasks in [III]. This can be attributed to the fact that onset and offset estimation of the sound events, for which the CRNNs excel over CNNs due to temporal modeling capabilities, is not required in this tagging task.

## 4.3 Convolutional Recurrent Neural Networks for Rare Sound Event Detection

### 4.3.1 Context and Motivation

In the article [V], the CRNN method proposed in [III] is applied with minor modifications on a rare sound event detection task, as part of the DCASE 2017 challenge task 2 [71]. The aim of a rare SED system is to detect the presence of a certain sound event that appears only for a small portion of an audio recording, and also to estimate the start and end times for this sound event. Therefore, the proposed method should be able to cope with the data imbalance, and learn the mapping between the acoustic features and the target events from the small amount of data where the sound event is present. Consequently, it is crucial to have robust temporal modeling to detect the sudden changes of the features from consecutive frames, as it may indicate the presence of the rare sound event. The convolutional layers of the CRNN can be used to learn high level, local shift invariant representations from the acoustic features. The gated recurrent layers of the CRNN are useful in long term temporal modeling for detecting rare sound events, as they can reset and update their cell state to reflect the sudden change of features among the consecutive frames due to the presence of a rare sound event.

### 4.3.2 Method

The proposed CRNN method in [III] is utilized in a combined single label setting in [V], where separate CRNNs are trained for each target sound event class. This

combined single label classification approach is similar to [II], where a combined single label DNN that is trained using the polyphonic audio recordings has provided comparable results with a multi-label DNN. Other than the utilized classifier (DNN vs. CRNN), the difference between the experiments in [II] and [V] lies in the polyphony of the audio recordings. For the DCASE 2017 challenge dataset, there can be at most one target sound event present in a recording, and the information of which sound event is possibly present in the recording is provided to the challenge participants. Therefore, it is more logical to utilize this information by training separate CRNN classifiers for each class, rather than training one multi-label classifier.

The acoustic features used in [V] are the log mel spectrogram calculated with 40 mel filters in 40 ms frames and 50% overlap. Each acoustic feature is normalized to zero mean and unit variance based on the mean and the variance calculated for the training set features. The log mel spectrogram is fed as input to convolutional layer block which consists of convolutional layers with linear activations, followed by batch normalization, ReLU activation, and max-pooling in frequency axis. The convolutional block outputs are stacked along the frequency axis and fed to GRU layers, which learn high level representations for each frame. The output of the GRU layer block is fed to a feed-forward layer with a single unit and logistic sigmoid activation. This feed-forward layer applies the same set of weights over the GRU block outputs at each frame, and the outputs of the feed-forward layer is treated as the frame-level class presence probabilities.

### 4.3.3 Evaluation

DCASE 2017 challenge rare SED dataset is used for the evaluation in [V] (see Section 2.8.2 for more details). For the DCASE 2017 challenge, the participants were allowed to generate as many mixtures as they wish for the development dataset, and the challenge results were then computed from the evaluation dataset with fixed size, whose annotations are only available for the challenge organizers. For [V], the same procedure is followed and 4500 development set recordings are generated.

The evaluation metric for rare SED differs from other SED tasks where the frame- or segment-level performance metrics such as F1 score is commonly used. The reason is that the amount of true negatives in the frame labels would be dominant over the true positives, so the classifier would be biased to predict true negatives for all the frames to boost the performance. Therefore, the evaluation metric for the challenge is selected as the event based error rate with onset tolerance of 500 ms [70]. The evaluation is based on the 1500 separate recordings given as a part of the challenge.

CNN is used as the baseline method in [V]. In order to find the optimal hyper-parameters for the CNN and CRNN methods, grid search is conducted. Since a separate classifier is trained for each target class, the optimal network hyper-

Table 4.4: Event-based error rate for the proposed CRNN and the baseline CRNN on the development test set and the evaluation dataset.

| | Development | | | | Evaluation | | | |
|---|---|---|---|---|---|---|---|---|
| Method | Baby cry | Glass break | Gun shot | Average | Baby cry | Glass break | Gun shot | Average |
| CNN-1 | 0.42 | 0.14 | 0.39 | 0.32 | 0.46 | 0.13 | 0.58 | 0.39 |
| CNN-2 | 0.27 | 0.11 | 0.42 | 0.27 | 0.38 | 0.15 | 0.53 | 0.35 |
| CNN-3 | 0.35 | 0.10 | 0.39 | 0.28 | 0.46 | 0.14 | 0.55 | 0.39 |
| CNN-4 | 0.30 | 0.08 | 0.38 | 0.25 | 0.42 | 0.14 | 0.53 | 0.36 |
| CRNN-1 | 0.22 | 0.03 | 0.23 | 0.16 | 0.27 | **0.07** | **0.20** | 0.18 |
| CRNN-2 | 0.19 | 0.03 | 0.19 | 0.14 | **0.18** | 0.10 | 0.23 | **0.17** |
| CRNN-3 | 0.22 | **0.01** | 0.19 | 0.14 | 0.27 | 0.14 | 0.47 | 0.29 |
| CRNN-4 | **0.17** | **0.01** | **0.18** | **0.12** | 0.21 | 0.11 | 0.24 | 0.19 |

parameters may differ between the sound event classes. In [5], both using optimal hyper-parameters for each class and using the hyper-parameters that produce the lowest average error rate among the three classes have been experimented. In addition, model ensemble has also been used in [V] in order to filter the outlier predictions among the top seven networks which produce the lowest error rate results. Therefore, four different CRNN methods have been evaluated in [V] and they are labeled as the following:

- CRNN-1: the architecture with the lowest error rate on average over three classes,
- CRNN-2: the ensemble of the seven best architectures with the lowest error rate on average over three classes,
- CRNN-3: the architecture with the lowest error rate for each of the three classes,
- CRNN-4: the ensemble of seven best architectures with the lowest error rate for each class.
- CNN methods have been obtained in the same fashion to CRNN methods as explained above.

The event based error rate results on the evaluation dataset are given in Table 4.4. The proposed CRNN methods outperform CNNs for rare SED. Considering the only difference between the CNN and the CRNN utilized in [V] is that the recurrent layers of CRNN are replaced by the feed-forward layers in CNN, the performance difference can be attributed to the recurrent layers. In addition, the model ensemble techniques provide a modest boost in the performance (CRNN-1 vs. CRNN-2 and CRNN-3 vs. CRNN-4). Finally, while CRNN-3 and CRNN-4 perform the best on the development test set, CRNN-1 and CRNN-2 give better results on the evaluation set, and the performance difference for some of the

classes between the development set and the evaluation set is noteworthy for CRNN-3 and CRNN-4. This may indicate that the network architectures whose hyper-parameters are selected based on the individual performance for each class overfit to the specific target sound event samples in the development set and struggle to generalize over the unseen samples in the evaluation set.

An insight on how the GRU layer activations contribute to the final network output is provided in Figure 4.4. In panel (a), it can be seen that multiple GRU units trigger the contribution of the candidate output (see Eq. 2.17) at the time when the sound event starts. For the rest of the time when the sound event is present, the network output is controlled by the previous frame outputs of the GRU layer, which is given in panel (b). As a result, the onset and offset times for this event is predicted almost perfectly, as visualized in panel (d).

### 4.3.4 Contributions and Limitations

In [V], CRNN is proposed for rare SED as a part of DCASE 2017 challenge. The proposed method suits the rare SED task due to its capability of learning high level features for the rare sound events and reflecting the sudden changes between the consecutive frame features. This theoretical advantage is also supported by the evaluation results, where the proposed CRNN method comes second in the challenge (the winner also uses a modified version of CRNN [65]). The article [V] also provides a rare, visualized case-study on how the GRU's previous and candidate outputs contribute to the network output.

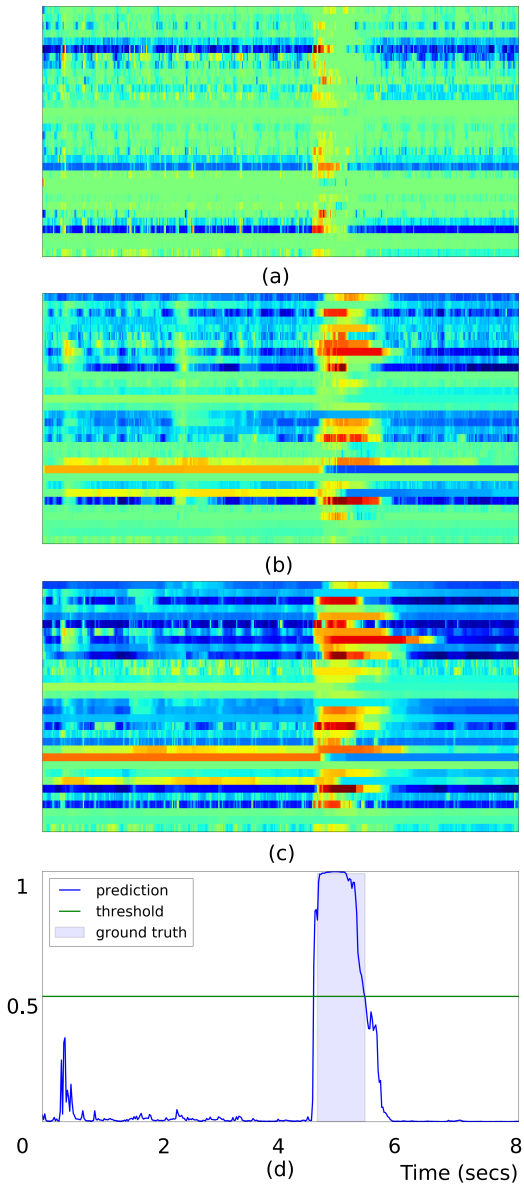


Figure 4.4: (a): contribution of candidate output $\hat{c}_t$, (b): previous output $c_{t-1}$, (c): total contribution of GRU layer to the model output (sum of the candidate and previous outputs); (d): class presence probability vs. time for a portion of a sample including baby cry in the development test set.

# 5 Feature Learning for Polyphonic Sound Event Detection

Machine learning for SED is often performed using perception based sound representations as the input to a machine learning classifier. Advanced machine learning methods such as deep learning classifiers have reduced the need to convert the raw data into a higher level representation before feeding it into the classifier, due to their ability to learn the complex input-target output relationships directly from raw data. In [VI] and [VII], the hand-crafted high level sound representations are replaced with lower level representations such as magnitude spectrograms and even raw audio signals, while integrating the human auditory perception knowledge to the classifier through the network weight initialization.

## 5.1 Filterbank Learning for Deep Neural Network Based Polyphonic Sound Event Detection

Human perception based sound representations utilize empirically obtained knowledge about how humans hear differently the lower and higher frequency components of an audio signal. Most machine learning systems for SED first convert the audio signal into a perception based sound representation such as mel spectrogram, and then train a classifier to find a mapping between this representation and the target outputs. However, it is *not* a certainty that a sound representation that reflects the human sound perception is the optimal representation for a machine learning classifier for SED. Besides, it is claimed that the experiments leading to most commonly used perception scales today such as mel scale are subject to errors. For instance, hysteresis effect (*i.e.* the phenomenon that human auditory perception for sounds with incremental pitch difference is different vs. decremental) is ignored which causes bias in the experiments [35]. Besides, the claim that equal number of mels agree with equal cochlear distances [98] has been long disproven by Greenwod et al. [36], which implies the relationship between human auditory perception and mel scale may be rather weak. In addition, the

empirical, listener-specific nature of the experiments has led to several different hand-crafted formulas for the scales [66, 76, 97, 98]. Nevertheless, perception based representations can be beneficial for an SED system as a starting point for the acoustic features, as they often provide a compact representation of the spectral content of the audio signals. Finally, using modern machine learning algorithms such as deep learning to express the highly nonlinear relationship between the raw data and the target output has been an ongoing area of research for SED. The capability of deep learning classifiers such as DNNs and CNNs to learn high level representations directly from raw data has eliminated the need for extracting hand-crafted features *e.g.* for image recognition [5] and automatic speech recognition [46].

Motivated by these factors, the feature learning capabilities of CNNs and the empirical knowledge of human perception is proposed to be combined in [VI]. This combination is conducted by initializing the first convolutional layer filter weights of a CNN with *e.g.* mel filterbank magnitude response. During the network training, mel filterbank coefficients will be updated to minimize a certain loss function for the SED, thus resulting with an ad-hoc filterbank more optimal for the given SED task than the original mel filterbank.

### 5.1.1 Method

For the sound representation in [VI], the magnitude spectrogram $\mathbf{X} \in \mathbb{R}^{M \times T}$ is used by calculating STFT over the raw audio signals, where $M$ is the number of STFT bins and $T$ is the number of time frames. CNN is used as the machine learning classifier. In order to incorporate temporal information at the input, the input to the CNN is the context window $\hat{\mathbf{X}} \in \mathbb{R}^{M \times C}$ for the magnitude spectrum vector $\mathbf{x}$ at each frame, where $C$ is the number of frames in the context window. Based on the magnitude spectrum $\mathbf{x}_t$ at frame $t$, the context window matrix $\hat{\mathbf{X}}_t$ is obtained by concatenating the magnitude spectrum vectors from $\frac{C-1}{2}$ preceding and succeeding consecutive frames as

$$\hat{\mathbf{X}}_t = \begin{bmatrix} X_{1,t-\frac{C-1}{2}} & X_{1,t-\frac{C+1}{2}} & \dots & X_{1,t} & \dots & X_{1,t+\frac{C-1}{2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{M,t-\frac{C-1}{2}} & X_{M,t-\frac{C+1}{2}} & \dots & X_{M,t} & \dots & X_{M,t+\frac{C-1}{2}} \end{bmatrix} \tag{5.1}$$

The input - target output pair of $(\hat{\mathbf{X}}_\mathbf{t}, \mathbf{y}_t)$ represents a single training example for the network, where $\mathbf{y}_t$ is the binary target output vector for the frame $t$ obtained from the reference annotations.

For SED, the input to the machine learning classifier is often a perception based sound representation, such as mel band energies. Mel band energies are calculated by applying a mel filterbank over the magnitude spectrum. In [VI], the filter weights for the first layer of the CNN are initialized with the mel filterbank

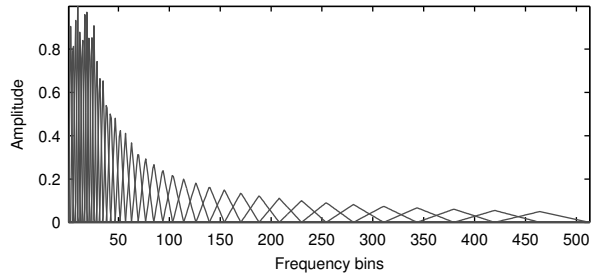


Figure 5.1: Mel filterbank magnitude response.

magnitude response, which effectively results with the mel band energies at the output of the first layer before the training.

### Mel filterbank magnitude response

Mel scale is a perceptually motivated frequency scale whose intervals correspond to equal perceptual pitch increments. The reference point is often chosen as setting 1000 mels to 1 kHz. Although there are several formulas on how to convert $f$ hertz to $l$ mels, in [VI], the commonly used formula in [76] is utilized as:

$$l = 2595 \log_{10}(1 + \frac{f}{700}) \tag{5.2}$$

Mel filterbank magnitude response $\mathbf{F} \in \mathbb{R}^{B \times M}$ is composed of the coefficients for $B$ triangular band-pass filters whose central frequencies are equally spaced in mel scale. This leads to more narrowly spaced filters in the lower frequency levels, and widely spaced filters in the higher frequency levels. The coefficients of the triangular filters are selected in the range $[0, 1]$ and they are scaled so that the area under each triangle for magnitude response is approximately the same over the mel bands. Mel filterbank magnitude response is visualized in Figure 5.1.

### CNN with mel filterbank weight initialization

The classifier used for SED in [VI] is the CNN whose first layer filter weights are initialized with mel filterbank magnitude response. The weight sharing property of CNNs is utilized to apply the same filterbank over the magnitude spectrum for each time frame.

The CNN used in [VI] consists of a single convolutional layer followed by two feed-forward hidden layers and a feed-forward output layer. The convolutional layer consists of $B$ filters with shape $[M, 1]$. Given the context window input matrix $\hat{\mathbf{X}}$, the output $\mathbf{Z}$ of the convolutional layer is calculated as

$$Z_{b,c} = \sigma(\sum_{m=1}^{M} W_{b,m} \hat{X}_{m,c}) \qquad (b = 1, 2, ..., B,\ c = 1, 2, ..., C) \tag{5.3}$$

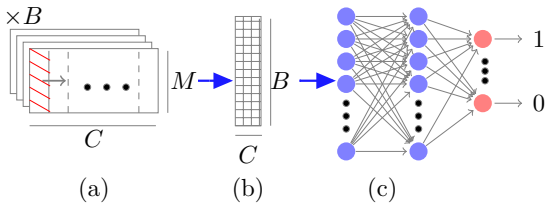


Figure 5.2: CNN topology with convolutional layer weights initialized using mel filterbank coefficients. The feature map outputs for $B$ band are shown in separate channels. ©2016 IEEE.

where $\sigma$ is the activation function and $W_{b,m}$ is the $m^{th}$ filter weight for $b^{th}$ filter.

The convolutional layer filter weights $\mathbf{W}$ are initially set as the mel filterbank magnitude response $\mathbf{F}$. In addition, the activation function $\theta$ is selected as ReLU, thus before training the network, the convolutional layer output $\mathbf{Z}$ is equal to the mel band energies for the given context window. In order to initially set $\mathbf{W}$ as $\mathbf{F}$ and get the mel band energies as the layer output, the filter length must be equal to the number of STFT bins, and the time dimension of the filter shape must be set to 1, so that the same filterbank is utilized at each frame. This differs from the usual way of utilizing the convolutional layers, where the filter shape is often smaller than the input shape and the filter is two dimensional, so that the convolution operation can be performed in both dimensions. For this reason, this network is called appropriately constrained CNN in [VI]. The network architecture is visualized in Figure 5.2.

The initial weights for feed-forward layers are small values that are randomly sampled from a uniform distribution $U(-\Delta, \Delta)$, where $\Delta$ is calculated from the randomized initialization method proposed in [29]. Thus the random weights for each layer have zero mean and their variance is calculated as $\sigma^2 = \frac{1}{12}(-\Delta - \Delta)^2$. This creates a mismatch between the mean and the variance of the initial convolutional layer weights $\mathbf{W}$, which can be detrimental to the network learning. Therefore, the mean and the variance of the convolutional layer weights are matched with the rest of the network layers before the network training. Finally, the initial zero weights of $\mathbf{W}$ are replaced with small random values drawn from a uniform distribution with zero mean and unit variance.

### 5.1.2 Evaluation

The dataset used in [VI] can be deemed as an early version of TUT-SED synthetic 2016 (explained in Section 2.8.2). There are nine sound event classes: alarm clock, baby crying, cat meowing, dog barking, door bell, footsteps, glass break, music and speech. From the isolated audio recordings for these classes, polyphonic mixtures are created by mixing the recordings from different classes with random onset. The mixing procedure is explained in more detail in [VI]. Around 9% of

Table 5.1: F1 score as a function of the acoustic feature, classifier method and context window length $C$.

| Feature | Architecture | C=1 | C=3 | C=5 | C=7 | C=9 | C=11 | C=13 |
|---|---|---|---|---|---|---|---|---|
| MEL | DNN random init | 69.3 | 75.3 | 77.1 | 77.9 | 77.9 | 78.9 | 78.9 |
| FFT | CNN random init | 71.4 | 76.1 | 78.2 | 79.4 | 80.6 | 81.2 | 80.7 |
| FFT | CNN mel init | 71.5 | 75.8 | 78.5 | 80.2 | 81.2 | 80.8 | 80.8 |
| FFT | CNN mel init ($\mu$ and $\sigma$ match) | 71.3 | 76.2 | 78.4 | 79.7 | 80.9 | **81.8** | 81.3 |

the dataset is polyphonic, *i.e.* two or more sound events are present at the same time.

The evaluation metric is frame-wise F1 score. For the baseline, a DNN with two hidden layers and mel band energies with context window input is implemented. This makes a direct comparison possible between the author's earlier work with DNNs [I, II] and the proposed method. The second baseline is a CNN with the exact same architecture with the proposed CNN, but the convolutional layer filter weights are initialized randomly. For both baseline and the proposed methods, each experiment has been repeated ten times with different random seed (to compensate the effect of random weight initialization on the performance) and the average F1 score has been presented.

The frame-wise F1 score results for the baseline and the proposed methods are given in Table 5.1. Among the whole experiment, the best performance is obtained by CNN with mel weight initialization and magnitude spectrum input with a context window of 11 frames. For almost all cases, the performance for the baseline and the proposed methods increase with the increased context window length $C$ up to $C = 11$, which reflects the importance of the temporal context for the learning.

### 5.1.3 Contributions and Limitations

The proposed method in [VI] is a first attempt to learn an ad-hoc filterbank for SED using the feature learning capabilities of deep learning methods and the empirical knowledge about the human auditory perception. Initializing the first convolutional layer filter weights of a CNN with the mel filterbank magnitude response provides a modest increase in performance over a CNN whose weights are initialized randomly.

The main limitation is the restriction on the convolutional layer filter shape with the mel weight initialization. The method makes the filter length tied to the number of STFT bins, although the filter length is usually selected as smaller than the input for convolutional layers. Besides, the method does not allow convolution in time domain, because the filterbank output obtained through the convolutional layer must be limited to a single frame so that the same filterbank is applied over each frame. This certainly reduces the modeling capabilities of the CNN. Another limitation is the limited temporal context modeling capabilities of the

CNNs. The best performance obtained by the CNN is with the context window input of $C = 11$ that is approximately 240 ms, which is quite short considering the average duration of the events. The performance of the CNN decreases with longer context window. An overall limitation of this work is the presentation of the results. Since the corresponding F1 scores for proposed and baseline methods are quite close, confidence intervals should have been calculated, which deemed to be rather challenging retroactively during the publication of this thesis due to missing experimental documentation. This weakens the paper's credibility on the recommendation of utilizing mel weight initialized networks. In hindsight, if the performance difference between the proposed and baseline methods are this close, then statistical significance tests have to be conducted. On the other hand, confidence interval experiments were in fact done for the evaluation of [VII], which investigates a similar approach (feature learning through mel weight initialization), and these experiments also show with more certainty that the performance gain for this approach is rather limited, if not none.

## 5.2 End-to-End Polyphonic Sound Event Detection Using Convolutional Recurrent Neural Networks with Learned Time-Frequency Representation Input

SED is often conducted in two stages: sound representation and classification. In the sound representation stage, hand-crafted acoustic features are extracted from the raw audio signal, and in the classification stage, a machine learning method is used to learn a mapping between the extracted features and the target sound event classes. These two stages are often implemented exclusively, *i.e.* they do not utilize any information from each other. Besides, while the sound representation methods offer a compact representation of the time-frequency domain content of the audio signal, they also discard some information that could have been useful for the machine learning classifier.

Recently, there were a few attempts that proposed using directly raw audio data as input in several sound classification tasks [18, 89], but the obtained performance was either below or on-par with the methods using hand-crafted acoustic features. For the case of SED, the research on using lower level representations as input is even more limited, while the main research direction is to use an established human perception based sound representation such as mel spectrogram as the input and focus on developing novel and effective machine learning methods.

### 5.2.1 Method

In [VII], the sound representation and classification stages of a typical SED system are proposed to be combined in a single deep learning classifier. The input to the

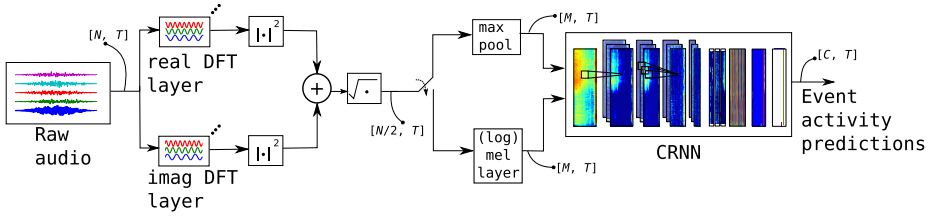


Figure 5.3: Method framework. The method output shape in various stages of the framework is given in brackets. ©2018 IEEE.

classifier is the raw audio signal in short time frames, and the sound representation stage is implemented as a feature extraction block of neural network layers, which has trainable parameters. This block is appended with a CRNN classifier which completes the network. Since the mapping between the raw audio signal and the target sound event is learned through a single classifier, the proposed method is an example of end-to-end SED system.

The method proposed in [VII] consists of a feature extraction block of feed-forward layers, and a convolutional recurrent layer block. The output of the feature extraction block is initially (*i.e.* before the network training starts) either the mel spectrogram, log mel spectrogram or the max pooled magnitude spectrogram. The extracted features from this block are fed to the convolutional recurrent block, which then maps these features onto target sound events. The two layer blocks are trained jointly, *i.e.* they update their weights iteratively to minimize the same SED loss function.

### Feature Extraction Block

The input $\mathbf{X} \in \mathbb{R}^{N \times T}$ to the feature extraction block is the raw audio signals in $T$ short time frames with $N$ samples, and Hamming window is applied to each frame. In order to initially obtain the magnitude spectrogram from the raw audio signals, $\mathbf{X}$ is fed into two feed-forward layers with weights $\mathbf{W}^{\text{re}}$ and $\mathbf{W}^{\text{im}}$ and no bias, which are initialized with the sine and cosine filter weights used in the Discrete Fourier Transform (DFT):

$$\begin{aligned} \mathbf{F}_{k,t} &= \sum_{n=0}^{N-1} \mathbf{X}_{n,t}[\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)] \\ \mathbf{W}^{\text{re}}_{k,n} \leftarrow \cos(2\pi kn/N), &\quad \mathbf{W}^{\text{im}}_{k,n} \leftarrow \sin(2\pi kn/N) \\ \mathbf{Z}^{\text{re}}_{k,t} = \sum_{n=0}^{N-1} \mathbf{W}^{\text{re}}_{k,n}\mathbf{X}_{n,t}, &\quad \mathbf{Z}^{\text{im}}_{k,t} = \sum_{n=0}^{N-1} \mathbf{W}^{\text{im}}_{k,n}\mathbf{X}_{n,t} \end{aligned} \tag{5.4}$$

for $k = 0, 1, ..., \frac{N}{2} - 1$ and $n = 0, ..., N - 1$, where $\mathbf{Z}$ is the weighted output for each feed-forward layer. The magnitude spectrogram $\mathbf{S}$ is obtained through the

root of the squared sum of the layer outputs:

$$\mathbf{S}_{k,t} = |\mathbf{F}_{k,t}| = \sqrt{(\mathbf{Z}^{\text{re}}_{k,t})^2 + (\mathbf{Z}^{\text{im}}_{k,t})^2} \tag{5.5}$$

where the square and root operations are also implemented as neural network layers. The obtained magnitude spectrogram is then either max pooled into $M$ features per frame, or it is fed to a feed-forward layer with weights $\mathbf{W}^{\text{mel}}$ initialized to get the mel spectrogram:

$$\mathbf{Z}^{\text{mel}}_{m,t} = \max(0, \sum_{k=0}^{N/2-1} \mathbf{W}^{\text{mel}}_{m,k} \mathbf{S}_{k,t}) \tag{5.6}$$

for $m = 0, 1, ...M-1$. In the former case, the purpose of max-pooling is to obtain features of same length for magnitude and mel spectrogram. The weights $\mathbf{W}^{\text{mel}}$ are initialized with the mel filterbank magnitude response in a similar manner as [VI]. ReLU activation function is used to avoid possible negative values in the mel spectrogram once the network starts training. For the experiments where this mel spectrogram layer is utilized, the parameters $\mathbf{W}^{\text{re}}$ and $\mathbf{W}^{\text{im}}$ are kept fixed so that the effect of making the mel filterbank coefficients learnable can be observed.

#### Convolutional Recurrent Block

The convolutional recurrent block follows the same method proposed in [III]. It consists of convolutional layers with ReLU activations and max-pooling only in frequency axis, GRU layers, and a feed-forward output layer with logistic sigmoid activation function. The same set of feed-forward layer weights is applied to GRU outputs at each frame. In the usage case, the frame-level class presence probabilities are converted into binary presence predictions by using a constant threshold of 0.5.

### 5.2.2 Evaluation

The dataset used for evaluation in [VII] is TUT-SED synthetic 2016 (see Section 2.8.2). While the original sampling rate is 44.1 kHZ for this dataset, the recordings are resampled to 8, 16, and 24 kHz in [VII] to reduce the computational complexity of the end-to-end training. The evaluation metrics are selected as frame-wise F1 score and error rate (see Section 2.8.1). Each experiment in run ten times with different random seeds and the mean and the standard deviation (given as $\pm$) are presented.

The F1 score and error rate results are given in Table 5.2. The experiments with fixed feature extraction parameters often outperform the ones with learned parameters with the same sampling rate. On the other hand, the performance at 8 kHz is the same with 60.8% F1 score 0.55 error rate for both fixed and learned feature extraction block. Considering that the original sampling rate is 44.1 kHz,

Table 5.2: Frame-level F1 score $F1_{\text{frm}}$ and error rate $ER_{\text{frm}}$ results for different time-frequency representation methods and sampling rates. "DFT" stands for magnitude spectrogram using linear frequency scale, "mel" stands for mel spectrogram, "fixed" and "learned" stands for whether the weights of the feature extraction block are kept fixed or updated during training.

| Method | $F1_{\text{frm}}$ | $ER_{\text{frm}}$ |
|---|---|---|
| DFT 8 kHz fixed | 60.8±0.8 | 0.55±0.01 |
| DFT 8 kHz learned | 60.8±0.8 | 0.55±0.01 |
| Mel 8 kHz fixed | 60.8±0.9 | 0.55±0.01 |
| Mel 8 kHz learned | 61.0±0.8 | 0.56±0.01 |
| Log mel 8 kHz fixed | 63.1±0.6 | 0.52±0.01 |
| Log mel 8 kHz learned | 58.6±1.6 | 0.56±0.01 |
| DFT 16 kHz fixed | 61.9±0.9 | 0.54±0.01 |
| DFT 16 kHz learned | 60.1±1.7 | 0.58±0.03 |
| Mel 16 kHz fixed | 62.3±0.7 | 0.54±0.01 |
| Mel 16 kHz learned | 60.6±0.9 | 0.57±0.02 |
| Log mel 16 kHz fixed | 65.8±1.4 | 0.50±0.01 |
| Log mel 16 kHz learned | 59.9±1.3 | 0.56±0.01 |
| DFT 24 kHz learned | 58.1±1.6 | 0.59±0.03 |
| Log mel 44.1 kHz fixed [III] | 66.4±0.6 | 0.48±0.01 |

downsampling to 8 kHz effectively means not utilizing any knowledge about the signal energy between 8-44.1 kHz, however the performance drop of 5.6% points in F1 score is quite limited. It can be considered to resample the audio recordings with lower sampling rates to lower the computational load without losing much performance for end-to-end SED systems.

While the feature extraction block with learned parameters could not outperform the fixed one, the advantage of using such a method is the insight that it provides based on the modifications of the parameters during training. The magnitude response peaks for the parameters $\mathbf{W}^{\text{re}}$ and $\mathbf{W}^{\text{im}}$ after training are visualized in Figure 5.4. For some context, since $\mathbf{W}^{\text{re}}$ and $\mathbf{W}^{\text{im}}$ are initialized with sine and cosine filter weights, the magnitude response before training equals to a single impulse at the center frequency of the filter. During training, the weights are updated and the sinusoidal property of the weights is distorted. Therefore the magnitude response of each filter is no longer only an impulse, but a nonlinear curve with positive values at various frequency bins and a peak value still at the center frequency of the filter. In Figure 5.2, these peaks are visualized for each filter, and the experiment is repeated for several sampling rates. It can be observed that although the systems using different sampling rates have different CRNN hyper-parameters (based on the grid search), there is a clear pattern for

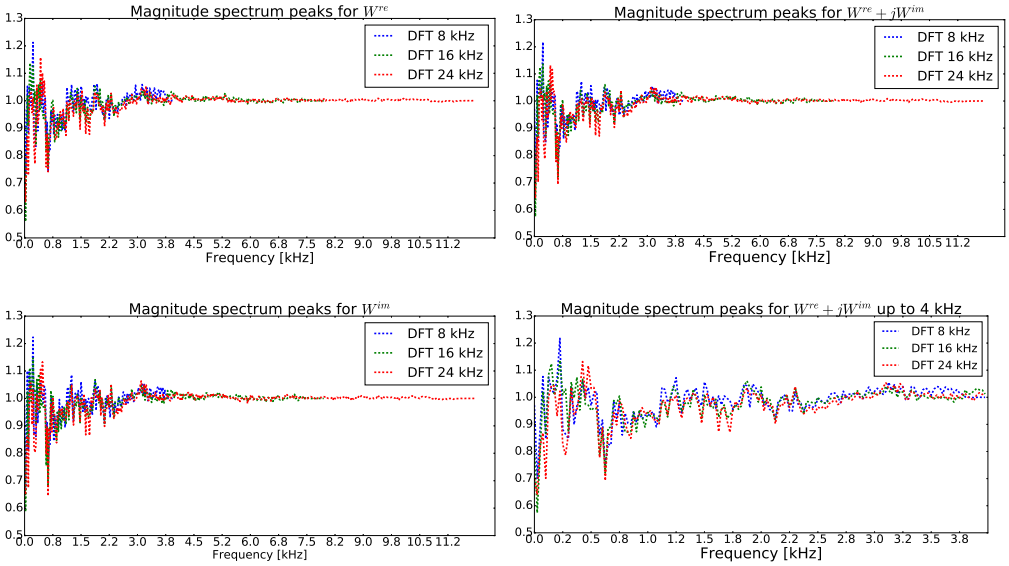


Figure 5.4: Magnitude spectrum peaks for the real ($\mathbf{W}^{\text{re}}$) and imaginary ($\mathbf{W}^{\text{im}}$) DFT layer filters after the training. The amplitude of the peak for each filter is positioned at the center frequency of the corresponding filter, resulting in a line plot covering the whole frequency range for the experiment with given sampling rate. ©2018 IEEE.

how the magnitude spectrum peak changes for each filter after the training. In addition, the peak changes are not monotonic in the frequency axis and the peaks fluctuate considerable in certain frequency regions. This implies that, based on the network training to minimize the SED loss, the sound representation obtained through the network is very different than magnitude and mel spectrogram.

### 5.2.3 Contributions and Limitations

The proposed method in [VII] aims to conduct SED using directly raw audio data as input. To the author's knowledge, the proposed method is the first to integrate the domain knowledge of the perception based sound representations into the parameters of a deep learning classifier to conduct end-to-end SED. Unlike the established way of initializing the network weights with small random values, a block of neural network layer parameters are initialized to extract the hand-crafted features at the beginning of the training. By observing how these parameters are updated during training, an insight can be gained on the optimal sound representation for the classifier to conduct SED. The proposed method also offers a data-driven approach on designing audio filterbanks for SED by optimizing the filterbank parameters with the gradient descent algorithm to minimize the SED loss.

The main limitation of the proposed method is that it does not offer a performance increase over using established acoustic features as input. In addition, the computational cost of using raw audio as input is higher than using a more compact representation such as mel spectrogram, due to the significant dimensionality reduction while obtaining mel spectrogram from the raw audio signal (representing one second of 16 kHz audio as 40 mel band energies extracted from 40 ms windows and 50% overlap corresponds to around 87% dimensionality reduction). Finally, even though the proposed method has the rare benefit of not simply using mel spectrogram as the acoustic feature and discarding the phase information completely, however the analysis of how the learned filters change the estimated phase of the raw audio signals is missing from the analysis.

# 6 Conclusions and Discussions

## 6.1 Conclusions

In this thesis, we investigate the effectiveness of several deep learning methods for various SED tasks. These tasks can be listed as polyphonic SED in real-life indoor and outdoor environments, polyphonic SED on the synthetic mixtures of individual sound events, rare SED, and bird audio detection.

In [I], we show that a DNN using mel band energy input with a context window can significantly outperform a traditional GMM-HMM classifier based method by 42% relative performance increase in F1 score. The performance can be further increased with the proposed median filtering based post-processing approach, which involves smoothing the frame-wise DNN outputs that are treated as class presence probabilities. While the context windowing is used to somehow introduce the temporal modeling for the DNN, we show that this temporal post-processing approach can further increase the performance up to 10% at various polyphony levels. Finally, we observe that the system performance is robust to increased polyphony levels, as the frame-level F1 score increases from 62.5% at polyphony level 1 to 64.5% at level 2.

In [II], we compare two different approaches to polyphonic SED: a multi-label DNN and a combined single label DNN. While the multi-label is trained to model multiple sound events simultaneously, the combined single label DNN approach is based on training a separate network for each sound event. We show that the performance for these two methods is similar, which refutes the earlier claims on the necessity of class correlation information for multi-label classification.

In [III], [IV] and [V], we show that using a CRNN with mel spectrogram as input provides superior performance to both conventional classifier methods such as GMM-HMM and NMF-HMM, and also other deep learning methods such as DNN, CNN, and RNN. CRNN combines the shift-invariant filters of convolutional layers and the long term temporal modeling capabilities of GRUs, resulting with a classifier that suits various SED tasks. We note that for bird audio detection [IV] and rare SED [V], it is beneficial to adjust the originally proposed CRNN architecture in [III] to include a temporal max pooling layer before the output layer. Lastly, using model ensemble boosts the system performance by

filtering the outliers in the outputs of the networks with different initial random weights.

In [VI] and [VII], we investigate the feature learning capabilities of deep learning methods. We show that it is possible to obtain comparable SED performance with deep learning methods using lower level sound representation inputs such as magnitude spectrogram or even raw audio signals, when the weights of the first layer are initialized with the filterbank coefficients of the commonly used sound representation techniques. Moreover, by comparing the initial weights and the final weights after the training, we are able to observe how the filterbank coefficients for calculating *e.g.* magnitude or mel spectrogram are updated using gradient descent optimization for the given SED task.

## 6.2 Discussions

Research on SED has been gaining a lot of interest in the recent years. There are certain factors that facilitate this interest. Some of them can be listed as the introduction of large-scale benchmark datasets, more standardized performance metrics and evaluation protocols. DCASE challenges, organized in 2013 [99] and annually since 2016 [71, 72], have been a major contributor for all these three factors.

In terms of machine learning algorithms, what increased the interest on SED research is the introduction and remarkable success of deep learning methods for SED. Similar to many other machine learning tasks such as image recognition, ASR, natural language processing (NLP) etc., deep learning methods such as DNNs, CNNs and CRNNs significantly outperformed the conventional classifier methods such as GMM-HMM for the task of SED.

This thesis covers some of the very early examples of deep learning methods applied to SED, such as DNNs [I] and CRNNs [III,IV,V]. It should be noted that the research on deep learning is currently very active, and novel, more powerful algorithms are proposed in a rapid fashion. While the DNN algorithm for SED in [I] was proposed in 2015 and it significantly outperformed the non-deep learning state-of-the-art method for SED at the time (42% relative performance increase on F1 score), it quickly became obsolete against the more efficient deep learning techniques such as bidirectional RNNs [77], CNNs [91] and CRNNs [III]. To our knowledge, the current state-of-the-art on different SED benchmark datasets are certain variants of CRNNs [1, 65],[III]. This is also supported by the fact that the winners of four out of seven total sound event detection tasks For DCASE 2017 and 2018 challenges use CRNNs as their machine learning classifier [1, 53, 65, 109]. On the other hand, considering the speed of progress in the deep learning methods, the readers should be aware that these methods may also become obsolete in due time.

It can be claimed that the recent progress on SED research, including the findings in this thesis, follow suit with other machine hearing tasks, and also contribute on specific cases which have not been widely experimented for other tasks. The early applications of deep learning methods (such as DNNs with unsupervised training [47]) on ASR [46] has shown substantial improvements over established GMM-HMM method. Consequently, this has been shown to be applicable for SED [26]. In [I], DNN's performance benefits over GMM-HMM were once again proven for SED, and in addition, the multi-label classification capabilities of DNNs were emphasized. While it was shown in [II] that the class correlation information is not essential for DNNs for SED, to the author's knowledge, similar experiments for other machine learning tasks are currently missing from the literature. Besides, this is worth re-visiting with the recent, more advanced deep learning methods before drawing more conclusions on the importance of class correlations for SED, and other machine hearing tasks. Variants of the CRNN method, whose successful application on SED is one of the key contributions in this thesis, are still being extensively utilized with high performance in many other machine hearing tasks such as audio chord estimation [54](MIREX2018 challenge winner) and ASR [38, 108](current top methods on Switchboard dataset [30]). To the author's knowledge, weight initialization based feature learning techniques similar to the ones proposed in [VI] and [VII] are currently only experimented on SED, while it is conceivable that these techniques would perform better and lead to more intuitive learned filterbanks for more structured data such as chord estimation and music transcription.

While the current state-of-the-art techniques have a quite similar framework for SED and other machine hearing tasks, the author sees that SED is currently a few steps behind in terms of productization, especially compared to ASR. Home-assistant devices such as Google Home and Amazon Alexa, and smartphone devices are readily providing robust ASR products. The author believes that SED fits very well in the consumer-product space, especially considering the fact that no user input is required for action (vs. ASR where a user speech input is required). It is always a challenge to know exactly which techniques are utilized in the proprietary products, but recent publications from these companies imply that the methods considered to be used for SED in these devices are in fact based on CNNs [45], RNNs [106] and CRNNs [55], that are very similar to the methods presented in this thesis. For curious readers, I recommend reading S. Krstulović's views on audio event recognition in smart home [58].

Currently, there is a considerable amount of deep learning research focused on the generative modeling algorithms such as Generative adversarial networks (GAN) and variational autoencoders (VAE). These algorithms are often used to model the underlying characteristics of the given dataset, and to produce more examples that are similar to the ones in the given dataset. A recent application of GANs on a machine hearing task was proposed in [74], which utilized GANs as a way of data augmentation (creating additional training examples) and won the

acoustic scene detection subtask for DCASE2017. I believe that in the future we will be very likely to encounter these recent deep learning methods for SED as well, considering the fact that deep learning methods often benefit from large datasets and data augmentation almost always increases the performance of these systems for SED. In addition, the recent introduction of very large-scale SEC datasets such as AudioSet [25] accelerated the research on SED systems that can detect the onset/offset of the sound events when the annotations are available only per recording. Such systems are said to learn *strong* labels from *weak* labels [59, 63, 109] and they can significantly reduce the cost of labor for the annotation of SED datasets in the future.

It is evident from the findings of [III, IV, V] that CRNNs provide a considerable numerical advantage in terms of SED performance over other deep learning based methods such as DNNs, CNNs and RNNs. While the theoretical and empirical benefits of CRNNs are listed in detail in this thesis, in the future, it may be wise to spend more effort for finding answers to questions such as: how does the long-term temporal modeling differ over the sound event classes with CRNNs, what is the effect of sequence length, how does a single CRNN incorporate the temporal modeling for the sound events that are very different in terms of temporal characteristics etc. The analysis of the contributions of candidate and present activations of the GRU to the network's output (please see Section 4.3.3) is a step towards understanding more of the recurrent layer's contribution in the context of SED. The author believes that the possible findings in these directions would be crucial for developing better, more robust SED methods in the future.

# Bibliography


[1] Adavanne, S. and Virtanen, T., "A report on sound event detection with different binaural features," DCASE2017 Challenge, Tech. Rep., September 2017.

[2] Allen, J. B., "Cochlear modeling," *IEEE Acoustics, Speech, and Signal Magazine*, vol. 2, no. 1, pp. 3–29, 1985.

[3] Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G. *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 173–182.

[4] Bello, J. P., Mydlarz, C., and Salamon, J., "Sound analysis in smart cities," in *Computational Analysis of Sound Scenes and Events*, Virtanen, T., Plumbley, M., and Ellis, D. P., Eds. Springer, 2018, pp. 373–397.

[5] Bengio, Y., "Learning deep architectures for AI," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[6] Boser, B. E., Guyon, I. M., and Vapnik, V. N., "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[7] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y., "On the properties of neural machine translation: Encoder-decoder approaches," *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[8] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *Conference on Empirical Methods in Natural Language Processing*, 2014.

[9] Choi, K., Fazekas, G., Sandler, M., and Cho, K., "Convolutional recurrent neural networks for music classification," *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017.

[10] Chu, S., Narayanan, S., Kuo, C., and Mataric, M., "Where am I? scene recognition for mobile robots using audio features," in *IEEE International Conference on Multimedia and Expo*, 2006, pp. 885–888.

[11] Chu, S., Narayanan, S., and Kuo, C.-C. J., "Environmental sound recognition with time–frequency audio features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1142–1158, 2009.

[12] Coates, A. and Ng, A. Y., "Learning feature representations with k-means," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 561–580.

[13] Dalal, N. and Triggs, B., "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886–893.

[14] Davis, S. B. and Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," in *Readings in speech recognition*. Elsevier, 1990, pp. 65–74.

[15] Dennis, J., Tran, H. D., and Li, H., "Spectrogram image feature for sound event classification in mismatched conditions," *IEEE Signal Processing Letters*, vol. 18, no. 2, pp. 130–133, 2011.

[16] Dennis, J. W., "Sound event recognition in unstructured environments using spectrogram image processing," Ph.D. dissertation, Nanyang Technological University, Singapore, 2014.

[17] Derryberry, E. P., "Ecology shapes birdsong evolution: variation in morphology and habitat explains variation in white-crowned sparrow song," *The American Naturalist*, vol. 174, no. 1, pp. 24–33, 2009.

[18] Dieleman, S. and Schrauwen, B., "End-to-end learning for music audio," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 6964–6968.

[19] Dikmen, O. and Mesaros, A., "Sound event detection using non-negative dictionaries learned from annotated overlapping events," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2013, pp. 1–4.

[20] Ehrlich, P., Dobkin, D., and Wheye, D., "Birds of Stanford essays," accessed: 2018-07-20. [Online]. Available: http://web.stanford.edu/group/stanfordbirds/text/uessays/essays.html

[21] Ellis, D. P., "Gammatone-like spectrograms," accessed: 2018-07-20. [Online]. Available: http://www.ee.columbia.edu/dpwe/resources/matlab/gammatonegram

[22] Eyben, F., Böck, S., Schuller, B., and Graves, A., "Universal onset detection with bidirectional long-short term memory neural networks," in *Proc. 11th Intern. Soc. for Music Information Retrieval Conference, ISMIR, Utrecht, The Netherlands*, 2010, pp. 589–594.

[23] Fix, E. and Hodges Jr, J. L., "Discriminatory analysis-nonparametric discrimination: consistency properties," California Univ Berkeley, Tech. Rep., 1951.

[24] Foster, P., Sigtia, S., Krstulovic, S., Barker, J., and Plumbley, M. D., "Chimehome: A dataset for sound source recognition in a domestic environment," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2015, pp. 1–5.

[25] Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M., "Audio set: An ontology and human-labeled dataset for audio events," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 776–780.

[26] Gencoglu, O., Virtanen, T., and Huttunen, H., "Recognition of acoustic events using deep neural networks," in *22nd European Signal Processing Conference*, 2014, pp. 506–510.

[27] Gerhard, D., *Audio signal classification: History and current techniques*. Citeseer, 2003.

[28] Glasberg, B. R. and Moore, B. C., "Derivation of auditory filter shapes from notched-noise data," *Hearing research*, vol. 47, no. 1-2, pp. 103–138, 1990.

[29] Glorot, X. and Bengio, Y., "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[30] Godfrey, J. J., Holliman, E. C., and McDaniel, J., "Switchboard: Telephone speech corpus for research and development," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 1. IEEE, 1992, pp. 517–520.

[31] Goetze, S., Schroder, J., Gerlach, S., Hollosi, D., Appell, J.-E., and Wallhoff, F., "Acoustic monitoring and localization for social care," *Journal of Computing Science and Engineering*, vol. 6, no. 1, pp. 40–50, 2012.

[32] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*. Boston, MA: MIT Press, 2016, http://www.deeplearningbook.org.

[33] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y., "Maxout networks." *ICML (3)*, vol. 28, pp. 1319–1327, 2013.

[34] Graves, A., Mohamed, A.-r., and Hinton, G., "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[35] Greenwood, D., "The mel scale's disqualifying bias and a consistency of pitch-difference equisections in 1956 with equal cochlear distances and equal frequency ratios," *Hearing research*, vol. 103, no. 1-2, pp. 199–224, 1997.

[36] Greenwood, D. D., "Critical bandwidth and the frequency coordinates of the basilar membrane," *The Journal of the Acoustical Society of America*, vol. 33, no. 10, pp. 1344–1356, 1961.

[37] Grill, T., "Source code for the BAD challenge submission, user: bulbul," https://jobim.ofai.at/gitlab/gr/bird_audio_detection_challenge_2017/tree/master, 2017.

[38] Han, K. J., Chandrashekaran, A., Kim, J., and Lane, I., "The capio 2017 conversational speech recognition system," *arXiv preprint arXiv:1801.00059*, 2017.

[39] Hawkins, D. M., "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[40] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[41] Heittola, T., Mesaros, A., Eronen, A., and Virtanen, T., "Audio context recognition using audio event histograms," in *18th European Signal Processing Conference*, 2010, pp. 1272–1276.

[42] ———, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, p. 1, 2013.

[43] Heittola, T., Mesaros, A., Virtanen, T., and Gabbouj, M., "Supervised model training for overlapping sound events based on unsupervised source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8677–8681.

[44] Heittola, T., Çakır, E., and Virtanen, T., "The machine learning approach for analysis of sound scenes and events," in *Computational Analysis of Sound Scenes and Events*, Virtanen, T., Plumbley, M., and Ellis, D. P., Eds. Springer, 2018, pp. 13–40.

[45] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B. *et al.*, "Cnn architectures for large-scale audio classification," in *Acoustics, Speech*

*and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131 – 135.

[46] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N. *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82 – 97, 2012.

[47] Hinton, G. E. and Salakhutdinov, R. R., "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[48] Hinton, G. E., Osindero, S., and Teh, Y.-W., "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527 – 1554, 2006.

[49] Hochreiter, S., "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107 – 116, 1998.

[50] Hochreiter, S. and Schmidhuber, J., "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735 – 1780, 1997.

[51] Huang, P.-S., Kim, M., Hasegawa-Johnson, M., and Smaragdis, P., "Deep learning for monaural speech separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 1562 – 1566.

[52] Ioffe, S. and Szegedy, C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 448 – 456.

[53] JiaKai, L., "Mean teacher convolution system for dcase 2018 task 4," DCASE2018 Challenge, Tech. Rep., September 2018.

[54] Junyan, J., Ke, C., Wei, L., and Guangyu, X., "Mirex 2018 submission: A structural chord representation for automatic large-vocabulary chord transcription," MIREX 2018, Tech. Rep., 2018.

[55] Kao, C.-C., Wang, W., Sun, M., and Wang, C., "R-crnn: Region-based convolutional recurrent neural network for audio event detection," *arXiv preprint arXiv:1808.06627*, 2018.

[56] Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," *International conference on learning representations*, 2014.

[57] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097 – 1105.

[58] Krstulović, S., "Audio event recognition in the smart home," in *Computational Analysis of Sound Scenes and Events*, Virtanen, T., Plumbley, M., and Ellis, D. P., Eds. Springer, 2018, pp. 335–371.

[59] Kumar, A. and Raj, B., "Audio event detection using weakly labeled data," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1038–1047.

[60] LeCun, Y., "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, 1989.

[61] LeCun, Y. and Bengio, Y., "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[62] Lee, D. D. and Seung, H. S., "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[63] Lee, D., Lee, S., Han, Y., and Lee, K., "Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input," DCASE2017 Challenge, Tech. Rep., September 2017.

[64] Lidy, T. and Schindler, A., "CQT-based convolutional neural networks for audio scene classification and domestic audio tagging," DCASE2016 Challenge, Tech. Rep., September 2016.

[65] Lim, H., Park, J., and Han, Y., "Rare sound event detection using 1D convolutional recurrent neural networks," DCASE2017 Challenge, Tech. Rep., September 2017.

[66] Makhoul, J. and Cosell, L., "LPCW: An LPC vocoder with linear predictive spectral warping," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 1976, pp. 466–469.

[67] Mallat, S. and Zhang, Z., "Matching pursuit with time-frequency dictionaries," Courant Institute of Mathematical Sciences New York United States, Tech. Rep., 1993.

[68] Mesaros, A., Heittola, T., Eronen, A., and Virtanen, T., "Acoustic event detection in real life recordings," in *18th European Signal Processing Conference*, 2010, pp. 1267–1271.

[69] Mesaros, A., Heittola, T., and Virtanen, T., "TUT database for acoustic scene classification and sound event detection," in *24th European Signal Processing Conference*, 2016.

[70] ———, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[71] Mesaros, A., Heittola, T., Diment, A., Elizalde, B., Shah, A., Vincent, E., Raj, B., and Virtanen, T., "Dcase 2017 challenge setup: Tasks, datasets and baseline system," in *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[72] Mesaros, A., Heittola, T., Benetos, E., Foster, P., Lagrange, M., Virtanen, T., and Plumbley, M. D., "Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 379–393, 2018.

[73] Mitchell, T. M. *et al.*, *Machine learning*. Boston, MA: McGraw-Hill, 1997.

[74] Mun, S., Park, S., Han, D., and Ko, H., "Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane," DCASE2017 Challenge, Tech. Rep., September 2017.

[75] Mydlarz, C., Salamon, J., and Bello, J. P., "The implementation of low-cost urban acoustic monitoring devices," *Applied Acoustics*, vol. 117, pp. 207–218, 2017.

[76] O'Shaughnessy, D., *Speech communication: human and machine*. Universities press, 1987.

[77] Parascandolo, G., Huttunen, H., and Virtanen, T., "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2016, pp. 6440–6444.

[78] Pellegrini, T., "Source code for the BAD challenge submission, user: topel," github.com/topel/bird_audio_detection_challenge, 2017.

[79] Pelvig, D., Pakkenberg, H., Stark, A., and Pakkenberg, B., "Neocortical glial cell numbers in human brains," *Neurobiology of aging*, vol. 29, no. 11, pp. 1754–1762, 2008.

[80] Peng, Y.-T., Lin, C.-Y., Sun, M.-T., and Tsai, K.-C., "Healthcare audio event classification using hidden markov models and hierarchical hidden markov models," in *IEEE International Conference on Multimedia and Expo*, 2009, pp. 1218–1221.

[81] Phan, H., Krawczyk-Becker, M., Gerkmann, T., and Mertins, A., "DNN and CNN with weighted and multi-task loss functions for audio event detection," DCASE2017 Challenge, Tech. Rep., September 2017.

[82] Piczak, K. J., "Environmental sound classification with convolutional neural networks," in *IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2015, pp. 1–6.

[83] ———, "ESC: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018.

[84] Polyak, B. T., "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[85] Radhakrishnan, R., Divakaran, A., and Smaragdis, A., "Audio analysis for surveillance applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE, 2005, pp. 158–161.

[86] Rakotomamonjy, A. and Gasso, G., "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 23, no. 1, pp. 142–153, 2015.

[87] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[88] Sainath, T. N., Vinyals, O., Senior, A., and Sak, H., "Convolutional, long short-term memory, fully connected deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 4580–4584.

[89] Sainath, T. N., Weiss, R. J., Senior, A., Wilson, K. W., and Vinyals, O., "Learning the speech front-end with raw waveform cldnns," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[90] Salamon, J. and Bello, J. P., "Unsupervised feature learning for urban sound classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 171–175.

[91] ———, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[92] Salamon, J., Jacoby, C., and Bello, J. P., "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM international conference on Multimedia.* ACM, 2014, pp. 1041–1044.

[93] Schilit, B., Adams, N., and Want, R., "Context-aware computing applications," in *First Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85 – 90.

[94] Schuster, M. and Paliwal, K. K., "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673 – 2681, 1997.

[95] Somervuo, P., Harma, A., and Fagerlund, S., "Parametric representations of bird sounds for automatic species recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 6, pp. 2252 – 2263, 2006.

[96] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929 – 1958, 2014.

[97] Stevens, S. S. and Volkmann, J., "The relation of pitch to frequency: A revised scale," *The American Journal of Psychology*, pp. 329 – 353, 1940.

[98] Stevens, S. S., Volkmann, J., and Newman, E. B., "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185 – 190, 1937.

[99] Stowell, D., Giannoulis, D., Benetos, E., Lagrange, M., and Plumbley, M., "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733 – 1746, Oct 2015.

[100] Stowell, D., "Computational bioacoustic scene analysis," in *Computational Analysis of Sound Scenes and Events*, Virtanen, T., Plumbley, M., and Ellis, D. P., Eds. Springer, 2018, pp. 303 – 333.

[101] Stowell, D., Wood, M., Stylianou, Y., and Glotin, H., "Bird detection in audio: a survey and a challenge," in *26th International Workshop on Machine Learning for Signal Processing*, 2016, pp. 1 – 6.

[102] Sutskever, I., Martens, J., Dahl, G., and Hinton, G., "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139 – 1147.

[103] Takahashi, N., Gygli, M., Pfister, B., and Van Gool, L., "Deep convolutional neural networks and data augmentation for acoustic event recognition," in *Interspeech*, 09 2016, pp. 2982 – 2986.

[104] Valero, X. and Alias, F., "Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification," *IEEE Transactions on Multimedia*, vol. 14, no. 6, pp. 1684 – 1689, 2012.

[105] Wang, J.-C., Lee, H.-P., Wang, J.-F., and Lin, C.-B., "Robust environmental sound recognition for home automation," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 1, pp. 25 – 31, 2008.

[106] Wang, W., Kao, C.-c., and Wang, C., "A simple model for detection of rare sound events," *arXiv preprint arXiv:1808.06676*, 2018.

[107] "Deafness and hearing loss, key facts," http://www.who.int/en/news-room/fact-sheets/detail/deafness-and-hearing-loss, WHO, accessed: 2018-04-05.

[108] Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., and Stolcke, A., "The microsoft 2017 conversational speech recognition system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5934 – 5938.

[109] Xu, Y., Kong, Q., Wang, W., and Plumbley, M. D., "Surrey-CVSSP system for DCASE2017 challenge task4," DCASE2017 Challenge, Tech. Rep., September 2017.

[110] Yamakawa, N., Takahashi, T., Kitahara, T., Ogata, T., and Okuno, H. G., "Environmental sound recognition for robot audition using matching-pursuit," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2011, pp. 1 – 10.

[111] Zhang, M.-L. and Zhou, Z.-H., "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338 – 1351, 2006.

[112] – – –, "A review on multi-label learning algorithms," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 8, pp. 1819 – 1837, 2014.

[113] Zhang, T. and Wu, J., "Learning long-term filter banks for audio source separation and audio scene classification," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2018, no. 1, p. 4, 2018.

# Publications

# Publication I

Emre Çakır, Toni Heittola, Heikki Huttunen, Tuomas Virtanen. "Polyphonic Sound Event Detection Using Multi Label Deep Neural Networks", in *Proceedings of the International Joint Conference on Neural Networks (IJCNN).* Killarney, Ireland, July 2015, pp. 1-7.

©2015 IEEE. Reprinted, with permission, from E. Çakır, T. Heittola, H. Huttunen, and T. Virtanen, Polyphonic Sound Event Detection Using Multi Label Deep Neural Networks, Proceedings of the International Joint Conference on Neural Networks (IJCNN), July 2015.

# Polyphonic Sound Event Detection Using Multi Label Deep Neural Networks

Emre Cakir, Toni Heittola, Heikki Huttunen and Tuomas Virtanen

*Abstract*— **In this paper, the use of multi label neural networks are proposed for detection of temporally overlapping sound events in realistic environments. Real-life sound recordings typically have many overlapping sound events, making it hard to recognize each event with the standard sound event detection methods. Frame-wise spectral-domain features are used as inputs to train a deep neural network for multi label classification in this work. The model is evaluated with recordings from realistic everyday environments and the obtained overall accuracy is 63.8%. The method is compared against a state-of-the-art method using non-negative matrix factorization as a pre-processing stage and hidden Markov models as a classifier. The proposed method improves the accuracy by 19% percentage points overall.**

## I. Introduction

Sound event is the audio segment that humans would label as a distinctive concept in an acoustic signal [1]. The aim of automatic sound event detection is to recognize the sound events present in a continuous acoustic signal. Monophonic sound event detection deals with the most prominent event at a time instance and polyphonic detection tackles the situations where multiple sound events happen simultaneously. The applications of sound event detection include multimedia indexing [2], scene recognition for mobile robots [3] and surveillance in living environments [4].

The additive nature of sound sources makes it difficult to find the robust features to detect them in polyphonic audio. Conventional classifiers that have been used in speech recognition and monophonic detection are not as successful in polyphonic detection. Monophonic sound event detection systems handle the polyphonic data by detecting only the prominent event, resulting with a loss of information in realistic environments [5]. Polyphonic detection is essential to get high accuracy in complex auditory scenes. State-of-the-art polyphonic detection systems are using Mel-Frequency Cepstral Coefficients (MFCC) to characterize the audio signals and using Hidden Markov Models (HMMs) as classifiers with consecutive passes of the Viterbi algorithm [6]. Recently, non-negative matrix factorization (NMF) was used as a pre-processing step to decompose the audio into streams and detect the most prominent event in each stream at a time [1]. However, the fixed constraint of the NMF on the number of overlapping events reduces its practicality when this number is not known *a priori*. The estimation of the number of overlapping events can be bypassed when using coupled NMF, as shown in [7]. In [8], local spectrogram features were combined with Generalized Hough Transform (GHT) voting system to detect the overlapping sound events. This offers a different path than traditional frame-based features and achieves high accuracy, being evaluated on five different sound events and their combinations.

Polyphonic detection can be formulated as a multi label classification problem. Multi label problem can be addressed by applying single label classification for each of the classes and combining the results. However, the single label encoding of the problem discards the correlation structure between the classes resulting in a weak expressive power [9]. Therefore, multi label classification is necessary to obtain the most of the available information from the real-world data. Some of the applications of multi label classification are used in overlapping sound event recognition [8], scene classification [10] and text categorization [11].

In this paper, we propose to use multi label feed-forward deep neural networks (DNN) for polyphonic sound event detection. In our earlier paper, we have shown that with sufficient numbers of hidden layers, hidden units and training data, DNNs can outperform HMM methods in sound event classification tasks [12]. However, in this paper we extend the work of [12] by encoding the problem as a multi label learning task with no limitations to the number of simultaneous events. the motivation of our work is that DNN can use different sets of its hidden units to model multiple simultaneous events in a given time instance, benefiting from a different nature of nonlinearity than the conventional mixture models [13]. We use spectral domain features to characterize the audio signals and DNNs to learn a mapping between features and sound events. The contribution of this paper is to extend the use of DNNs to the multi label analysis of realistic recordings from everyday environments and model overlapping sound events in a natural way. We also propose a post-processing method to filter the noise in the DNN outputs. The highly realistic and diverse audio material used in this work offers a firm insight on the usability of the method in real-world applications.

The structure of the paper is as follows: the task of the polyphonic sound event detection and the feature extraction process are explained in Section 2. The input-output structure and the architecture of the DNN is described in Section 3. A post-processing method to smoothen the DNN output is explained in Section 4. Section 5 contains the experimental results on the highly realistic material and comparison with the baseline results. In the end, our conclusions on the topic are given in Section 6.

*This work was done with the support of Audio Research Group in Department of Signal Processing, Tampere University of Technology, Finland.

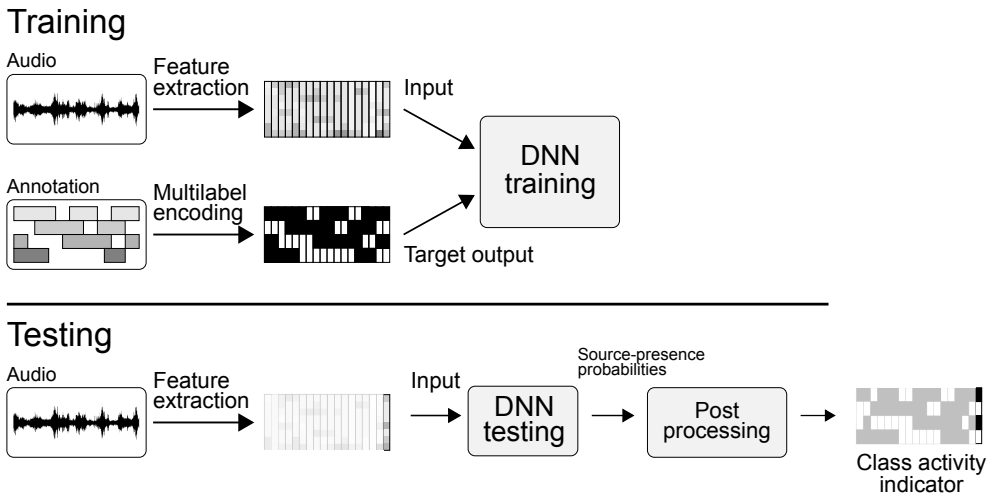


Fig. 1: Framework of the training and testing procedure for the proposed system.

## II. Sound Event Detection

The main objective of the proposed method is to temporally locate the sound events in a recording collected from a realistic auditory scene and give each event a label from a set of possible labels. The framework of the proposed sound event detection method is shown in Figure 1.

Auditory scenes are composed of multiple sound events occurring at the same time. Detecting the events separately from a realistic auditory scene leads to a multi label classification problem. Figure 2 illustrates the polyphonic nature of sound events in realistic environments.

As a pre-processing step for the feature extraction, the recordings are amplitude normalized, divided into frames and Hamming window with 50 ms duration and 50% overlap is applied. The spectral domain features (*e.g.* Mel-band and log Mel-band energies) and cepstral domain features (*e.g.* MFCCs) are extracted from the short time frames of the audio signal. For each time frame, a feature vector $\mathbf{x}_t$ is obtained, where $t$ is the frame index. Each feature vector corresponds to a learning instance for the neural network.

In order to extract the dynamic property information of the signal, a frame concatenation method is used. The feature vectors that are extracted from the adjacent time frames are concatenated together to form a single training instance. The resulting feature vector has a dimension of $|\mathbf{x}| = (2 \times N_{\text{adj}} + 1) \times N_{\text{f}}$ where $N_{\text{adj}}$ is the number of adjacent frames concatenated with the original frame and $N_{\text{f}}$ is the number of features extracted from the short time frame. This method is often called *context windowing* and has been used in many other studies as well [12], [14], [15].

For each frame, target output vector $\mathbf{y}_t$ includes the multi label encoding of the audio events present in the frame. Each sound event is assigned to a class which is encoded as a single binary variable. The events present in a frame are annotated with 1 and the rest is 0. An illustrative example of annotation can be found in Figure 2, where the target output vector $\mathbf{y}$ for frame $t$ is $\mathbf{y}_t = [1\ 0\ 1]$. The number of possible classes is known in advance and therefore the length of the output vector is fixed, but the number of active events in a frame is not known *a priori*.

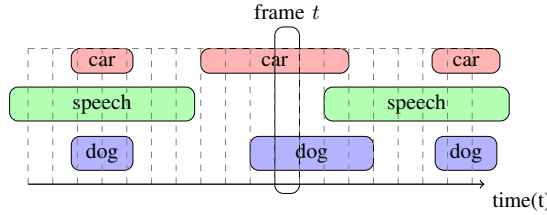


Fig. 2: Overlapping sound events in a recording from a realistic environment. Frame $t$ represents the short time frame from the recording where only car and dog barking events are present.

## III. Multi Label Neural Network

Feed-forward neural networks with multiple hidden layers, *i.e.*, deep neural networks (DNN) are used for multi label classification. Deep architectures build a hierarchy among the features. In each layer, higher level features are extracted implicitly by the composition of lower level features. This automatic structure eases the work of learning highly non-linear functions mapping the input to the output directly from data, therefore reducing the need to find human-crafted intermediate representations [16].

DNNs are composed of an input layer, multiple layers of hidden units with nonlinear activation functions and an output layer. The input vector $\mathbf{x}_t$ consists of the spectral features

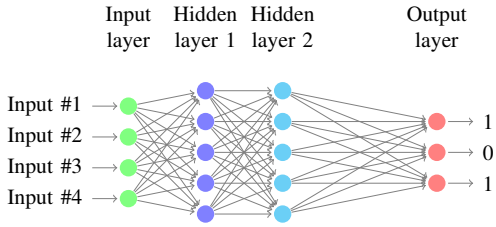


Fig. 3: Symbolic representation of the NN topology with two hidden layers and multi label outputs.

extracted from frame $t$. For simplicity, frame index $t$ will be omitted during the feed-forward algorithm description.

The output vector $\mathbf{h}^k \in \mathbb{R}^M$ for layer $k$ with $M$ units is calculated from the weighted sum of the outputs for the previous layer $\mathbf{h}^{k-1} \in \mathbb{R}^D$. Starting with $\mathbf{h}^0 = \mathbf{x}$ and

$$\mathbf{g}^k = \mathbf{W}^k \mathbf{h}^{k-1} + \mathbf{b}^k, \; 1 \leq k < L \tag{1}$$

$$\mathbf{h}^k = f(\mathbf{g}^k) \tag{2}$$

where $\mathbf{W}^k \in \mathbb{R}^{D \times M}$ is the weight matrix between $(k-1)^{\text{th}}$ layer with $D$ units and $k^{\text{th}}$ layer with $M$ units, $\mathbf{b}^k \in \mathbb{R}^M$ is the bias vector for the $k^{\text{th}}$ layer, $L$ is the number of layers and $f(\cdot)$ is the nonlinear activation function applied element-wise. Output $\mathbf{h}^L \in \mathbb{R}^N$ is used as the source presence prediction vector $\hat{\mathbf{y}} = \mathbf{h}^L$, where $\hat{\mathbf{y}}(i)$ is the source presence prediction for the event $i \in [1, 2, ..N]$ and $N$ is the number of sound events. During the training stage, $\hat{\mathbf{y}}$ is involved in calculating the cost function, explained below in detail. During the testing stage, $\hat{\mathbf{y}}$ is binarized with a threshold to get the binary detection vector $\mathbf{z}_t$. An illustration of a small-scale DNN with binarized output is presented in Figure 3.

Maxout function [17] for hidden layers and logistic sigmoid function (bounded between 0 and 1) for output layer are used as activation functions in DNN structure. Maxout is a piecewise linear activation function which can be seen as a generalization of rectified linear units [18]. Maxout calculates the maximum of a set of $R$ affine projections of the input. In mathematical terms, given $\mathbf{g}^k = [g^k(0), g^k(1), ..., g^k(j), ...g^k(M \times R - 1)]$, for non-overlapping pools of size $R$, maxout function implements

$$\mathbf{h}^k(i) = \max_{r=0}^{R-1} \mathbf{g}^k(j + r) \quad \text{where } j = i \cdot R \tag{3}$$

where $\mathbf{h}^k \in \mathbb{R}^M$, $\mathbf{g}^k \in \mathbb{R}^{M \times R}$ for layer $k$ with $M$ units and $R$ is the number of affine feature mappings. Hidden units with maxout functions at each layer are divided into non-overlapping pieces and each piece generates a single activation via the max pooling operation, as illustrated in Figure 4. Unlike conventional optimization functions, max-out is not bounded, it is easier to optimize and does not suffer from vanishing gradients problem by sparsifying the gradients [14].

Stochastic gradient descent (SGD) algorithm is used as the learning algorithm for the DNN. Training cost function for the neural network is selected as Kullback-Leibler (KL)

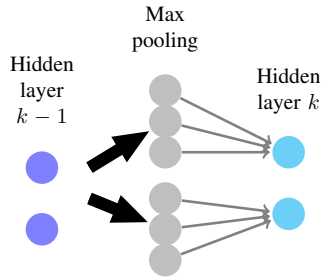


Fig. 4: Maxout activation function with $R = 3$ feature mappings.

divergence, as it is able to characterize the general accuracy of the class membership probabilities [19]. KL divergence is calculated as

$$\begin{aligned} KL(\mathbf{y}_t || \hat{\mathbf{y}}_t) = \sum_{i=1}^{N} & \mathbf{y}_t(i) \log \mathbf{y}_t(i) \\ & - \mathbf{y}_t(i) \log \hat{\mathbf{y}}_t(i) \\ & + (1 - \mathbf{y}_t(i)) \log (1 - \mathbf{y}_t(i)) \\ & - (1 - \mathbf{y}_t(i)) \log (1 - \hat{\mathbf{y}}_t(i)) \end{aligned} \tag{4}$$

where $\mathbf{y}_t(i)$ is the target output for the $i^{th}$ event, $\hat{\mathbf{y}}_t(i)$ is the source presence prediction obtained from the output layer for the $i^{th}$ event and $N$ is the total number of event classes. For binary $\mathbf{y}_t$, as in our case, some terms in (4) drop out and the resulting KL divergence is

$$\begin{aligned} KL(\mathbf{y}_t || \hat{\mathbf{y}}_t) = \sum_{i=1}^{N} & -\mathbf{y}_t(i) \log \hat{\mathbf{y}}_t(i) \\ & -(1 - \mathbf{y}_t(i)) \log (1 - \hat{\mathbf{y}}_t(i)) \end{aligned} \tag{5}$$

The DNN parameters such as number of hidden units, learning rate, initial weight bias etc. are selected by a grid search over the parameter values. The instances are processed over mini batches of size 50. The most successful topology for this task is found to be DNN with 2 hidden layers with 800 units each.

## IV. Post-processing

Environmental sound events naturally take at least a few seconds, once they are initiated. When we experimented with environmental audio, we noticed some abrupt changes between consecutive frames in the detection probabilities for some of the events. Our reasoning to this is as follows. The audio is processed in very short time frames and the events may contain intermittent periods. The annotation of the audio material is done with a rather coarse time resolution, since a human annotator would miss these less (if any) active frames in the events and do the annotation for larger chunks of frames. Although these frames are erroneously annotated with some of the labels, they do not have the spectral characteristics of the labels associated with them. Moreover,

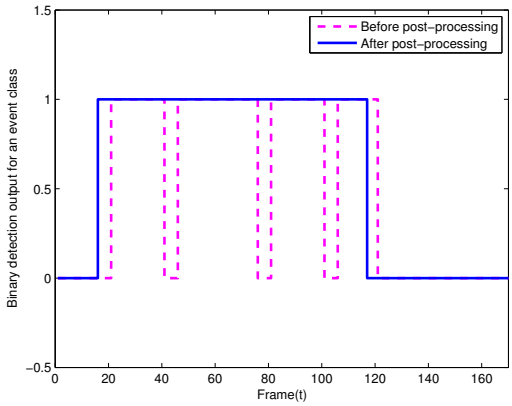


Fig. 5: Median filtering based post-processing method.

a greater problem occurs if the DNN *learns* these instances as belonging to a specific class with rather noise-like spectrum, such as the sound of rain or the wind on trees.

This causes some undesired intermittent behavior and *noise* in the DNN detection probabilities.

In order to filter this noise and smoothen the outputs in the testing stage, a median filtering based post-processing approach is implemented. The source presence predictions $\hat{\mathbf{y}}_t$ are obtained from the output layer of the DNN and then binarized by using a certain threshold value to give the binary estimation vector $\hat{\mathbf{z}}_t$. For each frame, the post-processed output $\hat{\mathbf{z}}_t$ is obtained by taking the median of the binary outputs in a 10-frame window as

$$\hat{\mathbf{z}}_t = \begin{cases} 1, \text{median}(\mathbf{z}_{(t-9):t}) = 1 \\ 0, \text{otherwise} \end{cases} \tag{6}$$

The method is continued by sliding this 10-frame window by 1 when every new frame is processed through the DNN. The effect of the median filtering on the detection outputs is illustrated in Figure 5.

## V. Evaluation

The proposed method is evaluated on realistic recordings from everyday contexts and compared with the baseline system. In addition, three different features are experimented individually: Mel-band energies, log Mel-band energies and MFCCs. The accuracy for various polyphony levels and the effect of post-processing is also investigated.

### A. Acoustic Data

The evaluation sound database contains recordings from various everyday environments. The same database has been previously used in different experiments on sound event detection [1], [5], [6]. It consists of a total of 103 recordings and each of them are 10 to 30 minutes long. The total duration of the recordings is 1133 minutes. Recordings were done using 44.1 kHz sampling rate and 24-bit resolution.

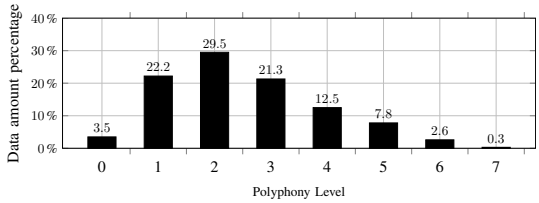


Fig. 6: The percentage of the amount of the sound material as a function of the polyphony level.

The recordings are collected from 10 different contexts: basketball match, beach, public bus, car, hallway, office, restaurant, shop, street and stadium. For each context, 8 to 14 recordings are present.

There are 61 different event classes categorized in this database. The start and end times of the events are manually annotated from the recordings. Some of the events included in the database are ”brakes squeaking”, ”cheering”, ”referee whistle” etc. In each context, 9 to 16 events are present. Some of the events can be found in multiple contexts (e.g. ”speech”) and some of the events are context specific (e.g. ”ball hitting the floor”). The total duration of each event in the database can be found in Figure 7. This database is a valuable source considering the lack of publicly available environmental polyphonic sound databases in the field. Figure 6 illustrates the amount of frames with different polyphony levels in the whole database.

### B. Evaluation Procedure

As the evaluation metric, F1 score is calculated inside non-overlapping one-second blocks. If an event

- is detected in one of the instances inside a block and it is also present in the same block of the annotated data, that event is regarded as correctly detected.
- is *not* detected in any of the instances inside a block but it is present in the same block of the annotated data, that event is regarded as missed.
- is detected in one of the instances inside a block but it is *not* present in the same block of the annotated data, that event is regarded as false alarm.

For each one-second block, the number of correct, missed and false alarm events are accumulated. Precision and recall are calculated according to these variables as

$$\textit{precision} = \frac{\textit{correct}}{\textit{correct} + \textit{false alarm}} \tag{7}$$

$$\textit{recall} = \frac{\textit{correct}}{\textit{correct} + \textit{missed}} \tag{8}$$

For each block, these two metrics are combined as their harmonic mean, *F1 score*, which can be formulated as

$$\textit{F1 score} = \frac{2 \times \textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}} \tag{9}$$

The results are presented by taking the average F1 scores of the one second blocks which correspond to the specific

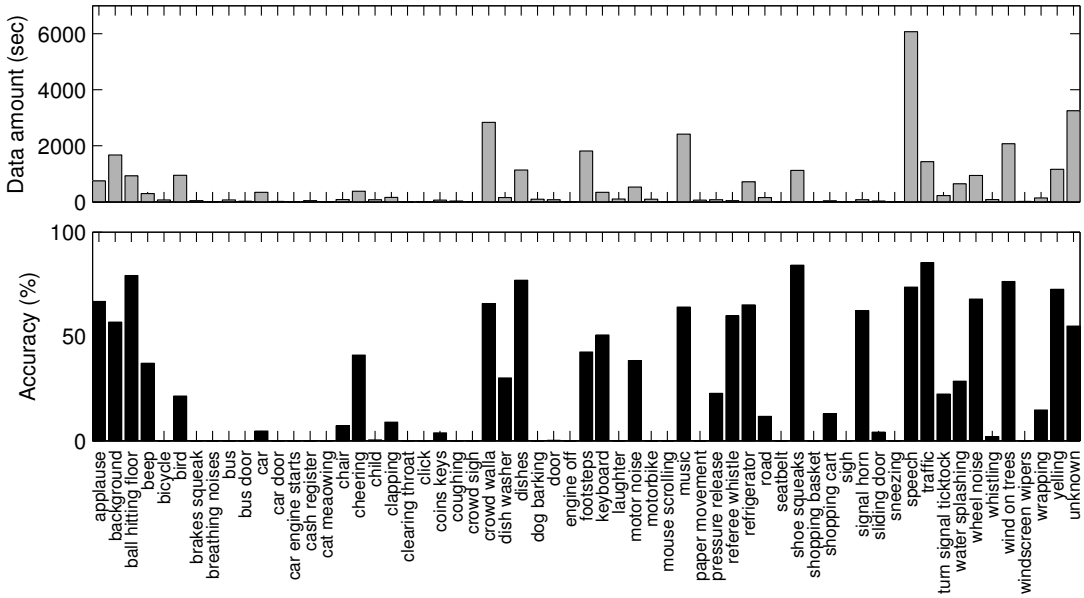


Fig. 7: The amount of annotated data (in seconds) and the accuracy for each sound event.

concept (context, polyphony level etc.) F1 score is referred as the *accuracy* throughout the rest of the paper.

Calculating the accuracy in one-second blocks is plausible for three different reasons. Firstly, the aim of the sound event detection is to detect an event with certainty when it happens, rather than finding the exact start and end time of the event with very high precision. Secondly, monitoring the outputs in every second and calculating the accuracy gives a reasonable time resolution without losing crucial information. Lastly, as noted in Section IV, the annotations have rather coarse time resolution. Therefore, in some cases they do not exactly match the time frames that they are annotated with, but they are nevertheless found in a one-second range. One-second block evaluation helps to compensate these minor mismatches in the annotations.

### C. Results

The proposed system is evaluated with stratified five-fold cross-validation. Once the features are extracted from all the recordings in the database, the feature data set is divided into five non-overlapping folds and one fold is used in the development stage for determining parameters of the DNN. The results from the other folds are averaged and presented. The grid search range and the final selected value for some essential DNN parameters during the development stage are presented in Table I. Log Mel-band energies are used as features in all the experiments, except the varying feature experiment given in Table II.

Our system, DNN with 2 hidden layers of 800 units each, log Mel-band energy features with 5-frame context

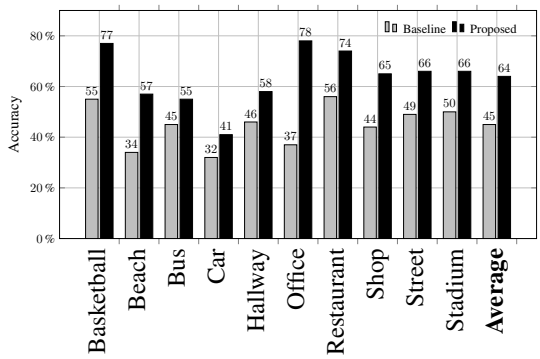


Fig. 8: Context-wise detection accuracies for proposed system with a comparison to the baseline system.

TABLE I: Grid search range for essential DNN parameters and the final values used in the experiments.

| Parameter | Range | Final |
|---|---|---|
| Learning rate | 0.001 - 1.0 | 0.02 |
| # hidden layers | 2 - 5 | 2 |
| # hidden units (in each layer) | 100 -1000 | 800 |
| Initial weight range | $\pm$0.001 - $\pm$0.05 | $\pm$0.001 |
| Context window length | 1 - 13 | 5 |

TABLE II: Overall detection accuracies with different input features before and after post-processing (PP).

| Feature | Before PP | After PP |
|---|---|---|
| MFCCs | 53.5 | 56.8 |
| Mel-band energies | 56.0 | 60.4 |
| Log Mel-band energies | 57.2 | 61.7 |

window, is evaluated against the baseline system [1], which is the state-of-the-art method for the polyphonic detection. The baseline method consists of decomposing the audio into different streams by non-negative matrix factorization. For each audio stream, sound event detection is done using MFCCs as features and HMM as a classifier. In our experiments, we observed that DNNs do not necessarily require this kind of sound source separation based pre-processing to determine how many events are active in a time instance. As illustrated in Figure 8, the proposed system outperforms the baseline method by a huge margin. Depending of the context, proposed method offers an increase in accuracy between 9-39% among different contexts and 19% units average increase. Due to the natural diversity of each context, the variance of the accuracy between contexts is quite high.

The relationship between the amount of annotated data and the accuracy for each sound event is illustrated in Figure 7. Differing from other experiments, the accuracy is calculated for each single sound event and therefore represents the single label accuracy. There is a clear correlation between the amount of data and the accuracy for each event. This brings the fact that DNNs require large training databases to learn the mappings between the features and the sound events. This also shows that there is still room for improvement in accuracy once the audio database is expanded. On a related note, we also investigated using shorter frame lengths and/or higher overlap in order to create more instances for the DNN learning. However, this increased the detrimental effect of the erroneous annotations without providing a significant boost on the accuracy. Nevertheless, the effect of the frame length is not investigated exhaustively in this work and therefore out of the scope of this paper.

The overall accuracy of the model trained with different features are presented in Table II. Mel-band energies and log Mel-band energies are calculated in 40 Mel-bands and the number of static MFCC coefficients are chosen to be 16, a standard value in event detection methods. The topology of DNN is kept fixed (except the number of inputs) while using different features in order to make a valid comparison. There is a slight increase in accuracy for Mel-band and log Mel-band energies over MFCCs. This can be explained with the loss of information caused by selecting the first few coefficients after the Discrete Cosine Transform (DCT) [13]. Another point would be that the sum of the MFCCs of different sound sources are not equal to the MFCCs of the mixture of these sources.

The detection predictions $\hat{\mathbf{y}}_t$ are binarized with various thresholds and the accuracy for each polyphony level (*i.e.*, the number of simultaneously active sound events) is given in Table III. For the majority of the levels, the accuracy takes its highest value around the threshold 0.5, which suits with the default guessing of a threshold for a prediction between 0 and 1. The accuracy is higher for high threshold values in the low polyphony levels. This can be explained by the fact that the prominent sound events have very high prediction value, *i.e.*, probability in lower polyphony levels and using high threshold effectively clears the non-present sound events with lower probability. On the other hand, the accuracy is higher for low threshold values in the high polyphony levels. Since the activation function for the output layer of the DNN is *logistic sigmoid*, the detection probabilities are bounded between 0 and 1. However, the sum of the predictions for each sound event is not bounded at all and this sum increases when the polyphony level is increased. When two events with similar spectra are simultaneously active, they share a lower probability compared to the case that only one of them is active. The detection probabilities are distributed over a higher number of sound events in high polyphony levels. Therefore, a lower threshold is required in order to detect multiple sound events.

TABLE III: F1 scores for various binarizing thresholds and polyphony levels for the proposed system after post-processing.

| | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|
| **1** | 44.6 | 51.8 | 57.6 | 62.5 | 66.0 | 66.9 | 64.5 | 56.3 |
| **2** | 51.8 | 56.8 | 59.7 | 61.1 | 61.1 | 58.9 | 54.2 | 44.4 |
| **3** | 55.7 | 59.3 | 60.9 | 61.3 | 59.8 | 56.6 | 50.4 | 39.1 |
| **4** | 60.3 | 63.3 | 64.5 | 64.5 | 63.2 | 60.1 | 54.6 | 42.8 |
| **5** | 64.5 | 65.5 | 65.2 | 64.5 | 62.5 | 59.3 | 53.3 | 40.7 |

The detection accuracy as a function of the polyphony level is given in Figure 9. Binarizing threshold value is fixed at 0.5 for all polyphony levels. The effect of median filtering-based post-processing is clearly visible, especially for lower polyphony levels. Post-processing offers a great boost on the accuracy for lower polyphony levels, *i.e.*, when less events are simultaneously active. As explained in Section IV, post-processing compensates the DNN's tendency to map the frames with low activity, which are found in low polyphony levels, to the non-present sound events. These frames hardly ever appear in very high polyphony levels, hence the effectiveness of the post-processing diminishes. The mapping of low activity frames with non-present events also explains the decreased accuracy in lower polyphony levels before the post-processing.

## VI. Conclusions

In this paper, using multi label DNNs for polyphonic sound event detection in realistic environments was proposed. Multi label DNN classification with median filtering-based post-processing was observed to be able to detect overlapping sound events with high accuracy. The proposed method outperforms the baseline method by 19%. Spectral domain features from short time frames of audio material

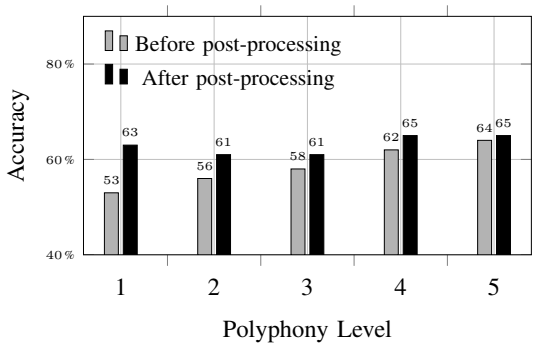


Fig. 9: Detection accuracy vs. polyphony level for the proposed system before and after the post-processing.

were extracted and used as input for the DNN. A post-processing method is proposed which increases the detection accuracy, especially when the number of simultaneously active events in a frame is lower than 4. It is also observed that using a higher binarizing threshold for low polyphony levels provide a better detection accuracy and *vice versa*. For future work, implementing better post-processing methods to handle the noise in the DNN output is planned. Training method extensions such as momentum and weight decay can also be implemented. Investigating more informative features for higher accuracy and robustness is also possible. Another future work direction would be to do the multi label DNN classification for each context separately, which requires a significant amount of data for each context and the reason why we choose the context independent approach in the first place.

## REFERENCES


[1] T. Heittola, A. Mesaros, T. Virtanen, and M. Gabbouj, "Supervised model training for overlapping sound events based on unsupervised source separation," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013, pp. 8677–8681.

[2] D. Zhang and D. Ellis, "Detecting sound events in basketball video archive," *Dept. Electronic Eng., Columbia Univ., New York*, 2001.

[3] S. Chu, S. Narayanan, C. Kuo, and M.J. Mataric, "Where am I? scene recognition for mobile robots using audio features," in *IEEE Int. Conf. Multimedia and Expo (ICME)*. IEEE, 2006, pp. 885–888.

[4] A. Harma, M.F. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in *IEEE Int. Conf. Multimedia and Expo (ICME)*. IEEE, 2005, pp. 4–pp.

[5] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2010, pp. 1267–1271.

[6] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, pp. 1, 2013.

[7] O. Dikmen and A. 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)*. IEEE, 2013, pp. 1–4.

[8] J. Dennis, H.D. Tran, and E.S. 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.

[9] M. Zhang and Z. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Trans. Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[10] M.R. Boutell, J. Luo, X. Shen, and C.M. Brown, "Learning multi-label scene classification," *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[11] A. McCallum, "Multi-label text classification with a mixture model trained by EM," in *Workshop on Text Learning*, 1999, pp. 1–7.

[12] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic events using deep neural networks," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2014.

[13] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T.N. Sainath, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[14] Pawel Swietojanski, Jinyu Li, and Jui-Ting Huang, "Investigation of maxout networks for speech recognition," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2014, pp. 7649–7653.

[15] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2014, pp. 1562–1566.

[16] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[17] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. Int. Conf. Machine Learning (ICML)*, 2013, pp. 1319–1327.

[18] G.E. Dahl, T.N. Sainath, and G.E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8609–8613.

[19] S. Kullback and R.A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, pp. 79–86, 1951.

# Publication II

Emre Çakır, Toni Heittola, Heikki Huttunen, Tuomas Virtanen. "Multi-label vs. Combined Single-label Sound Event Detection with Deep Neural Networks", in *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO).* Nice, France, September 2015, pp. 2551-2555.

©2015 IEEE. Reprinted, with permission, from E. Çakır, T. Heittola, H. Huttunen, and T. Virtanen, Multi-label vs. Combined Single-label Sound Event Detection with Deep Neural Networks, Proceedings of the 23rd European Signal Processing Conference, September 2015.

# MULTI-LABEL VS. COMBINED SINGLE-LABEL SOUND EVENT DETECTION WITH DEEP NEURAL NETWORKS

*Emre Cakir, Toni Heittola, Heikki Huttunen and Tuomas Virtanen*

Department of Signal Processing, Tampere University of Technology, Finland

**ABSTRACT**

In real-life audio scenes, many sound events from different sources are simultaneously active, which makes the automatic sound event detection challenging. In this paper, we compare two different deep learning methods for the detection of environmental sound events: combined single-label classification and multi-label classification. We investigate the accuracy of both methods on the audio with different levels of polyphony. Multi-label classification achieves an overall 62.8% accuracy, whereas combined single-label classification achieves a very close 61.9% accuracy. The latter approach offers more flexibility on real-world applications by gathering the relevant group of sound events in a single classifier with various combinations.

***Index Terms***— Sound event detection, deep neural networks, multi-label classification, binary classification, audio analysis

## 1. INTRODUCTION

Sound event detection (SED) systems aim to recognize and distinguish particular events related to human, nature or machine presence. In realistic environments, there are often multiple sound sources and the sound events originating from them can overlap in time. Birds singing, footsteps, motorbike engine etc. can be given as examples for the sound events in realistic environments. SED systems tackle the problem for two different cases: monophonic and polyphonic detection. In monophonic detection, the aim is to find the prominent event in the sound data. It is used in video keyword tagging [1] and real-life event and context detection [2, 3].

Polyphonic SED is capable of detecting multiple sound events in the same time instance of the sound data. Each instance is associated with a set of labels, *i.e.*, the labels of the sound events that are active in the given instance. The aim is to map each instance with its associated label set. The number of sound events active in an instance is not known *a priori*, which introduces a different level of complexity in detection.

Polyphonic SED systems require multi-label classification, which is not widely experimented in audio information retrieval tasks. Generalized Hough transform (GHT) voting system has been used to recognize overlapping sound events by summing up the local spectrogram keypoint information to produce onset hypotheses [4]. In [5], non-negative matrix factorization (NMF) has been used to first decompose the audio into streams and then recognize a single event from each stream by using prominent stream selection or stream elimination. In our previous work we proposed to use multi-label deep neural networks (DNN) for polyphonic SED and showed that it exceeds the state-of-the-art NMF + hidden Markov model (HMM) based approach [5] in accuracy [6].

DNNs are classifiers that are capable of extracting high level representations of their inputs through the multiple hidden layers. This has been found to provide better discrimination capability in certain pattern recognition tasks. Deep learning methods have recently given state-of-the-art results for many applications in environmental SED [2, 6] and speech recognition [7].

In this paper, we explore the use of DNNs in environmental SED with two different approaches: multi-label (ML) and combined single-label (CSL) methods. The proposed methods are illustrated in Figure 1. First, we train DNNs with multi-label outputs with polyphonic material in a supervised setting. Then, we train several DNNs with single-label outputs again with the same polyphonic material in a supervised setting. We combine the outputs of the single-label DNNs to obtain a multi-label output for each time instance. In [8], it is claimed that decomposing a multi-label classification into several binary classification problems will lose the correlation information between different labels in a single instance. However, the flexibility of making different sets of labels for different applications can be valuable and useful at the expense of slightly decreased accuracy for some applications, especially in SED systems. Moreover, using a set of single-label classifiers allows dynamic inclusion of new labels by training classifiers only for the new sound events instead of re-training the complete framework. To the best of our knowledge, this is the first work that compares these two deep learning approaches on polyphonic SED. Both methods are experimented on realistic sound material with single element.

The organization of this paper is as follows. The problem is stated and the DNN input and target output is explained in Section 2. The methodology, including ML and CSL DNN classification methods, are explained in detail in Section 3. The experiment material, evaluation procedure and experimental results are given in Section 4. Finally, comments and conclusions are given in Section 5.

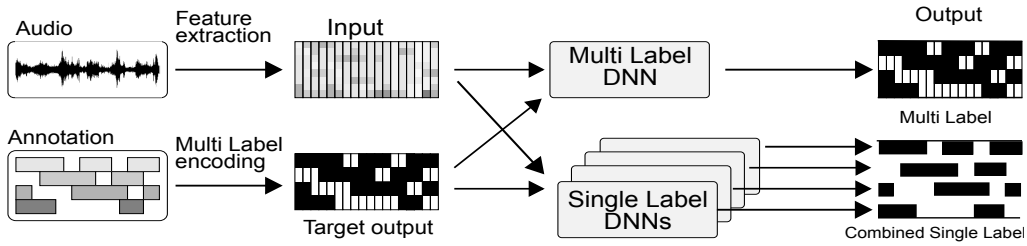


**Fig. 1**: Framework for the proposed CSL DNN and ML DNN classification methods.

## 2. PROBLEM STATEMENT

The goal of polyphonic SED is to estimate the start and end times of sound events in an audio signal, and classify the events into their predefined classes. When the audio is processed in short frames, this can be viewed as a multi-label classification problem. Multi-label learning tackles the problems where each instance in the training set can be associated with multiple labels. When it comes to multi-label environmental SED, the sound data typically contains overlapping events, *e.g*, a sound recording taken from a street may contain traffic noise, speech and the rain sound all at the same time.

### 2.1. Audio Representation

In order to do robust classification on the polyphonic material, one should choose the features that makes a good discrimination over the possible outcomes. Mel band energies are used as audio features in this work, since they have been proved to obtain better performance over the traditional Mel frequency cepstral cofficients (MFCC) in polyphonic SED and speech recognition [6, 7]. The reasoning for this can be that DNNs do not especially require their inputs to be uncorrelated and by applying discrete cosine transform (DCT), MFCCs discard some information from the audio material [9]. The recordings are first amplitude normalized, divided into frames and then short-time Fourier transform (STFT) is applied on 50 ms time frames with 50% overlap. Mel filterbank with 40 Mel bands is used to extract the feature vector $\mathbf{u}_t$ in each time frame, where $t$ denotes the position in the domain.

In order to make use of the dynamic properties of the audio, the feature vectors are concatenated with their two preceding and two succeeding feature vectors. This method is known as *context windowing*. Concatenated feature vector $\mathbf{x}_t$ is obtained as $\mathbf{x}_t = [\mathbf{u}_{t-2}^{\mathrm{T}}\ \mathbf{u}_{t-1}^{\mathrm{T}}\ \mathbf{u}_t^{\mathrm{T}}\ \mathbf{u}_{t+1}^{\mathrm{T}}\ \mathbf{u}_{t+2}^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathbf{u}_t$ denotes the extracted feature vector. The concatenated feature vector $\mathbf{x}_t$ is used as a single training instance for the DNN.

### 2.2. DNN Target Output

The training of the network is performed in a supervised setting. The start and end times for each sound event in a recording are manually annotated each time they occur in the recording. For each time frame, a target output vector $\mathbf{y}_t$ of length $N$ is obtained, where $N$ is the total number of possible sound events. The elements of the target vector $\mathbf{y}_t$ are binary and determined as

$$y_t(l) = \begin{cases} 1, & \text{if } l^{th} \text{ event is active in frame } t \\ 0, & \text{if } l^{th} \text{ event is not active in frame } t \end{cases} \quad (1)$$

where $y_t(l)$ is the $l^{\text{th}}$ entry of target output vector $\mathbf{y}_t$ and $1 \leq l \leq N$.

## 3. METHODOLOGY

We consider two methods for encoding the presence of simultaneous events in an audio recording. One method is to train a single-label classifier for each label $l$ and then combine the outputs from each classifier to obtain a multi-label output. Second method is to train a multi-label classifier, which produces a multi-label detection output vector.

### 3.1. Combined Single-Label DNN classification

For the CSL DNN classification, each label $l$ is trained and tested with a different DNN, independent from other labels. The input features are extracted from polyphonic sound signals. The reasons for using polyphonic signals is as follows. Firstly, even for single-label classification, the sound events are hardly ever isolated in a realistic environment and it is difficult to separate signals produced by individual sources. Secondly, using polyphonic data makes the comparison between the CSL DNN and the ML DNN methods easier to interpret and analyze.

CSL DNN provides significant flexibility on real-world applications. To illustrate, if the number of sound events in a database is $N$, then $N$ different models can be trained and grouped together in various combinations depending which of the classes are of interest in a certain application. Besides, new classes can be easily added to the overall CSL DNN system by training a single-label DNN for the new class with the additional database.

The single-label DNN architecture is composed of an input layer, two or more hidden layers and output layer with a

single output unit. Fully-connected feed-forward DNNs are used in this work. Starting from $\mathbf{h}^1 = \mathbf{x}$, the outputs $\mathbf{h}^k$ of the units for layer $k$ are calculated as

$$\mathbf{g}^k = \mathbf{W}^k \mathbf{h}^{k-1} + \mathbf{b}^k, 2 \leq k < M \quad (2)$$

$$\mathbf{h}^k = f(\mathbf{g}^k) \quad (3)$$

where $\mathbf{W} \in \mathbb{R}^{D \times S}$ is the weight matrix between layers $k-1$ and $k$, $D$ and $S$ are the number of units for layers $k-1$ and $k$, respectively, $\mathbf{b} \in \mathbb{R}^S$ is the bias vector for layer $k$, $f(\cdot)$ is the activation function applied element-wise on the output of each unit in layer $k$, and $M$ is the total number of layers in the DNN. For the hidden layer activation functions, *maxout* [10] function is used. Instead of applying a conventional non-linearity on the weighted sum $\mathbf{g}^k$, maxout function groups the weighted sums and passes the maximum to $\mathbf{h}^k$, increasing the sparsity of the gradients and preventing the network suffering from the vanishing gradients since the activation outputs are unbounded [11]. For the output layer activation function, the more conventional logistic sigmoid function is chosen. Since the sigmoid activation function output $h^M$ is bounded between 0 and 1, it is possible and logical to interpret the DNN output as the detection probability. For each training instance $\mathbf{x}_t$, the CSL DNN output with single element $h^M$ is used as the source-presence prediction $\hat{y}_t$.

Each single-label DNN is trained with the corresponding target output $y_t(l)$ for label $l$. In order to estimate the distance between the source-presence prediction and the target output for label $l$, cross-entropy cost function $C_l(\hat{y}_t, y_t(l))$ is calculated as

$$C_l(\hat{y}_t, y_t(l)) = -y_t(l)\log(\hat{y}_t) - (1 - y_t(l))\log(1 - \hat{y}_t) \quad (4)$$

where $y_t(l)$ is either 0 or 1 and $\hat{y}_t \in [0, 1]$. $C_l(\hat{y}_t, y_t(l))$ is guaranteed to be non-negative and when $y_t(l)$ and $\hat{y}_t$ are closer to each other, it goes closer to zero. Therefore, cross-entropy cost function is to be minimized by updating the weight $\mathbf{W}$ and bias $\mathbf{b}$ vectors. For this purpose, stochastic gradient descent algorithm (SGD) is used. The gradients in each layer are computed using the backpropagation algorithm [12].

When the separate training for each label is finished, the test instances are evaluated by the single-label DNNs and the source-presence predictions $\hat{y}_t$ are obtained. The source presence-predictions from each single-label DNN are combined in the multi-label vector $\hat{\mathbf{y}}_t = [\hat{y}_t(1)\ \hat{y}_t(2)...\hat{y}_t(N)]$. Then, $\hat{\mathbf{y}}_t$ is binarized by thresholding with a certain constant, leading to a binary estimation vector $\mathbf{z}_t$ of length $N$. The effects of the binarizing threshold is examined in Section 4.

### 3.2. Multi-Label DNN classification

ML DNN training differs from the CSL DNN training only in the way that the number of units in the output layer is equal to the number of sound events $N$, therefore we get the source-presence prediction vector $\hat{\mathbf{y}}_t$ of length $N$ for each frame $t$. This leads to another information to be learned: the correlation structure between the events. Some of the events may appear together in a large number of training instances and it can be a valuable information for the DNN.

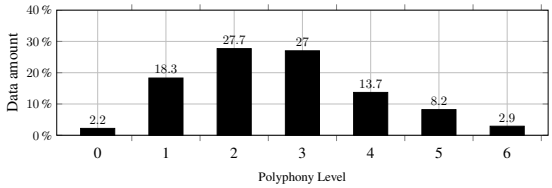


**Fig. 2**: The percentage of the amount of the test material as a function of the polyphony level.

Instead of calculating the cross-entropy cost function for a single-label, ML DNN computes the cost function as

$$C(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\mathbf{y}_t \cdot \log(\hat{\mathbf{y}}_t) - (1 - \mathbf{y}_t) \cdot \log(1 - \hat{\mathbf{y}}_t) \quad (5)$$

where the operator $(\cdot)$ denotes the dot product and the logarithm operator is applied element-wise. The cost value is the sum of the costs over each label and therefore depends on the source-presence predictions for each label $l$.

### 3.3. Post-processing

Our experiments with realistic audio material showed that environmental sound events typically have some short bursts of less active periods. To illustrate, a dog gives a small break to breathe before each bark, or the footsteps make sounds periodically. Since the annotation of the audio material is done with a rather coarse time resolution, these less active bursts are mapped to a sound event of which they do not possess the spectral characteristics. This results with a rather noisy behaviour in DNN outputs.

A median filtering based post-processing approach is implemented to filter this noise and smoothen the outputs in the testing stage for both CSL DNN and ML DNN. For each frame, the post-processed output $\hat{\mathbf{z}}_t$ is obtained by taking the median of the binary outputs $\mathbf{z}_t$ in a 10-frame window (corresponds to 250 ms of audio) as

$$\hat{\mathbf{z}}_t = \begin{cases} 1, \text{if median}(\mathbf{z}_{(t-9):t}) = 1 \\ 0, \text{otherwise} \end{cases} \quad (6)$$

The method is applied on each label separately and continued by sliding this 10-frame window when every new frame is processed through the DNN.

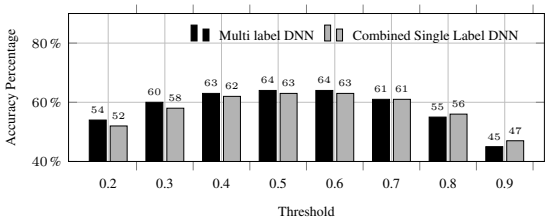


**Fig. 3**: Detection accuracy vs. binarizing threshold for ML DNN and CSL DNN.

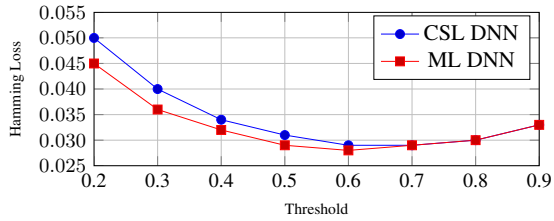


**Fig. 4**: Hamming loss vs. threshold for CSL DNN and ML DNN classification.

## 4. EXPERIMENTS AND RESULTS

### 4.1. Sound Material and Setting

The evaluation of both CSL and ML DNN methods are performed on a sound database collected from highly realistic everyday environments. Recordings from 10 different environments, such as beach, bus, street etc. are used to gather a database of 1133 minutes. From the recordings, 61 most prominent events, such as clapping, dogs barking, cash register beep etc. are selected to be evaluated. The recordings have varying number of active sound events in each instance, *i.e.*, the frames have varying polyphony levels. The amount of test material according to their polyphony levels are illustrated in Figure 2. The label cardinality, *i.e.*, the average number of active events in each frame is 2.55. The database is divided into non-overlapping groups as 60% training, 20% test and 20% validation sets. Validation set is not used in training and it is required to determine the optimum parameters without overfitting the network on the training set. More detailed information on the sound database can be found in [13].

DNN hyper-parameters such as learning rate, hidden unit number, initial weight and bias range etc. are selected by doing a grid search over possible values to get the best accuracy on the validation set. The best performance is obtained with two hidden layers of 800 units for ML DNN and two hidden layers of 400 units for CSL DNN. The learning rates for both methods are 0.02.

### 4.2. Evaluation Metric

The methods are evaluated by using two different metrics that are commonly used in multi-label evaluation. First one is the block-wise F1 score evaluation metric. A sound event is regarded as correctly detected if it is marked as present in any instance of the time block and if it is also present in the annotations of the time block. Missed and wrongly detected events are calculated in the same manner. This approach fits well with the goal of environmental SED, since it is rather interested in detecting the event rather than the exact start and end times. Precision and recall are calculated according to the number of correctly detected, missed and wrongly detected events. F1 score, the harmonic mean of the precision and recall, is calculated in non-overlapping one second blocks. The final F1 score is calculated by averaging the F1 scores in the one second time blocks of the test dataset and presented as the accuracy percentage.

The second multi-label evaluation metric is Hamming loss [14]. It evaluates how many times a frame is misclassified. It implements exclusive-or (*xor*) operation between binary estimation vector $\hat{\mathbf{z}}_t$ and target output vector $\mathbf{y}_t$ as

$$\text{Hamming loss}(\hat{\mathbf{z}}, \mathbf{y}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}(\hat{\mathbf{z}}_t \,\Delta\, \mathbf{y}_t) \tag{7}$$

where $T$ is the number of time frames, $N$ is the number of sound events and the operator $\Delta$ gives the symmetric difference between $\hat{\mathbf{z}}_t$ and $\mathbf{y}_t$ as

$$\hat{\mathbf{z}}_t(l) \,\Delta\, \mathbf{y}_t(l) = \begin{cases} 0, & \text{if } \hat{\mathbf{z}}_t(l) = \mathbf{y}_t(l) \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

### 4.3. Results

While converting the DNN outputs $\hat{\mathbf{y}}_t$ into binary form as $\mathbf{z}_t$, several threshold values have been experimented. For various thresholds, the average F1 score is presented as the accuracy percentage for ML DNN and CSL DNN in Figure 3. Both methods provide a huge improvement over the state-of-the-art NMF+HMM-based method in [5], which provides 44.9% accuracy on the same database. For both methods, the accuracy takes its maximum value around threshold 0.5, which indicates that DNN outputs make a balanced probability distribution estimation between 0 and 1. Hamming loss results from Figure 4 also supports this theory. Hamming losses for both methods reach to their minimum around threshold 0.6 (note that they are very close for thresholds above 0.7). ML DNN classification provides a 2-3% better accuracy compared to CSL DNN for low threshold values, but the situation reverses for higher threshold values. This can be explained with the fact that the whole activity of a single frame is bundled in one single DNN output for CSL DNN, whereas in ML DNN, it is distributed in all the events. Therefore, in a polyphonic

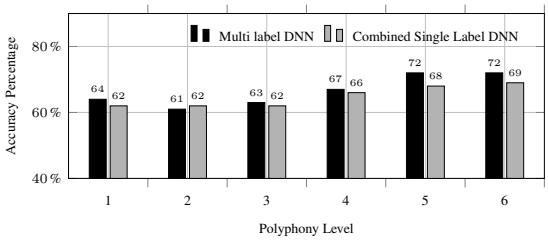


**Fig. 5**: Detection accuracy vs. polyphony level.

frame, for each active event, the source-presence predictions are higher for CSL DNN than ML DNN and better discrimination is obtained with high threshold values.

The detection accuracy is calculated separately for different levels of polyphony and examined in Figure 5. The binarizing threshold is set to 0.5. When the accuracy is averaged among the polyphony levels according to the data amount for each polyphony level, CSL DNN achieves an overall 61.9% accuracy, while ML DNN achieves 62.8% accuracy. CSL DNN classification provides very similar accuracy compared to ML DNN, regardless of the polyphony level. The results show that decomposing a multi-label sound event classification problem into multiple single-label problems do not suffer from losing the correlation structure between the labels.

## 5. CONCLUSIONS AND COMMENTS

In this paper, two different deep learning methods are proposed and compared for polyphonic SED in real-life environments. The first method consists of using a multi-label DNN for classification of all possible sound events, whereas the second method uses a single-label DNN for each single sound event and combines the outputs of each DNN for a single time frame. Although the hypothesis was that CSL DNN would be affected from losing the correlation information, it provides very similar accuracy compared to ML DNN. CSL DNN also presents multiple implementation options by grouping different event models together. For the future work, it would be interesting to apply CSL DNN on other multi-label classification problems. Also, context dependent CSL DNN methods can be experimented by grouping the CSL DNN models for the events that are likely to occur together, thus creating a single classifier for a certain context. Finally, an interesting path would be to apply other multi-label learning methods on sound event detection and see if our conclusion for multi-label DNN learning is extensible for other approaches.

## REFERENCES

[1] D. Zhang and D. Ellis, "Detecting sound events in basketball video archive," *Dept. Electronic Eng., Columbia Univ., New York*, 2001.

[2] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic events using deep neural networks," in *Proc. 22nd European Signal Processing Conference (EUSIPCO)*, 2014, pp. 506–510.

[3] S. Chu, S. Narayanan, and C-CJ Kuo, "Environmental sound recognition with time–frequency audio features," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1142–1158, 2009.

[4] J. Dennis, H.D. Tran, and E.S. Chng, "Overlapping sound event recognition using local spectrogram features and the generalised hough transform," *Pattern Recognition Letters*, vol. 34, no. 9, 2013.

[5] T. Heittola, A. Mesaros, T. Virtanen, and M. Gabbouj, "Supervised model training for overlapping sound events based on unsupervised source separation," in *Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 8677–8681.

[6] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multilabel deep neural networks," in *Int. Joint Conf. on Neural Networks (IJCNN)*, 2015, accepted.

[7] J. Li, D. Yu, J. Huang, and Y. Gong, "Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM," in *Spoken Language Technology Workshop*, 2012, pp. 131–136.

[8] M. Zhang and Z. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Trans. Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[9] G. Hinton, L. Deng, D. Yu, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[10] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *ICML*, 2013.

[11] Pawel Swietojanski, Jinyu Li, and Jui-Ting Huang, "Investigation of maxout networks for speech recognition," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2014, pp. 7649–7653.

[12] P. Werbos, *Beyond regression: New tools for prediction and analysis in the behavioral sciences*, Ph.D. thesis, Harvard Univ., Comm. Appl. Math., 1974.

[13] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Proc. 18th European Signal Processing Conference (EUSIPCO)*, 2010, pp. 1267–1271.

[14] R.E. Schapire and Y. Singer, "Boostexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2-3, pp. 135–168, 2000.

# Publication III

Emre Çakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, Tuomas Virtanen. "Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection", in *IEEE/ACM Transactions on Audio, Speech and Language Processing*, volume 25, issue 6, pp. 1291-1303, 2017.

©2017 IEEE. Reprinted, with permission, from E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection, IEEE/ACM Transactions on Audio, Speech and Language Processing, 2017.

# Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection

Emre Çakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen

***Abstract*—Sound events often occur in unstructured environments where they exhibit wide variations in their frequency content and temporal structure. Convolutional neural networks (CNN) are able to extract higher level features that are invariant to local spectral and temporal variations. Recurrent neural networks (RNNs) are powerful in learning the longer term temporal context in the audio signals. CNNs and RNNs as classifiers have recently shown improved performances over established methods in various sound recognition tasks. We combine these two approaches in a Convolutional Recurrent Neural Network (CRNN) and apply it on a polyphonic sound event detection task. We compare the performance of the proposed CRNN method with CNN, RNN, and other established methods, and observe a considerable improvement for four different datasets consisting of everyday sound events.**

***Index Terms*—sound event detection, deep neural networks, convolutional neural networks, recurrent neural networks**

## I. Introduction

IN our daily lives, we encounter a rich variety of sound events such as dog bark, footsteps, glass smash and thunder. Sound event detection (SED), or acoustic event detection, deals with the automatic identification of these sound events. The aim of SED is to detect the onset and offset times for each sound event in an audio recording and associate a textual descriptor, *i.e.*, a label for each of these events. SED has been drawing a surging amount of interest in recent years with applications including audio surveillance [1], healthcare monitoring [2], urban sound analysis [3], multimedia event detection [4] and bird call detection [5].

In the literature the terminology varies between authors; common terms being sound event *detection*, *recognition*, *tagging* and *classification*. Sound events are defined with predetermined labels called sound event *classes*. In our work, sound event classification, sound event recognition, or sound event tagging, all refer to labeling an audio recording with the sound event classes present, regardless of the onset/offset times. On the other hand, an SED task includes both onset/offset detection for the classes present in the recording and classification within the estimated onset/offset, which is typically the requirement in a real-life scenario.

Sound events often occur in unstructured environments in real-life. Factors such as environmental noise and overlapping sources are present in the unstructured environments and they may introduce a high degree of variation among the sound events from the same sound event class [6]. Moreover, there can be multiple sound sources that produce sound events belonging to the same class, *e.g.*, a dog bark sound event can be produced from several breeds of dogs with different acoustic characteristics. These factors mainly represent the challenges over SED in real-life situations.

SED where at most one simultaneous sound event is detected at a given time instance is called *monophonic* SED. Monophonic SED systems can only detect at most one sound event for any time instance regardless of the number of sound events present. If the aim of the system is to detect all the events happening at a time, this is a drawback concerning the real-life applicability of such systems, because in such a scenario, multiple sound events are very likely to overlap in time. For instance, an audio recording from a busy street may contain footsteps, speech and car horn, all appearing as a mixture of events. An illustration of a similar situation is given in Figure 1, where as many as three different sound events appear at the same time in a mixture. A more suitable method for such a real-life scenario is *polyphonic* SED, where multiple overlapping sound events can be detected at any given time instance.

SED can be approached either as *scene-dependent* or *scene-independent*. In the former, the information about the acoustic scene is provided to the system both at training and test time, and a different model can therefore be trained for each scene. In the latter, there is no information about the acoustic scene given to the system.

Previous work on sound events has been mostly focused on sound event classification, where audio clips consisting of sound events are classified. Apart from established classifiers—such as support vector machines [1], [3]—deep learning methods such as deep belief networks [7], convolutional neural networks (CNN) [8], [9], [10] and recurrent neural networks (RNN) [4], [11] have been recently proposed. Initially, the interest on SED was more focused on monophonic SED. Gaussian mixture model (GMM) - Hidden Markov model (HMM) based modeling—an established method that has been widely used in automatic speech recognition—has been proposed to model individual sound events with Gaussian mixtures and detect each event through HMM states using Viterbi algorithm [12], [13]. With the emergence of more

G. Parascandolo and E. Cakir contributed equally to this work.

The authors are with the Department of Signal Processing, Tampere University of Technology (TUT), Finland e-mail: emre.cakir@tut.fi.

The research leading to these results has received funding from the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND.

G. Parascandolo has been funded by Google's Faculty Research award.

The authors wish to acknowledge CSC IT Center for Science, Finland, for computational resources.

The paper has a supporting website at
http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/

Manuscript received July 12, 2016 (revised January 19, 2016).

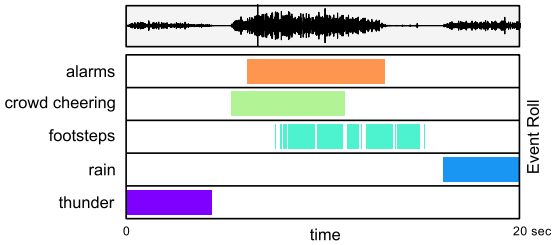


Fig. 1: Sound events in a polyphonic recording synthesized with isolated sound event samples. Upper panel: audio waveform, lower panel: sound event class activity annotations.

advanced deep learning techniques and publicly available real-life databases that are suitable for the task, polyphonic SED has attracted more interest in recent years. Non-negative matrix factorization (NMF) based source separation [14] and deep learning based methods (such as feedforward neural networks (FNN) [15], CNN [16] and RNN [11]) have been shown to perform significantly better compared to established methods such as GMM-HMM for polyphonic SED.

Deep neural networks [17] have recently achieved remarkable success in several domains such as image recognition [18], [19], speech recognition [20], [21], machine translation [22], even integrating multiple data modalities such as image and text in image captioning [23]. In most of these domains, deep learning represents the state-of-the-art.

Feedforward neural networks have been used in monophonic [7] and polyphonic SED in real-life environments [15] by processing concatenated input frames from a small time window of the spectrogram. This simple architecture—while vastly improving over established approaches such as GMM-HMMs [24] and NMF source separation based SED [25], [26]—presents two major shortcomings: (1) it lacks both time and frequency invariance—due to the fixed connections between the input and the hidden units—which would allow to model small variations in the events; (2) temporal context is restricted to short time windows, preventing effective modeling of typically longer events (*e.g.*, rain) and events correlations.

CNNs [27] can address the former limitation by learning filters that are shifted in both time and frequency [8], lacking however longer temporal context information. Recurrent neural networks (RNNs), which have been successfully applied to automatic speech recognition (ASR) [20] and polyphonic SED [11], solve the latter shortcoming by integrating information from the earlier time windows, presenting a theoretically unlimited context information. However, RNNs do not easily capture the invariance in the frequency domain, rendering a high-level modeling of the data more difficult. In order to benefit from both approaches, the two architectures can be combined into a single network with convolutional layers followed by recurrent layers, often referred to as convolutional recurrent neural network (CRNN). Similar approaches combining CNNs and RNNs have been presented recently in ASR [21], [28], [29] and music classification [30].

In this paper we propose the use of multi-label convolutional recurrent neural network for polyphonic, scene-independent sound event detection in real-life recordings. This approach integrates the strengths of both CNNs and RNNs, which have shown excellent performance in acoustic pattern recognition applications [4], [8], [9], [10], while overcoming their individual weaknesses. We evaluate the proposed method on three datasets of real-life recordings and compare its performance to FNN, CNN, RNN and GMM baselines. The proposed method is shown to outperform previous sound event detection approaches.

The rest of the paper is organized as follows. In Section II the problem of polyphonic SED in real-life environments is described formally and the CRNN architecture proposed for the task is presented. In Section III we present the evaluation framework used to measure the performance of the different neural networks architectures. In Section IV experimental results, discussions over the results and comparisons with baseline methods are reported. In Section V we summarize our conclusions from this work.

## II. Method

### A. Problem formulation

The aim of polyphonic SED is to temporally locate and label the sound event classes present in a polyphonic audio signal. Polyphonic SED can be formulated in two stages: sound representation and classification. In sound representation stage, frame-level sound features (such as mel band energies and mel frequency cepstral coefficients (MFCC)) are extracted for each time frame $t$ in the audio signal to obtain a feature vector $\mathbf{x}_t \in \mathbb{R}^F$, where $F \in \mathbb{N}$ is the number of features per frame. In the classification stage, the task is to estimate the probabilities $p(\mathbf{y}_t(k) \mid \mathbf{x}_t, \boldsymbol{\theta})$ for event classes $k = 1, 2, \ldots, K$ in frame $t$, where $\boldsymbol{\theta}$ denotes the parameters of the classifier. The event activity probabilities are then binarized by thresholding, *e.g.* over a constant, to obtain event activity predictions $\hat{\mathbf{y}}_t \in \mathbb{R}^K$.

The classifier parameters $\boldsymbol{\theta}$ are trained by supervised learning, and the target outputs $\mathbf{y}_t$ for each frame are obtained from the onset/offset annotations of the sound event classes. If class $k$ is present during frame $t$, $\mathbf{y}_t(k)$ will be set to 1, and 0 otherwise. The trained model will then be used to predict the activity of the sound event classes when the onset/offset annotations are unavailable, as in real-life situations.

For polyphonic SED, the target binary output vector $\mathbf{y}_t$ can have multiple non-zero elements since several classes can be present in the same frame $t$. Therefore, polyphonic SED can be formulated as a multi-label classification problem in which the sound event classes are located by multi-label classification over consecutive time frames. By combining the classification results over consecutive time frames, the onset/offset times for each class can be determined.

Sound events possess temporal characteristics that can be beneficial for SED. Certain sound events can be easily distinguished by their impulsive characteristics (*e.g.*, glass smash), while some sound events typically continue for a long time period (*e.g.* rain). Therefore, classification methods that can preserve the temporal context along the sequential feature vectors are very suitable for SED. For these methods, the input

features are presented as a context window matrix $\mathbf{X}_{t:t+T-1}$, where $T \in \mathbb{N}$ is the number of frames that defines the sequence length of the temporal context, and the target output matrix $\mathbf{Y}_{t:t+T-1}$ is composed of the target outputs $\mathbf{y}_t$ from frames $t$ to $t+T-1$. For the sake of simplicity and ease of notation, $\mathbf{X}$ will be used to denote $\mathbf{X}_{t:t+T-1}$—and similarly $\mathbf{Y}$ for $\mathbf{Y}_{t:t+T-1}$— throughout the rest of the paper.

### B. Proposed Method

The CRNN proposed in this work, depicted in Fig. 2, consists of four parts: (1) at the top of the architecture, a time-frequency representation of the data (a context window of $F$ log mel band energies over $T$ frames) is fed to $L_c \in \mathbb{N}$ convolutional layers with non-overlapping pooling over frequency axis; (2) the feature maps of the last convolutional layer are stacked over the frequency axis and fed to $L_r \in \mathbb{N}$ recurrent layers; (3) a single feedforward layer with sigmoid activation reads the final recurrent layer outputs and estimates event activity probabilities for each frame and (4) event activity probabilities are binarized by thresholding over a constant to obtain event activity predictions.

In this structure the convolutional layers act as feature extractors, the recurrent layers integrate the extracted features over time thus providing the context information, and finally the feedforward layer produce the activity probabilities for each class. The stack of convolutional, recurrent and feedforward layers is trained jointly through backpropagation. Next, we present the general network architecture in detail for each of the four parts in the proposed method.

*1) Convolutional layers:* Context window of log mel band energies $\mathbf{X} \in \mathbb{R}^{F \times T}$ is fed as input to the CNN layers with two-dimensional convolutional filters. For each CNN layer, after passing the feature map outputs through an activation function (rectified linear unit (ReLU) used in this work), non-overlapping max pooling is used to reduce the dimensionality of the data and to provide more frequency invariance. As depicted in Fig. 2, the time dimension is maintained intact (*i.e.* does not shrink) by computing the max pooling operation in the frequency dimension only—as done in [21], [31]—and by zero-padding the inputs to the convolutional layers (also known as *same* convolution). This is done in order to preserve alignment between each target output vector $\mathbf{y}_t$ and hidden activations $\mathbf{h}_t$.

After $L_c$ convolutional layers, the output of the CNN is a tensor $\mathcal{H} \in \mathbb{R}^{M \times F' \times T}$, where $M$ is the number of feature maps for the last CNN layer, and $F'$ is the number of frequency bands remaining after several pooling operations through CNN layers.

*2) Recurrent layers:* After stacking the feature map outputs over the frequency axis, the CNN output $\mathbf{H} \in \mathbb{R}^{(M \cdot F') \times T}$ for layer $L_c$ is fed to the RNN as a sequence of frames $\mathbf{h}_t^{L_c}$. The RNN part consists of $L_r$ stacked recurrent layers each computing and outputting a hidden vector $\mathbf{h}_t$ for each frame as

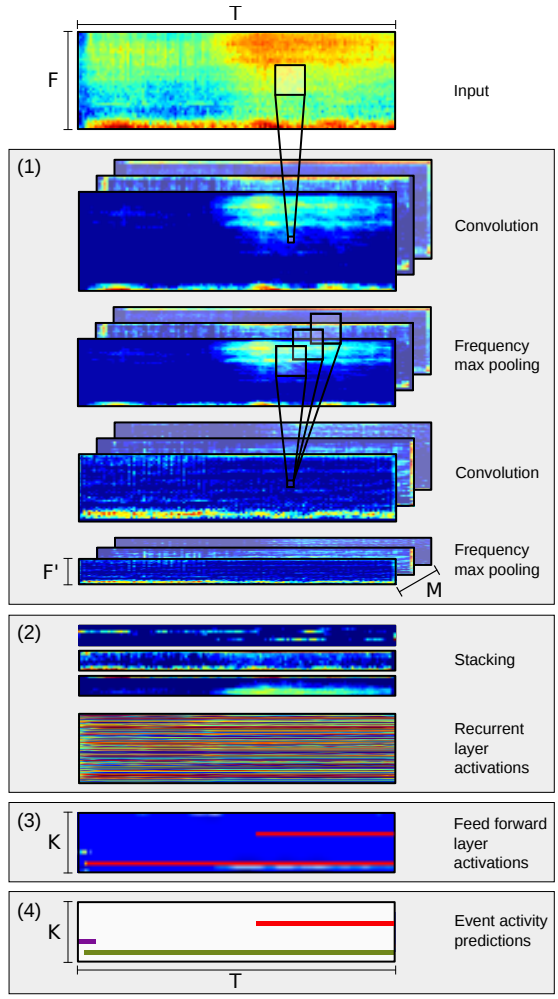


Fig. 2: Overview of the proposed CRNN method. (1): Multiple convolutional layers with max pooling in frequency axis, (2): The outputs of the last convolutional layer stacked over frequency axis and fed to multiple stacked recurrent layers, (3): feedforward layer as output layer and (4): binarization of event activity probabilities.

$$\begin{aligned} \mathbf{h}_t^{L_c+1} &= \mathcal{F}(\mathbf{h}_t^{L_c}, \mathbf{h}_{t-1}^{L_c+1}) \\ \mathbf{h}_t^{L_c+2} &= \mathcal{F}(\mathbf{h}_t^{L_c+1}, \mathbf{h}_{t-1}^{L_c+2}) \\ &\vdots \\ \mathbf{h}_t^{L_c+L_r} &= \mathcal{F}(\mathbf{h}_t^{L_c+L_r-1}, \mathbf{h}_{t-1}^{L_c+L_r}) \end{aligned} \tag{1}$$

The function $\mathcal{F}$, which can represent a long short term memory (LSTM) unit [32] or gated recurrent unit (GRU) [33], has two inputs: The output of the current frame of the previous layer (*e.g.,* $\mathbf{h}_t^{L_c}$), and the output of the previous frame of the current layer (*e.g.,* $\mathbf{h}_{t-1}^{L_c+1}$).

*3) Feedforward layer:* recurrent layers are followed by a single feedforward layer which will be used as the output layer of the network. The feedforward layer outputs are obtained from the last recurrent layer activations $\mathbf{h}_t^{L_c+L_r}$ as

$$\mathbf{h}_t^{L_c+L_r+1} = \mathcal{G}(\mathbf{h}_t^{L_c+L_r}), \tag{2}$$

where $\mathcal{G}$ represents a feedforward layer with sigmoid activation. Feedforward layer applies the same set of weights for the features extracted from each frame.

*4) Binarization:* The outputs $\mathbf{h}_t^{L_c+L_r+1}$ of the feedforward layer are used as the event activity probabilities for each class $k = 1, 2, ...K$ as

$$p(\mathbf{y}_t(k) \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) = \mathbf{h}_t^{L_c+L_r+1} \tag{3}$$

where $K$ is the number of classes and $\boldsymbol{\theta}$ represents the parameters of all the layers of the network combined. Finally, event activity predictions $\hat{\mathbf{y}}_t$ are obtained by thresholding the probabilities over a constant $C \in (0, 1)$ as

$$\hat{\mathbf{y}}_t(k) = \begin{cases} 1, & p(\mathbf{y}_t(k) \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) \geq C \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

*Regularization:* In order to reduce overfitting, we experimented with dropout [34] regularization in the network, which has proven to be extremely effective in several deep learning applications [18]. The basic idea behind dropout is to temporarily remove at training time a certain portion of hidden units from the network, with the dropped units being randomly chosen at each iteration. This reduces units co-adaptation, approximates model averaging [34], and can be seen as a form of data augmentation without domain knowledge. For the recurrent layers we adopted the dropout proposed in [35], where the choice of dropped units is kept constant along a sequence.

To speed up the training phase we train our networks with batch normalization layers [36] after every convolutional or fully connected layer. Batch normalization reduces the internal covariate shift—*i.e.*, the distribution of network activations during training—by normalizing a layer output to zero mean and unit variance, using approximate statistics computed on the training mini-batch.

*Comparison to other CRNN architectures:* The CRNN configuration used in this work has several points of similarity with the network presented in [21] for speech recognition. The main differences are the following: (i) We do not use any linear projection layer, neither at the end of the CNN part of the CRNN, nor after each recurrent layer. (ii) We use 5x5 kernels in all of our convolutional layers, compared to the 9x9 and 4x3 filters for the first and second layer respectively. (iii) Our architecture has also more convolutional layers (up to 4 instead of 2) and recurrent layers (up to 3 instead of 2). (iv) We use GRU instead of LSTM. (v) We use much longer sequences, up to thousands of steps, compared to 20 steps in [21]. While very long term context is not helpful in speech processing, since words and utterances are quite short in time, in SED there are several events that span over several seconds. (vi) For the experiments on CHiME-Home dataset we incorporate a new max pooling layer (only on time domain) before the output layer. Therefore, if we have $N$ mid-level features for $T$ frames of a context window, we end up with $N$ features for the whole context window to be fed to the output layer.

*CNNs and RNNs:* It is possible to see CNNs and RNNs as specific instances of the CRNN architecture presented in this section: a CNN is a CRNN with zero recurrent layers, and an RNN is a CRNN with zero convolutional layers. In order to assess the benefits of using CRNNs compared to CNNs or RNNs alone, in Section III we directly compare the three architectures by removing the recurrent or convolutional layer, *i.e.*, CNNs and RNNs respectively.

## III. Evaluation

In order to test the proposed method, we run a series of experiments on four different datasets. We evaluate the results by comparing the system outputs to the annotated references. Since we are approaching the task as scene-independent, on each dataset we train a single model regardless of the presence of different acoustic scenes.

### A. Datasets and Settings

We evaluate the proposed method on four datasets, one of which is artificially generated as mixtures of isolated sound events, and three are recorded from real-life environments.

While an evaluation performed on real audio data would be ideal, human annotations tend to be somewhat subjective, especially when precise onset and offset are required for overlapping events. For this reason we create our own synthetic dataset—from here onwards referred to as *TUT Sound Events Synthetic 2016* — where we use frame energy based automatic annotation of sound events.

In order to evaluate the proposed method in real-life conditions, we use *TUT Sound Events 2009*. This proprietary dataset contains real-life recordings from 10 different scenes and has been used in many previous works. We also compute and show results on the *TUT Sound Events 2016 development* and *CHiME-Home* dataset, which were used as part of DCASE2016 challenge [1].

*a) TUT Sound Events Synthetic 2016 (TUT-SED Synthetic 2016):* The primary evaluation dataset consists of synthetic mixtures created by mixing isolated sound events from 16 sound event classes. Polyphonic mixture were created by mixing 994 sound event samples. From the 100 mixtures created, 60% are used for training, 20% for testing and 20% for validation. The total length of the data is 566 minutes. Different instances of the sound events are used to synthesize the training, validation and test partitions. Mixtures were created by randomly selecting event instance and from it, randomly, a segment of length 3-15 seconds. Mixtures do not contain any additional background noise. Dataset creation procedure explanation and metadata can be found in the supporting website for the paper[2].

[1] http://www.cs.tut.fi/sgn/arg/dcase2016/

[2] http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/tut-sed-synthetic-2016

*b) TUT Sound Events 2009 (TUT-SED 2009):* This dataset, first presented in [37], consists of 8 to 14 binaural recordings from 10 real-life scenes. Each recording is 10 to 30 minutes long, for a total of 1133 minutes. The 10 scenes are: basketball game, beach, inside a bus, inside a car, hallway, office, restaurant, shop, street and stadium with track and field events. A total of 61 classes were defined, including (wind, yelling, car, shoe squeaks, etc.) and one extra class for unknown or rare events. The average number of events active at the same time is 2.53. Event activity annotations were done manually, which introduces a degree of subjectivity. The database has a five-fold cross-validation setup with training, validation and test set split, each consisting of about 60%, 20% and 20% of the data respectively from each scene. The dataset unfortunately can not be made public due to licensing issues, however three $\sim 10$ minutes samples from the dataset are available at [3].

*c) TUT Sound Events 2016 development (TUT-SED 2016):* This dataset consists of recordings from two real-life scenes: residential area and home [38]. The recordings are captured each in a different location (*i.e.*, different streets, different homes) leading to a large variability on active sound event classes between recordings. For each location, a 3-5 minute long binaural audio recording is provided, adding up to 78 minutes of audio. The recordings have been manually annotated. In total, there are seven annotated sound event classes for residential area recordings and 11 annotated sound event classes for home recordings. The dataset and metadata is available through [4] and [5].

The four-fold cross-validation setup published along with the dataset [38] is used in the evaluations. Twenty percent of the training set recordings are assigned for validation in the training stage of the neural networks. Since in this work we investigate scene-independent SED, we discard the information about the scene, contrary to the DCASE2016 challenge setup. Therefore, instead of training a separate classifier for each scene, we train a single classifier to be used in all scenes. In TUT-SED 2009 all audio material for a scene was recorded in a single location, whereas TUT-SED 2016 contains multiple locations per scene.

*d) CHiME-Home:* CHiME-Home dataset [39] consists of 4-second audio chunks from home environments. The annotations are based on seven sound classes, namely child speech, adult male speech, adult female speech, video game / TV, percussive sounds, broadband noise and other identifiable sounds. In this work, we use the same, *refined* setup of CHiME-Home as it is used in audio tagging task in DCASE2016 challenge [40], namely 1946 chunks for development (in four folds) and 846 chunks for evaluation.

The main difference between this dataset and the previous three is that the annotations are made per chunk instead of per frame. Each chunk is annotated with one or multiple labels. In order to adapt our architecture to the lack of frame-level annotations, we simply add a temporal max-pooling layer—that pools the predictions over time—before the output layer for FNN, CNN, RNN and CRNN. CHiME-Home dataset is available at [6].

[3]http://arg.cs.tut.fi/demo/CASAbrowser/

[4]http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/#tut-sed-2016

[5]https://zenodo.org/record/45759#.WBoUGrPIbRY

### B. Evaluation Metrics

In this work, segment-based evaluation metrics are used. The segment lengths used in this work are (1): a single time frame (40 ms in this work) and (2): a one-second segment. The segment length for each metric is annotated with the subscript (*e.g.*, $F1_{\text{frm}}$ and $F1_{\text{1sec}}$).

Segment-based F1 score calculated in a single time frame ($F1_{\text{frm}}$) is used as the primary evaluation metric [41]. For each segment in the test set, intermediate statistics, *i.e.*, the number of true positive (*TP*), false positive (*FP*) and false negative (*FN*) entries, are calculated as follows. If an event

- is detected in one of the frames inside a segment and it is also present in the same segment of the annotated data, that event is regarded as *TP*.
- is *not* detected in any of the frames inside a segment but it is present in the same segment of the annotated data, that event is regarded as *FN*.
- is detected in one of the frames inside a segment but it is *not* present in the same segment of the annotated data, that event is regarded as *FP*.

These intermediate statistics are accumulated over the test data and then over the folds. This way, each active instance per evaluated segment has equal influence on the evaluation score. This calculation method is referred to as micro-averaging, and is the recommended method for evaluation of classifier [42]. Precision ($P$) and recall ($R$) are calculated from the accumulated intermediate statistics as

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \tag{5}$$

These two metrics are finally combined as their harmonic mean, *F1 score*, which can be formulated as

$$F1 = \frac{2 \cdot P \cdot R}{P + R} \tag{6}$$

More detailed and visualized explanation of segment-based F1 score in multi label setting can be found in [41].

The second evaluation metric is segment-based error rate as proposed in [41]. For error rate, intermediate statistics, *i.e.*, the number of substitutions (**s**), insertions (**i**), deletions (**d**) and active classes from annotations (**a**) are calculated per segment as explained in detail in [41]. Then, the total error rate is calculated as

$$ER = \frac{\sum_{t=1}^{N} \mathbf{s}_t + \sum_{t=1}^{N} \mathbf{i}_t + \sum_{t=1}^{N} \mathbf{d}_t}{\sum_{t=1}^{R} \mathbf{a}_t} \tag{7}$$

where subscript $t$ represents segment index and N is the total number of segments.

Both evaluation metrics are calculated from the accumulated sum for their corresponding intermediate statistics over the segments of the whole test set. If there are multiple scenes in

[6]https://archive.org/details/chime-home

TABLE I: Final hyperparameters used for the evaluation based on the validation results from the hyperparameter grid search.

| | TUT-SED Synthetic 2016 | | | TUT-SED 2009 | | | TUT-SED 2016 | | | CHiME-Home | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNN | RNN | CRNN | CNN | RNN | CRNN | CNN | RNN | CRNN | CNN | RNN | CRNN |
| # CNN layers | 3 | - | 3 | 3 | - | 3 | 3 | - | 3 | 3 | - | 4 |
| pool size | (2,2,2) | - | (5,4,2) | (5,4,2) | - | (5,4,2) | (5,4,2) | - | (2,2,2) | (5,4,2) | - | (2,2,2,1) |
| # RNN layers | - | 3 | 1 | - | 3 | 1 | - | 3 | 3 | - | 2 | 1 |
| # FNN layers | 3 | 2 | - | 1 | 4 | - | 1 | 4 | - | 1 | 1 | - |
| # feature maps/hidden units | 256 | 512 | 256 | 256 | 256 | 256 | 256 | 256 | 96 | 256 | 256 | 256 |
| sequence length (s) | 2.56 | 5.12 | 20.48 | 2.56 | 20.48 | 20.48 | 2.56 | 20.48 | 2.56 | 4 | 4 | 4 |
| # Parameters | 3.7M | 4.5M | 3.6M | 3.4M | 1.3M | 3.7M | 3.4M | 1.3M | 743K | 3.6M | 690K | 6.1M |

the dataset, evaluation metrics are calculated for each scene separately and then the results are presented as the average across the scenes.

The main metric used in previous works [11], [14], [15] on TUT-SED 2009 dataset differs from the F1 score calculation used in this paper. In previous works, F1 score was computed in each segment, then averaged along segments for each scene, and finally averaged across scene scores, instead of accumulating intermediate statistics. This leads to measurement bias under high class imbalance between the classes and also between folds. However, in order to give a comprehensive comparison of our proposed method with previous works on this dataset, we also report the results with this legacy F1 score in Section IV-B.

For CHiME-Home dataset, equal error rate (EER) has been used as the evaluation metric in order to compare the results with DCASE2016 challenge submissions, where EER has been the main evaluation metric.

### C. Baselines

For this work, we compare the proposed method with two recent approaches: the Gaussian mixture model (GMM) of [38] and the feedforward neural network model (FNN) from [15]. GMM has been chosen as a baseline method since it is an established generative modeling method used in many sound recognition tasks [12], [13], [43]. In parallel with the recent surge of deep learning techniques in pattern recognition, FNNs have been shown to vastly outperform GMM based methods in SED [15]. Moreover, this FNN architecture represents a straightforward deep learning method that can be used as a baseline for more complex architectures such as CNN, RNN and the proposed CRNN.

*GMM:* The first baseline system is based on a binary frame-classification approach, where for each sound event class a binary classifier is set up [38]. Each binary classifier consists of a positive class model and a negative class model. The positive class model is trained using the audio segments annotated as belonging to the modeled event class, and a negative class model is trained using the rest of the audio. The system uses MFCCs as features and a GMM-based classifier. MFCCs are calculated using 40 ms frames with Hamming window and 50% overlap and 40 mel bands. The first 20 static coefficients are kept, and delta and acceleration coefficients are calculated using a window length of 9 frames. The 0th order static coefficient is excluded, resulting in a frame-based feature vector of dimension 59. For each sound event, a positive model and a negative model are trained. The models are trained using expectation-maximization algorithm, using k-means algorithm to initialize the training process and diagonal covariance matrices. The number of parameters for GMM baseline is $3808 * K$, where $K$ is the number of classes. In the detection stage, the decision is based on the likelihood ratio between the positive and negative models for each individual sound class event, with a sliding window of one second. The system is used as a baseline in the DCASE2016 challenge [44], however, in this study the system is used as scene-independent to match the setting of the other methods presented.

*FNN:* The second baseline system is a deep multi-label FNN with temporal context [15]. As the sound features, 40 log mel band energy features are extracted for each 40 ms time frame with 50% overlap. For the input, consecutive feature vectors are stacked in five vector blocks, resulting in a 100 ms context window. As the hidden layers, two feedforward layers of 1600 hidden units with maxout activation [45] with pool size of 2 units are used. For the output layer, a feedforward layer of $K$ units with sigmoid activation is used to obtain event activity probabilities per context window, where $K$ is the number of classes. The sliding window post-processing of the event activity probabilities in [15] has not been implemented for the baseline experiments in order to make a fair comparison based on classifier architecture for different deep learning methods. The number of parameters in the baseline FNN model is around 1.6 million.

### D. Experiments set-up

*Preprocessing:* For all neural networks (FNN, CNN, RNN and CRNN) we use log mel band energies as acoustic features. We first compute short-time Fourier transform (STFT) of the recordings in 40 ms frames with 50% overlap, then compute mel band energies through mel filterbank with 40 bands spanning 0 to 22050 Hz, which is the Nyquist rate. After computing the logarithm of the mel band energies, each energy band is normalized by subtracting its mean and dividing by its standard deviation computed over the training set. The normalized log mel band energies are finally split into sequences. During training we use overlapped sequences, *i.e.* we sample the sub-sequences with a different starting point at every epoch, by moving the starting index by a fixed amount that is not a factor of the sequence length (73 in our experiments). The stride is not equal to 1 in order to have

TABLE II: F1 score and error rate results for single frame segments ($F1_{\text{frm}}$ and $ER_{\text{frm}}$) and one second segments ($F1_{\text{1sec}}$ and $ER_{\text{1sec}}$). Bold face indicates the best performing method for the given metric.

| | TUT-SED Synthetic 2016 | | | | TUT-SED 2009 | | | | TUT-SED 2016 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $F1_{\text{frm}}$ | $ER_{\text{frm}}$ | $F1_{\text{1sec}}$ | $ER_{\text{1sec}}$ | $F1_{\text{frm}}$ | $ER_{\text{frm}}$ | $F1_{\text{1sec}}$ | $ER_{\text{1sec}}$ | $F1_{\text{frm}}$ | $ER_{\text{frm}}$ | $F1_{\text{1sec}}$ | $ER_{\text{1sec}}$ |
| GMM [38] | 40.5 | 0.78 | 45.3 | 0.72 | 33.0 | 1.34 | 34.1 | 1.60 | 14.1 | 1.12 | 17.9 | 1.13 |
| FNN [15] | 49.2±0.8 | 0.68±0.02 | 50.2±1.4 | 1.1±0.1 | 60.9±0.4 | 0.56±0.01 | 57.1±0.2 | 1.1±0.01 | 26.7±1.4 | **0.99±0.03** | **32.5±1.2** | 1.32±0.06 |
| CNN | 59.8±0.9 | 0.56±0.01 | 59.9±1.2 | 0.78±0.08 | 64.8±0.2 | 0.50±0.0 | 63.2±0.5 | 0.75±0.02 | 23.0±2.6 | 1.02±0.06 | 26.4±1.9 | 1.09±0.06 |
| RNN | 52.8±1.5 | 0.6±0.02 | 57.1±0.9 | 0.64±0.01 | 62.4±1.0 | 0.52±0.01 | 61.8±0.8 | 0.55±0.01 | **27.6±1.8** | 1.04±0.02 | 29.7±1.4 | 1.10±0.04 |
| **CRNN** | **66.4±0.6** | **0.48±0.01** | **68.7±0.7** | **0.47±0.01** | **69.7±0.4** | **0.45±0.0** | **69.3±0.2** | **0.48±0.0** | **27.5±2.6** | **0.98±0.04** | 30.3±1.7 | **0.95±0.02** |

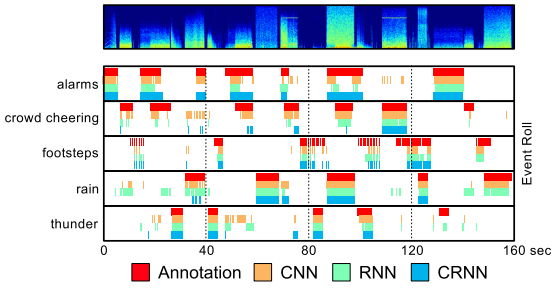


Fig. 3: Annotations and event activity predictions for CNN, RNN and CRNN over a mixture from TUT-SED Synthetic 2016. For clarity, the classes that are not present in the mixture are omitted.

effectively different sub-sequences from one training epoch to the next one. For validation and test data we do not use any overlap.

While finer frequency resolution or different representations could improve the accuracy, our main goal is to compare the architectures. We opted for this setting as it was recently used with very good performance in several works on SED [11], [15].

*Neural network configurations:* Since the size of the dataset usually affects the optimal network architecture, we do a hyperparameter search by running a series of experiments over predetermined ranges. We select for each network architecture the hyperparameter configuration that leads to the best results on the validation set, and use this architecture to compute the results on the test set.

For TUT-SED Synthetic 2016 and CHiME-Home datasets, we run a hyperparameter grid search on the number of CNN feature maps and RNN hidden units {96, 256} (set to the same value); the number of recurrent layers {1, 2, 3}; and the number of CNN layers {1, 2, 3 ,4} with the following frequency max pooling arrangements after each convolutional layer {(4), (2, 2), (4, 2), (8, 5), (2, 2, 2), (5, 4, 2), (2, 2, 2, 1), (5, 2, 2, 2)}. Here, the numbers denote the number of frequency bands at each max pooling step; *e.g.,* the configuration (5, 4, 2) pools the original 40 bands to one band in three stages: 40 bands $\rightarrow$ 8 bands $\rightarrow$ 2 bands $\rightarrow$ 1 band.

All networks have batch normalization layers after convolutional layers and dropout rate 0.25, which were found to be helpful in preliminary experiments. The output layer consists of a node for each class and has the sigmoid as activation function. In convolutional layers we use filters with shape $(5, 5)$; in recurrent layers we opted for GRU, since preliminary experiments using LSTM yielded similar results and GRU units have a smaller number of parameters. The weights are initialized according to the scheme proposed in [46]. Binary cross-entropy is set as the loss function, and all networks are trained with Adam [47] as gradient descent optimizer, with the default parameters proposed in the original paper.

To evaluate the effect of having both convolutional and recurrent layers in the same architecture, we compare the CRNN with CNNs and RNNs alone. For both CNN and RNN we run the same hyperparameter optimization procedure described for CRNN, replacing recurrent layers with feedforward layers for CNNs, and removing convolutional layers for RNNs while adding feedforward layers before the output layer. This allows for a fair comparison, providing the possibility of having equally deep networks for all three architectures.

After this first optimization process, we use the best CRNNs, CNNs and RNNs to separately test the effect of varying other hyperparameters. More specifically we investigate how performance is affected by variation of the CNN filter shapes and the sequence length. For the CRNN we test filter shapes in the set {(3,3), (5,5), (11,11), (1,5), (5,1), (3,11), (11,3)}, where $(*, *)$ represents the filter lengths in frequency and time axes, respectively. For CRNN and RNN, we test shorter and longer sequences than the initial value of 128 frames, experimenting in the range {8, 32, 128, 256, 512, 1024, 2048} frames, which correspond to {0.16, 0.64, 2.56, 5.12, 10.24, 20.48, 40.96} seconds respectively. We finally use the hyperparameters that provide the highest validation scores as our final CRNN, CNN and RNN models.

For the other two datasets (TUT-SED 2009 and TUT-SED 2016) we select a group of best performing model configurations on validation data from TUT-SED Synthetic 2016 experiments and to account for the different amount of data we run another smaller hyperparameter search, varying the amount of dropout and the sequence length. Again, we then select the best performing networks on the validation score to compute the test results. The hyperparameters used in the evaluation for all three datasets is presented in Table I.

The event activity probabilities are thresholded at $C = 0.5$, in order to obtain the binary activity matrix used to compute the reference metrics based on the ground truth. All networks are trained until overfitting starts to arise: as a criterion we use early stopping on the validation metric, halting the training if the score is not improving for more than 100 epochs and reverting the weights to the values that best performed on

TABLE III: $F1_{\text{frm}}$ for CNN, RNN and CRNN for each class in TUT-SED Synthetic 2016.

| Class | avg. (secs) | total (secs) | CNN | RNN | CRNN |
|---|---|---|---|---|---|
| glass smash | 1.2 | 621 | **57±8.6** | 48±2.0 | **54±6.7** |
| gun shot | 1.7 | 534 | 53±5.9 | 64±2.3 | **73±1.8** |
| cat meowing | 2.1 | 941 | **37±4.6** | 29±4.5 | **42±3.9** |
| dog barking | 5.0 | 716 | **69±3.3** | 51±2.5 | **73±3.1** |
| thunder | 5.9 | 3007 | 55±3.3 | 46±2.2 | **63±1.9** |
| bird singing | 6.1 | 2298 | 44±1.2 | 41±3.1 | **53±2.3** |
| horse walk | 6.4 | 1614 | **46±2.1** | 39±2.7 | **45±2.4** |
| baby crying | 6.9 | 2007 | 46±5.7 | 46±1.1 | **59±3.0** |
| motorcycle | 7.0 | 3691 | **47±3.1** | **44±2.2** | **47±2.7** |
| footsteps | 7.1 | 1173 | 41±2.0 | 34±1.2 | **47±1.7** |
| crowd applause | 7.3 | 3278 | **68±1.8** | 57±1.5 | **71±0.6** |
| bus | 7.8 | 3464 | 60±2.0 | 55±2.5 | **66±2.4** |
| mixer | 7.9 | 4020 | 62±5.6 | 57±6.4 | **82±2.7** |
| crowd cheering | 8.1 | 4825 | 72±2.9 | 64±2.7 | **77±1.1** |
| alarms | 8.2 | 4405 | **64±2.2** | 50±5.3 | **66±2.9** |
| rain | 8.2 | 3975 | **71±2.0** | 59±2.6 | **72±1.9** |

validation.

For feature extraction, the Python library Librosa [48] has been used in this work. For classifier implementations, deep learning package Keras (version 1.1.0) [49] is used with Theano (version 0.8.2) as backend [50]. The networks are trained on NVIDIA Tesla K40t and K80 GPUs.

## IV. Results

In this section, we present results for all the datasets and experiments described in Section III. The evaluation of CNN, RNN and CRNN methods are conducted using the hyperparameters given in Table I. All the reported results are computed on the test sets. Unless otherwise stated, we run each neural network based experiment ten times with different random seeds (five times for TUT-SED 2009) to reflect the effect of random weight initialization. We provide the mean and the standard deviation of these experiments in this section. Best performing method is highlighted with bold face in the tables of this section. The methods whose best performance among the ten runs is within one standard deviation of the best performing method is also highlighted with bold face.

The main results with the best performing (based on the validation data) CRNN, CNN, RNN, and the GMM and FNN baselines are reported in Table II. Results are calculated according to the description in Section III-B where each event instance irrespective of the class is taken into account in equal manner. As shown in the table, the CRNNs consistently outperforms CNNs, RNNs and the two baseline methods on all three datasets for the main metric.

### A. TUT Sound Events Synthetic 2016

As presented in Table II, CRNN improved by absolute 6.6% and 13.6% on frame-based F1 compared to CNN and RNN respectively for TUT-SED synthetic 2016 dataset. Considering the number of parameters used for each method (see Table I), the performance of CRNN indicates an architectural advantage compared to CNN and RNN methods. All the four deep learning based methods outperform the baseline GMM method.

TABLE IV: $F1_{\text{frm}}$ for accuracy vs. convolution filter shape for TUT-SED Synthetic 2016 dataset. $(*, *)$ represents filter lengths in frequency and time axis, respectively.

| Filter shape | (3,3) | (5,5) | (11,11) | (1,5) | (5,1) | (3,11) | (11,3) |
|---|---|---|---|---|---|---|---|
| $F1_{\text{frm}}$ | 67.2 | **68.3** | 62.6 | 28.5 | 60.6 | 67.4 | 61.2 |

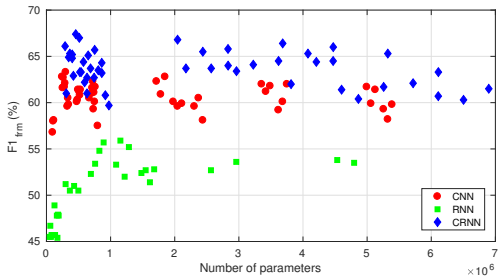


Fig. 4: Number of parameters vs. accuracy for CNN, RNN and CRNN.

As claimed in [51], this may be due to the capability of deep learning methods to use different subsets of hidden units to model different sound events simultaneously. An example mixture from TUT-SED Synthetic 2016 test set is presented in Figure 3 with annotations and event activity predictions from CNN, RNN and CRNN.

*1) Class-wise performance:* The class-wise performance with $F1_{\text{frm}}$ metric for CNN, RNN and CRNN methods along with the average and total duration of the classes are presented in Table III. CRNN outperforms both CNN and RNN on almost all classes. It should be kept in mind that each class is likely to appear together with different classes rather than isolated. Therefore the results in Table III present the performance of the methods for each class in a polyphonic setting, as would be the case in a real-life environment. The worst performing class for all three networks is cat meowing, which consists of short, harmonic sounds. We observed that cat meowing samples are mostly confused by baby crying, which has similar acoustic characteristics. Besides, short, non-impulsive sound events are more likely to be masked by another overlapping sound event, which makes their detection more challenging. CRNN performance is considerably better compared to CNN and RNN for gun shot, thunder, bird singing, baby crying and mixer sound events. However, it is hard to make any generalizations on the acoustic characteristics of these events that can explain the superior performance.

*2) Effects of filter shape:* The effect of the convolutional filter shape is presented in Table IV. Since these experiments were part of the hyperparameter grid search, each experiment is conducted only once. Small kernels, such as (5,5) and (3,3), were found to perform the best in the experiments run on this dataset. This is consistent with the results presented in [31] on a similar task. The very low performance given for the filter shape (1,5) highlights the importance of including multiple frequency bands in the convolution when spectrogram based features are used as input for the CRNN.

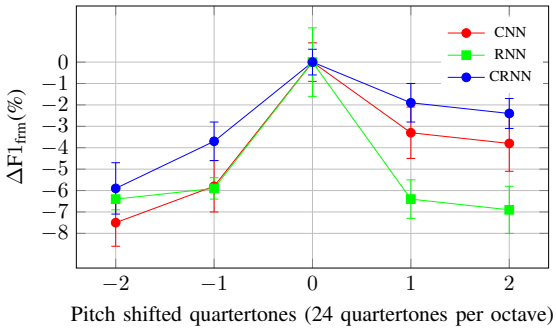


Fig. 5: Absolute accuracy change vs. pitch-shifting over $\pm 2$ quartertones for CNN, RNN and CRNN.

*3) Number of parameters vs. accuracy:* The effect of number of parameters on the accuracy is investigated in Figure 4. The points in the figure represent the test accuracy with $F1_{\text{frm}}$ metric for the hyperparameter grid search experiments. Each experiment is conducted one time only. Two observations can be made from the figure. For the same number of parameters, CRNN has a clear performance advantage over CNN and RNN. This indicates that the high performance of CRNN can be explained with the architectural advantage rather than the model size. In addition, there can be a significant performance shift for the same type of networks with the same number of parameters, which means that a careful grid search on hyperparameters (e.g. shallow with more hidden units per layer vs. deep with less hidden units per layer) is crucial in finding the optimal network structure.

*4) Frequency shift invariance:* Sound events may exhibit small variations in their frequency content. In order to investigate the robustness of the networks to small frequency variations, pitch shift experiments are conducted and the absolute changes in frame-based F1 score are presented in Figure 5. For these experiments, each network is first trained with the original training data. Then, using Librosa's pitch-shift function, the pitch for the mixtures in the test set is shifted by $\pm 2$ quartertones. The test results show a significant absolute drop in accuracy for RNNs when the frequency content is shifted slightly. As expected, CNN and CRNN are more robust to small changes in frequency content due to the convolution and max-pooling operations. However, accuracy decrease difference between the methods diminishes for negative pitch shift, for which the reasons should be further investigated. It should be also noted that RNN has the lowest base accuracy, so it is relatively more affected for the same amount of absolute accuracy decrease (see Table II).

*5) Closer look on network outputs:* A comparative study on the neural network outputs, which are regarded as event activity probabilities, for a 13-second sequence of the test set is presented in Figure 6. For the parts of the sequence where *dog barking* and *baby crying* appear alone, all three networks successfully detect these events. However, when a *gun shot* appears overlapping with *baby crying*, only CRNN can detect the *gun shot* although there is a significant change in the input

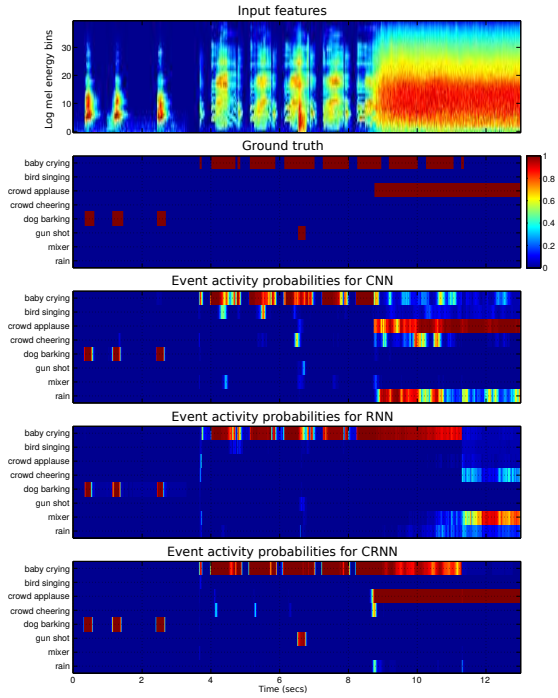


Fig. 6: Input features, ground truth and event activity probabilities for CNN, RNN and CRNN from a sequence of test examples from TUT-SED synthetic 2016.

feature content. This indicates the efficient modeling of the *gun shot* by CRNN which improves the detection accuracy even in polyphonic conditions. Moreover, when *crowd applause* begins to appear in the signal, it almost completely masks *baby crying*, as it is evident from the input features. CNN correctly detects *crowd applause*, but misses the masked *baby crying* in this case, and RNN ignores the significant change in features and keeps detecting *baby crying*. RNN's insensitivity to the input feature change can be explained with its input gate not passing through new inputs to recurrent layers. On the other hand, CRNN correctly detects both events and almost perfectly matches the ground truth along the whole sequence.

### B. TUT-SED 2009

For a comprehensive comparison, results with different methods applied to the same cross-validation setup and published over the years are shown in Table V. The main metric used in these previous works is averaged over folds, and may be influenced by distribution of events in the folds (see Section III-B). In order to allow a direct comparison, we have computed all metrics in the table the same way.

First published systems were scene-dependent, where information about the scene is provided to the system and separate event models are trained for each scene [14], [24], [25]. More recent work [11], [15], as well as the current study, consist of scene-independent systems. Methods [24], [25] are

TABLE V: Results for TUT-SED 2009 based on the legacy F1. Methods marked with ⋆ are trained in scene-dependent setting.

| Method | Legacy $F1_{1\text{sec}}$ |
|---|---|
| HMM multiple Viterbi decoding⋆ [24] | 20.4 |
| NMF-HMM⋆ [25] | 36.7 |
| NMF-HMM + stream elimination⋆ [25] | 44.9 |
| GMM⋆ [38] | 34.6 |
| Coupled NMF⋆ [14] | 57.8 |
| FNN [15] | 63.0 |
| BLSTM [11] | 64.6 |
| CNN | 63.9±0.4 |
| RNN | 62.2±0.8 |
| **CRNN** | **69.1±0.4** |

HMM based, using either multiple Viterbi decoding stages or NMF pre-processing to do polyphonic SED. In contrast, the use of NMF in [14] does not build explicit class models, but performs coupled NMF of spectral representation and event activity annotations to build dictionaries. This method performs polyphonic SED through direct estimation of event activities using learned dictionaries.

The results on the dataset show significant improvement with the introduction of deep learning methods. CRNN has significantly higher performance than previous methods [14], [24], [25], [38], and it still shows considerable improvement over other neural network approaches.

### C. TUT-SED 2016

The CRNN and RNN architectures obtain the best results in terms of framewise *F1*. The CRNN outperforms all the other architectures for *ER* framewise and on 1-second blocks. While the FNN obtains better results on the 1-second block *F1*, this happens at the expense of a very large 1-second block *ER*.

For all the analyzed architectures, the overall results on this dataset are quite low compared to the other datasets. This is most likely due the fact that TUT-SED 2016 is very small and the sounds events occur sparsely (i.e. a large portion of the data is silent). In fact, when we look at class-wise results (unfortunately not available due to space restrictions), we noticed a significant performance difference between the classes that are represented the most in the dataset (e.g. bird singing and car passing by, $F1_{\text{frm}}$ around 50%) and the least represented classes (e.g. cupboard and object snapping, $F1_{\text{frm}}$ close to 0%). Some other techniques might be applied to improve the accuracy of systems trained on such small datasets, e.g. training a network on a larger dataset and then retraining the output layer on the smaller dataset (transfer learning), or incorporating unlabeled data to the learning process (semi-supervised learning).

### D. CHiME-Home

The results obtained on CHiME-Home are reported in Table VI. For all of our three architectures there is a significant improvement over the previous results reported on the same dataset on the DCASE2016 challenge, setting new state-of-the-art results.

TABLE VI: Equal error rate (EER) results for CHiME-Home development and evaluation datasets.

| Method | Development EER | Evaluation EER |
|---|---|---|
| Lidy *et al.* [52] | 17.8 | 16.6 |
| Cakir *et al.* [53] | 17.1 | 16.8 |
| Yun *et al.* [54] | 17.6 | 17.4 |
| CNN | **12.6±0.5** | **10.7±0.6** |
| RNN | 16.0±0.3 | 13.8±0.4 |
| CRNN | **13.0±0.3** | **11.3±0.6** |
| CNN (no batch norm) | 15.1±1.7 | 11.9±1.0 |

After the first series of experiments the CNN obtained slightly better results compared to the CRNN. The CRNN and CNN architecture used are almost identical, with the only exception of the last recurrent (GRU) layer in the CRNN being replaced by a fully connected layer followed by batch normalization. In order to test if the improvement in the results was due to the absence of recurrent connections or to the presence of batch normalization, we run again the same CNN experiments removing the normalization layer. As shown in the last row of VI, over 10 different random initializations the average EER increased to values above those obtained by the CRNN.

### E. Visualization of convolutional layers

Here we take a peek at the representation learned by the networks. More specifically, we use the technique described in [55] to visualize what kind of patterns in the input data different neurons in the convolutional layers are looking for. We feed the network a random input whose entries are independently drawn from a Gaussian distribution with zero mean and unit variance. We choose one neuron in a convolutional layer, compute the gradient of its activation with respect to the input, and iteratively update the input through gradient ascent in order to increase the activation of the neuron. If the gradient ascent optimization does not get stuck into a weak local maximum, after several updates the resulting input will strongly activate the neuron. We run the experiment for several convolutional neurons in the CRNN networks trained on TUT-SED Synthetic 2016 and TUT-SED 2009, halting the optimization after 100 updates. In Figure 7 we present a few of these inputs for several neurons at different depth. The figure confirms that the convolutional filters have specialized into finding specific patterns in the input. In addition, the complexity of the patterns looked for by the filters seems to increase as the layers become deeper.

## V. Conclusions

In this work, we proposed to apply a CRNN—a combination of CNN and RNN, two complementary classification methods—on a polyphonic SED task. The proposed method

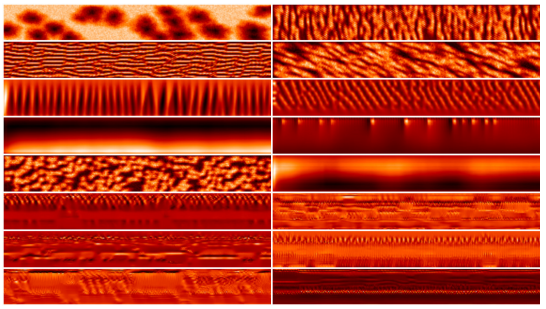

Fig. 7: Two columns of crops from input patterns that would strongly activate certain neurons from different layers of the CRNN. On the horizontal axis is time, on the vertical axis mel bands. On both columns the rows 1 and 2 are from neurons in the first convolutional layer, rows 3 to 5 from the second, and rows from 6 to 8 from the third.

first extracts higher level features through multiple convolutional layers (with small filters spanning both time and frequency) and pooling in frequency domain; these features are then fed to recurrent layers, whose features in turn are used to obtain event activity probabilities through a feedforward fully connected layer. In CRNN, CNN's capability to learn local translation invariant filters and RNN's capability to model short and long term temporal dependencies are gathered in a single classifier. The evaluation results over four datasets show a clear performance improvement for the proposed CRNN method compared to CNN, RNN, and other established methods in polyphonic SED.

Despite the improvement in performance, we identify a limitation to this method. As presented in TUT-SED 2016 results in Table II, the performance of the proposed CRNN (and of the other deep learning based methods) strongly depends on the amount of available annotated data. TUT-SED 2016 dataset consists of 78 minutes of audio of which only about 49 minutes are annotated with at least one of the classes. When the performance of CRNN for TUT-SED 2016 is compared to the performance on TUT-SED 2009 (1133 minutes) and TUT-SED Synthetic 2016 (566 minutes), there is a clear performance drop both in the absolute performance and in the relative improvement with respect to other methods. Dependency on large amounts of data is a common limitation of current deep learning methods.

The results we observed in this work, and in many other classification tasks in various domains, prove that deep learning is definitely worth further investigation on polyphonic SED. As a future work, semi-supervised training methods can be investigated to overcome the limitation imposed by small datasets. Transfer learning [56], [57] could be potentially applied with success in this setting: by first training a CRNN on a large dataset (such as TUT-SED Synthetic 2016), the last feedforward layer can then be replaced with random weights and the network fine-tuned on the smaller dataset.

Another issue worth investigating would be a detailed study over the activations from different stages of the proposed CRNN method. For instance, a class-wise study over the higher level features extracted from the convolutional layers might give an insight on the common features of different sound events. Finally, recurrent layer activations may be informative on the degree of relevance of the temporal context information for various sound events.

## REFERENCES


[1] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Reliable detection of audio events in highly noisy environments," *Pattern Recognition Letters*, vol. 65, pp. 22–28, 2015.

[2] S. Goetze, J. Schroder, S. Gerlach, D. Hollosi, J.-E. Appell, and F. Wallhoff, "Acoustic monitoring and localization for social care," *Journal of Computing Science and Engineering*, vol. 6, no. 1, pp. 40–50, 2012.

[3] J. Salamon and J. P. Bello, "Feature learning with deep scattering for urban sound analysis," in *2015 23rd European Signal Processing Conference (EUSIPCO)*. IEEE, 2015, pp. 724–728.

[4] Y. Wang, L. Neves, and F. Metze, "Audio-based multimedia event detection using deep recurrent neural networks," in *2016 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2742–2746.

[5] D. Stowell and D. Clayton, "Acoustic event detection for multiple overlapping similar sources," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2015, pp. 1–5.

[6] J. W. Dennis, "Sound event recognition in unstructured environments using spectrogram image processing," *Nanyang Technological University, Singapore*, 2014.

[7] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic events using deep neural networks," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2014.

[8] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *2015 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 559–563.

[9] H. Phan, L. Hertel, M. Maass, and A. Mertins, "Robust audio event recognition with 1-max pooling convolutional neural networks," *Interspeech*, 2016.

[10] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Int. Workshop on Machine Learning for Signal Processing (MLSP)*, 2015, pp. 1–6.

[11] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *2016 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 6440–6444.

[12] L.-H. Cai, L. Lu, A. Hanjalic, H.-J. Zhang, and L.-H. Cai, "A flexible framework for key audio effects detection and auditory context inference," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 1026–1039, 2006.

[13] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2010, pp. 1267–1271.

[14] A. Mesaros, O. Dikmen, T. Heittola, and T. Virtanen, "Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations," in *2015 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 151–155.

[15] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multilabel deep neural networks," in *Int. Joint Conf. on Neural Networks (IJCNN)*, 2015, pp. 1–7.

[16] E. Cakir, E. Ozan, and T. Virtanen, "Filterbank learning for deep neural network based polyphonic sound event detection," in *Int. Joint Conf. on Neural Networks (IJCNN)*, 2016.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[20] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.

[21] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[22] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[23] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3128–3137.

[24] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, p. 1, 2013.

[25] T. Heittola, A. Mesaros, T. Virtanen, and M. Gabbouj, "Supervised model training for overlapping sound events based on unsupervised source separation," in *Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 8677–8681.

[26] O. Dikmen and A. Mesaros, "Sound event detection using non-negative dictionaries learned from annotated overlapping events," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2013, pp. 1–4.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," *arXiv preprint arXiv:1512.02595*, 2015.

[29] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Proc. Interspeech*, 2015.

[30] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," *arXiv preprint arXiv:1609.04243*, 2016.

[31] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[34] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[35] Y. Gal, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in neural information processing systems*, 2016.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 448–456.

[37] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms," in *Proc. of the 18th European Signal Processing Conference (EUSIPCO)*, 2010, pp. 1272–1276.

[38] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *24th European Signal Processing Conference (EUSIPCO)*, 2016.

[39] P. Foster, S. Sigtia, S. Krstulovic, J. Barker, and M. D. Plumbley, "Chime-home: A dataset for sound source recognition in a domestic environment," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2015, pp. 1–5.

[40] T. Heittola. DCASE2016 challenge - audio tagging. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2016/task-audio-tagging

[41] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[42] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, pp. 49–57, 2010.

[43] W.-H. Cheng, W.-T. Chu, and J.-L. Wu, "Semantic context detection based on hierarchical audio models," in *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, 2003, pp. 109–115.

[44] T. Heittola, A. Mesaros, and T. Virtanen. (2016) DCASE2016 baseline system. [Online]. Available: https://github.com/TUT-ARG/DCASE2016-baseline-system-python

[45] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks." *ICML (3)*, vol. 28, pp. 1319–1327, 2013.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE Int. Conf. on Computer Vision*, 2015, pp. 1026–1034.

[47] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International conference on learning representations*, 2015.

[48] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, 2015.

[49] F. Chollet. (2016) Keras. [Online]. Available: https://github.com/fchollet/keras

[50] T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[51] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[52] T. Lidy and A. Schindler, "CQT-based convolutional neural networks for audio scene classification and domestic audio tagging," DCASE2016 Challenge, Tech. Rep., September 2016.

[53] E. Cakir, T. Heittola, and T. Virtanen, "Domestic audio tagging with convolutional neural networks," DCASE2016 Challenge, Tech. Rep., September 2016.

[54] S. Yun, S. Kim, S. Moon, J. Cho, and T. Kim, "Discriminative training of GMM parameters for audio scene classification," DCASE2016 Challenge, Tech. Rep., September 2016.

[55] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *ICLR Workshop*, 2014.

[56] Y. Bengio *et al.*, "Deep learning of representations for unsupervised and transfer learning." *ICML Unsupervised and Transfer Learning*, vol. 27, pp. 17–36, 2012.

[57] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.



**Emre Cakir** received his BSc. degree in electrical-electronics engineering from Middle East Technical University, Ankara, Turkey in 2013 and the MSc. degree in Information Technology from Tampere University of Technology (TUT), Finland in 2015. He has been with the Audio Research Group in TUT since February, 2014 where he currently continues his PhD studies. His main research interests are sound event detection in real-life environments and deep learning.



**Giambattista Parascandolo** received his B.Sc. degree from the Department of Mathematics at University of Rome Tor Vergata, Italy, in 2013, and his M.Sc. degree in Information Technology from Tampere University of Technology (TUT), Finland, in 2015. He is a Project Researcher at the Audio Research Group in TUT, where he has been since February, 2015. His main research interests are deep learning and machine learning.

**Toni Heittola** received his M.Sc. degree in Information Technology from Tampere University of Technology (TUT), Finland, in 2004. He is currently pursuing the Ph.D. degree at TUT. His main research interests are sound event detection in real-life environments, sound scene classification and audio content analysis.

**Heikki Huttunen** received his Ph.D. degree in Signal Processing at Tampere University of Technology (TUT), Finland, in 1999. Currently he is a university lecturer at the Department of Signal Processing at TUT. He is an author of over 100 research articles on signal and image processing and analysis. His research interests include Optical Character Recognition, Deep learning, and Pattern recognition and Statistics.

**Tuomas Virtanen** is an Academy Research Fellow and an adjunct professor at Department of Signal Processing, Tampere University of Technology (TUT), Finland. He received the M.Sc. and Doctor of Science degrees in information technology from TUT in 2001 and 2006, respectively. He is known for his pioneering work on single-channel sound source separation using non-negative matrix factorization based techniques, and their application to noise-robust speech recognition, music content analysis and audio event detection. In addition to the above topics, his research interests include content analysis of audio signals in general and machine learning.He has received the IEEE Signal Processing Society 2012 best paper award.

# Publication IV

Emre Çakır, Sharath Adavanne, Giambattista Parascandolo, Konstantinos Drossos, Tuomas Virtanen. "Convolutional Recurrent Neural Networks for Bird Audio Detection", in *Proceedings of the 25th European Signal Processing Conference (EUSIPCO).* Kos, Greece, September 2017, pp. 1744-1748.

©2017 IEEE. Reprinted, with permission, from E. Çakır, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, Convolutional Recurrent Neural Networks for Bird Audio Detection, in Proceedings of the 25th European Signal Processing Conference, September 2017.

# CONVOLUTIONAL RECURRENT NEURAL NETWORKS FOR BIRD AUDIO DETECTION

*Emre Çakır, Sharath Adavanne, Giambattista Parascandolo, Konstantinos Drossos, Tuomas Virtanen*

Department of Signal Processing, Tampere University of Technology

**ABSTRACT**

Bird sounds possess distinctive spectral structure which may exhibit small shifts in spectrum depending on the bird species and environmental conditions. In this paper, we propose using convolutional recurrent neural networks on the task of automated bird audio detection in real-life environments. In the proposed method, convolutional layers extract high dimensional, local frequency shift invariant features, while recurrent layers capture longer term dependencies between the features extracted from short time frames. This method achieves 88.5% Area Under ROC Curve (AUC) score on the unseen evaluation data and obtains the second place in the Bird Audio Detection challenge.

***Index Terms*—** Bird audio detection, convolutional recurrent neural network

## 1. INTRODUCTION

Bird audio detection (BAD) is defined as identifying the presence of bird sounds in a given audio recording. In many conventional, remote wildlife-monitoring projects, the monitoring/detection process is not fully automated and requires heavy manual labor to label the obtained data (e.g. by employing video or audio) [1, 2]. In certain cases such as dense forests and low illumination, automated detection of birds in wildlife can be more effective through their sounds compared to visual cues. Besides, acoustic monitoring devices can be easily deployed to cover wide ranges of land. This indicates the need for automated BAD systems in various aspects of biological monitoring. For instance, it can be applied in the automatic monitoring of biodiversity, migration patterns, and bird population densities [2, 3]. BAD systems can be augmented with another classifier to determine the species of the detected birds [4]. Using an automated BAD system as preprocessing/filtering step to determine the bird species would be beneficial especially for remote acoustic monitoring projects, where large amount of audio data is employed.

The research leading to these results has received funding from the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND. The authors wish to acknowledge CSC-IT Center for Science, Finland, for computational resources.

In this regard, the Bird Audio Detection challenge [5] is organized with an objective to stimulate the research on BAD systems which can work on real life bioacoustics monitoring projects. The challenge provides three bird audio datasets recorded in different acoustic environments. Two of the datasets are provided with bird call annotations to be used as development data. The final dataset consists of recordings from a different physical environment and it is employed as the evaluation data. An extensive review on the recent work on BAD can also be found in [5].

Bird sounds can be broadly categorized as vocal and non-vocal sounds (such as bill clattering, and drumming of woodpeckers) [6]. Since non-vocal bird sounds are harder to be associated with birds without any visual cues, the research on BAD has been mostly focused on vocal sounds, as in this work. Vocal sounds can be further categorized as bird calls and bird songs. Bird calls are often short and serve a particular function such as alarming or keeping the flock in contact. Bird songs are typically longer and more complex than bird calls, and they often possess temporal structure which are melodious to human ears [7]. Mating calls can be given as example to bird songs. Vocal bird sounds include distinctive spectral content often including harmonics. Alarm calls tend to be high-pitched with rapid modulations (to get maximum attention), whereas lower frequency calls are common in densely vegetated areas to avoid signal degradation due to reverberation [8]. Furthermore, depending on the environmental conditions (e.g. ambient noise level, vegetation density) and the bird species, bird sounds may exhibit certain local frequency shift variations [8]. Therefore, a BAD system should be able to capture melodic cues in time domain, and also should be robust to local frequency shifts.

Convolutional neural networks (CNN) are able to extract higher level features that are invariant to local spectral and temporal shifts. Recurrent neural networks (RNNs) are powerful in learning the longer term temporal context in the audio signals. In this work, we combine these two approaches in a convolutional recurrent neural network (CRNN) and apply it over spectral acoustic features for the BAD challenge. This method consists of slight modification (temporal max-pooling to obtain file-level estimation instead of frame-level estimation) and hyperparameter fine-tuning for the challenge over the CRNN proposed in [9], where it has provided state-

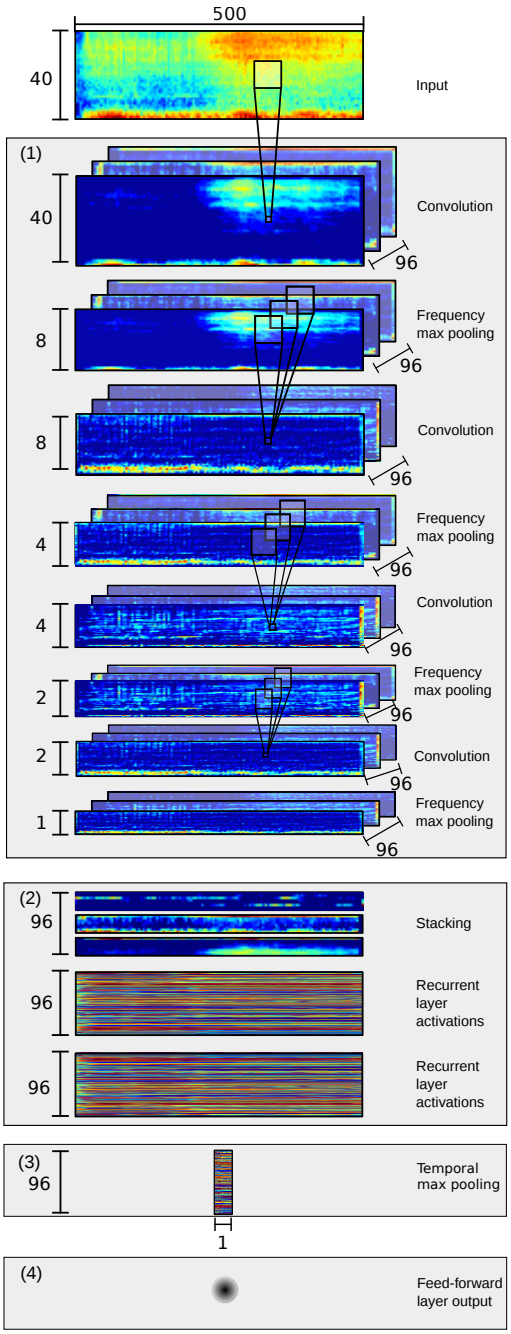


**Fig. 1**. Illustration of the CRNN architecture proposed for bird audio detection.

of-the-art results on various polyphonic sound event detection and audio tagging tasks. Similar approaches combining CNNs and RNNs have been presented recently in ASR [10] and music classification [11].

The rest of the paper is organized as follows. The employed acoustic features and the proposed CRNN for the BAD are presented in Section 2. Dataset settings, metrics, and method configuration are reported in Section 3. In Section 4 are the results and their discussion, followed by the conclusions in Section 5.

## 2. METHOD

The proposed method consists of two stages. In the first stage, spectro-temporal features (spectrogram) are extracted from the raw audio recordings to be used as the sound representation. In the second stage, a CRNN is used to map the acoustic features to a binary estimate of bird song presence. CRNN parameters are obtained by supervised learning using material that consists of acoustic features extracted from a training database and the annotations of bird song activity.

### 2.1. Features

The utilized spectro-temporal features are log mel-band energies, extracted from short frames. These features has been shown to perform well in various audio tagging and sound event detection tasks [12, 13, 9]. First, we obtained the magnitude spectrum of the audio signals by using short-time Fourier transform (STFT) over 40 ms audio frames of 50% overlap, windowed with Hamming window. Duration of each audio file in the challenge dataset is 10 seconds, resulting to 500 frames for each file. Then, 40 log mel-band energy features were extracted from the magnitude spectrum. Librosa library [14] was used in the feature extraction process.

Keeping in mind that bird sounds are often contained in a relatively small portion of the frequency range (mostly around 2-8 kHz), extracting features from that range seems like a good approach. However, experiments with features from the whole frequency range (from 0 Hz to Nyquist frequency) provided better results, and were therefore utilized in the proposed method.

### 2.2. Convolutional recurrent neural networks

The CRNN proposed in this work, depicted in Figure 1, consists of four parts:

1. convolutional layers with rectified linear unit (ReLU) activations and non-overlapping pooling over frequency axis
2. gated recurrent unit (GRU) [15] layers
3. a temporal max-pooling layer, and

4. a single feedforward layer with a single unit and sigmoid activation, as the classification layer.

A time-frequency representation of the data is fed to the convolutional layers and the activations from the filters of the last convolutional layer are stacked over the frequency axis and fed to the first GRU layer. The extracted representations over each time frame (from the last GRU layer) are used as input to the temporal max-pooling layer. Output of the max-pooling layer is employed as input to the classification layer. Output of the classification layer is treated as the bird audio probability for the audio file. The aim of the network learning is to get the estimated bird audio probabilities as close as to their binary target outputs, where target output is 1 if any bird sound is present in a given recording, and 0 vice versa.

The network is trained with back-propagation through time using Adam optimizer [16] and binary cross-entropy as the loss function. In order to reduce overfitting of the model, early stopping was used to stop training if the validation data AUC score did not improve for 50 epochs. For regularization, batch normalization [17] was employed in convolutional layers and dropout [18] with rate 0.25 was employed in convolutional and recurrent layers. Keras deep learning library [19] has been used to implement the network.

The proposed method differs from our other submission [20] for the challenge (which came in fifth place) in the following ways: we use a single set of acoustic features, smaller max pool size in frequency domain and no max pooling in time domain, no maxout for the output layer, and the whole method consists of a single branch with unidirectional GRU. In addition, considering the auxiliary data augmentation and domain adaptation techniques applied in [20], the proposed method is less complex and still performs better in the given BAD challenge.

## 3. EVALUATION

### 3.1. Datasets

The Bird Audio Detection challenge [5] consists of a development and an evaluation set. The development set consists of *freefield1010* (field recordings gathered by the [1]FreeSound project) and *warblr* (crowd-sourced recordings collected through smartphone app) datasets, and the evaluation set consists of *chernobyl* (collected by unattended recorders in Chernobyl exclusion zone) dataset. Recordings in all the datasets are around 10 seconds long, single channel, and sampled at 44.1 kHz. The annotations for the recordings are binary - bird calls present or absent. The total duration of the available recordings is approximately 68 hours, which makes the dataset a valuable source for detection methods that require large amount of material. The statistics of the datasets are presented in Table 1.

[1]http://freesound.org/

**Table 1**. Bird audio detection challenge [5] dataset statistics

| Dataset | Bird call | | |
|---|---|---|---|
| | Present | Absent | Total |
| freefield1010 | 5755 | 1935 | 7690 |
| warblr | 1955 | 6045 | 8000 |
| chernobyl | ? | ? | 8620 |
| Total | 7710 + ? | 7980 + ? | 24310 |

**Table 2**. Final hyperparameters used for the evaluation based on the validation results from the hyperparameter grid search.

| | Hyperparameters |
|---|---|
| # convolutional layers | 4 |
| Filter shape | 5-by-5 |
| pool size | (5,2,2,2) |
| # recurrent layers | 2 |
| # feature maps/hidden units | 96 |
| # Parameters | 806K |

From the development set, we create five different splits with 60% training, 20% validation, and 20% testing set distribution. Each split has an equal distribution of birds call present and absent, i.e. 60% of all the development data with present bird call annotation is included in training data, and the same is valid for absent bird call annotations. Different splits are obtained by randomly shuffling the recordings list and re-partitioning the data in given proportions. All development set results are the average performance over the splits. For the challenge submission, the CRNN is trained on single split of 80% training and 20% validation done on development set, with equal distribution of classes.

### 3.2. Evaluation Metric and Configuration

The BAD system output is evaluated from the receiver operating characteristic (ROC) using the AUC measurement.

In order to obtain the optimal hyperparameters for the given task, we run a hyperparameter grid search over the validation set. The grid search covers each of the combinations of the following hyperparameter values: the number of CNN feature maps/RNN hidden units (the same amount for both) $\{96, 256\}$; the number of recurrent layers $\{1, 2, 3\}$; and the number of convolutional layers $\{1, 2, 3 ,4\}$ with the following frequency max pooling arrangements after each convolutional layer $\{(4), (2, 2), (4, 2), (8, 5), (2, 2, 2), (5, 4, 2), (2, 2, 2, 1), (5, 2, 2, 2)\}$. Here, the numbers denote the number of frequency bands at each max pooling step; e.g., the configuration (5, 4, 2) pools the original 40 bands to one band in three stages: 40 bands $\rightarrow$ 8 bands $\rightarrow$ 2 bands $\rightarrow$ 1 band. The final network configuration is given in Table 2.

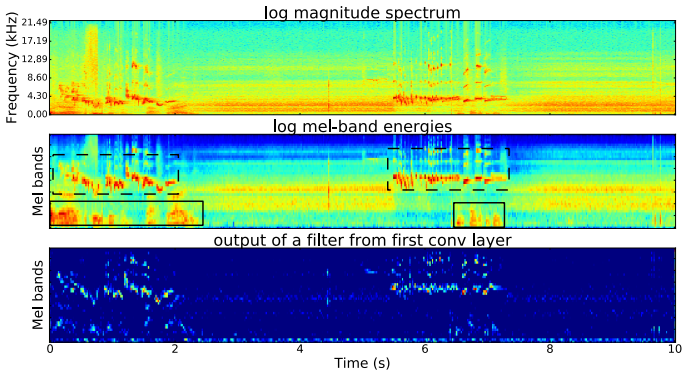


**Fig. 2**. Log magnitude spectrum (top), log mel-band energies (middle) and a single filter output from first convolutional layer (bottom) for *000a3cad-ef99-4e5e-9845.wav*. Dashed boxes mark the components due to bird sounds, and solid boxes mark the components due to two people speaking.

### 3.3. Baseline

In this work, we trained a CNN to be used as a baseline and also to understand the benefit of using recurrent layers after the convolutional layers. Based on the information given after the challenge, most of the submissions also use CNN as their classifier, and therefore it can be deemed as an appropriate baseline for the proposed method. The optimal parameters for CNN is found with a similar grid search as explained in Section 3.2, the only difference is that we replace the recurrent layers with feedforward layers. Each feedforward layer had shared weights between timesteps.

For comparison, we also provide the scores from the top three submissions for the challenge. Both methods use CNN as classifier (therefore labeled as *CNN2* [21] and *CNN3* [22]), they use mel spectrogram as input features, and they apply frequency and time shift as data augmentation techniques. Both methods apply pseudo-labeling (i.e. including the very confident detections from the test set into training set) and they further apply model ensembling over the networks.

## 4. RESULTS AND DISCUSSION

AUC scores for the baseline CNN and the proposed CRNN methods on development and evaluation sets are presented in Table 3. AUC for development set is obtained from the mean test AUC of the five splits. Although the performance difference between CNN and CRNN is minimal for the development data, CRNN performs significantly better for the evaluation data. Considering that the evaluation data includes recordings from different environmental and recording conditions than the development data, one can say that CRNN does a better job of generalizing over bird sounds in different conditions. For both methods, the validation data AUC score reaches to about 92% in the very first epoch and reaches its peak in about 20 epochs. To compare with the other top submissions, CNN2 reaches 88.7% AUC and CNN3 obtains 88.2% on the evaluation data.

**Table 3**. AUC scores on development and evaluation sets

| Dataset | Method | |
|---|---|---|
| | CNN | CRNN |
| Development | 95.3 | **95.7** |
| Evaluation | 85.5 | **88.5** |

In order to provide some insight on the features and network outputs, one of the recordings from the evaluation set (namely *000a3cad-ef99-4e5e-9845.wav*) has been specifically investigated. The top panel represents the magnitude spectrum (in log scale) for the recording, the middle panel shows the normalized log mel band energies which are used as input for the network, and the bottom panel represents the output from one of the filters in the first convolutional layer before max-pooling. When we compare the top two panels, we notice that with log mel band energies, the frequency components due to speech and bird sounds become very distinguishable. In addition, by looking at the filter outputs in the bottom panel, one can say that this filter has learned to react to the bird sound components and mostly ignore the rest for the given audio recording. The trained CRNN outputs a probability of 94.7% for a bird sound in this recording.

Since the amount of available material is quite large (about 68 hours), we did not further experiment on various data augmentation techniques. For the challenge submission, we experimented with a model ensemble method: 11 networks with the same architecture and different initial random weights (obtained by sampling from different random seeds) were trained and the estimated probabilities from each network were averaged to obtain the ensemble output. Although

this method improved the prior AUC results (calculated from a small portion of the evaluation data) from 88.3 to 89.4, it performed worse in the final results (88.2 vs. 88.5). The authors do not have a clear reasoning for this contradiction.

## 5. CONCLUSION

In this work, we propose using convolutional recurrent neural networks for bird audio detection as a part of a research challenge. The proposed method shows robustness for the local frequency shifts and is able to utilize longer term temporal information. Both of these features are essential for a generalized, context independent BAD system. The method achieves 88.5% AUC score and obtains the second place in Bird Audio Detection challenge.

## 6. REFERENCES


[1] R. T. Buxton and I. L. Jones, "Measuring nocturnal seabird activity and status using acoustic recording devices: applications for island restoration," *Journal of Field Ornithology*, vol. 83, no. 1, pp. 47–60, 2012.

[2] T. A. Marques *et al.*, "Estimating animal population density using passive acoustics," *Biological Reviews*, vol. 88, no. 2, pp. 287–309, 2013.

[3] A. L. Borker *et al.*, "Vocal activity as a low cost and scalable index of seabird colony size," *Conservation biology*, vol. 28, no. 4, pp. 1100–1108, 2014.

[4] M. Graciarena *et al.*, "Bird species recognition combining acoustic and sequence modeling," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 341–344.

[5] D. Stowell, M. Wood, Y. Stylianou, and H. Glotin, "Bird detection in audio: a survey and a challenge," in *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2016, pp. 1–6.

[6] S. N. Howell and S. Webb, *A guide to the birds of Mexico and northern Central America*. Oxford University Press, 1995.

[7] P. Ehrlich, D. Dobkin, and D. Wheye, "Birds of stanford essays." [Online]. Available: http://web.stanford.edu/group/stanfordbirds/text/uessays/essays.html

[8] E. P. Derryberry, "Ecology shapes birdsong evolution: variation in morphology and habitat explains variation in white-crowned sparrow song," *The American Naturalist*, vol. 174, no. 1, pp. 24–33, 2009.

[9] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," in *IEEE/ACM TASLP Special Issue on Sound Scene and Event Analysis*, 2017, accepted for publication.

[10] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4580–4584.

[11] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," *arXiv preprint arXiv:1609.04243*, 2016.

[12] "Detection and classification of acoustic scenes and events (DCASE)," 2016. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2016/task-sound-event-detection-in-real-life-audio

[13] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi-label deep neural networks," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015.

[14] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, 2015.

[15] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv:1412.6980 [cs.LG]*, 2014.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," in *Journal of Machine Learning Research (JMLR)*, 2014.

[19] F. Chollet, "Keras," github.com/fchollet/keras, 2015.

[20] S. Adavanne, D. Konstantinos, E. Cakir, and T. Virtanen, "Stacked convolutional and recurrent neural networks for bird audio detection," in *European Signal Processing Conference (EUSIPCO)*, 2017, submitted.

[21] T. Grill, "Source code for the BAD challenge submission, user: bulbul," https://jobim.ofai.at/gitlab/gr/bird_audio_detection_challenge_2017/tree/master, 2017.

[22] T. Pellegrini, "Source code for the BAD challenge submission, user: topel," github.com/topel/bird_audio_detection_challenge, 2017.

# Publication V

Emre Çakır, Tuomas Virtanen. "Convolutional Recurrent Neural Networks for Rare Sound Event Detection", in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017).* Munich, Germany, November 2017, pp. 27-31.

©2017 IEEE. Reprinted, with permission, from Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE2017).

# CONVOLUTIONAL RECURRENT NEURAL NETWORKS FOR RARE SOUND EVENT DETECTION

*Emre Çakır and Tuomas Virtanen*

Tampere University of Technology
Finland
emre.cakir@tut.fi

## ABSTRACT

Sound events possess certain temporal and spectral structure in their time-frequency representations. The spectral content for the samples of the same sound event class may exhibit small shifts due to intra-class acoustic variability. Convolutional layers can be used to learn high-level, shift invariant features from time-frequency representations of acoustic samples, while recurrent layers can be used to learn the longer term temporal context from the extracted high-level features. In this paper, we propose combining these two in a convolutional recurrent neural network (CRNN) for rare sound event detection. The proposed method is evaluated over DCASE 2017 challenge dataset of individual sound event samples mixed with everyday acoustic scene samples. CRNN provides significant performance improvement over two other deep learning based methods mainly due to its capability of longer term temporal modeling.

***Index Terms***— Sound Event Detection, Convolutional Neural Network, Recurrent Neural Network, Machine learning

## 1. INTRODUCTION

The aim of sound event detection (SED) is to temporally locate and label the sound event class(es) present in an acoustic signal. For an SED task, a set of target sound event classes should be determined. For instance, an SED task can be defined as the detection of dog barking, door bell, and baby crying sounds for any given acoustic signal. Recently, SED has been utilized in application areas such as wildlife bird audio monitoring [1, 2], audio surveillance [3], and multimedia event detection [4].

Recently, the research on SED has been mainly shifted from traditional classifier approaches such as Gaussian mixture models (GMM) - hidden Markov models (HMM) to deep learning based methods such as feed-forward neural networks (FNN) [5, 6], convolutional neural networks (CNN) [7], recurrent neural networks (RNN) [8], and convolutional recurrent neural networks (CRNN) [2, 9]. Feed-forward neural networks have the benefit of higher expressional capability over nonlinear functions compared to GMM-HMMs. However, their drawback is the fixed connections (each weight is connected to a fixed pair of neurons) which makes them less robust to slight spectral shifts in the acoustic features of the same sound event class. These slight shifts are a major factor in the inherent acoustic variability of sound event classes. This problem has mainly been overcome with the introduction of CNNs for SED, however the temporal context that can be modeled with CNNs is rather short. CRNN combines the long-term modeling capabilities of gated recurrent unit (GRU) [10] layers and the robustness of CNN to small spectral shift variations.

There are several difficulties on developing SED systems to be utilized in real-life environments. Some of these can be listed as the inherent acoustic variability of the sounds belonging to the same event class, overlapping (simultaneously occurring) sound events, environmental noise, variability in the acoustic characteristics of the background acoustic scene, and rarely occurring sound events.

The main problem encountered with the detection of the rare sound events using neural networks is the data imbalance. To elaborate, in an SED task, the classifier is trained to learn the relationship between the target class and its input representation, which is composed of acoustic features extracted in short time frames of an acoustic signal. During training, the classifier makes an estimation for the class presence probabilities for each frame, and calculates the error in the estimation through a loss function (which will be used to update the classifier parameters). In a rare SED task, the target class is not present in a significantly higher portion of time frames of each signal. Unless the training procedure of the classifier is adjusted correspondingly, the classifier will be biased on predicting ”non-present” for all the frames, because it will reach low error even if it fails to detect the frames where the target class is present. Data imbalance is a very common problem in machine learning and methods such as data augmentation using time stretching and block mixing [8], oversampling [11] and synthesizing new samples through generative methods [12] have been previously proposed to limit the negative effect of data imbalance.

In this work, we propose to utilize CRNNs for combined single-class, rare SED in the presence of a real-life acoustic scene in the background. The convolutional layers of CRNN are used to extract shift invariant features from the input time-frequency representation. The gated recurrent layers are especially effective in detecting rare sound events, because they can reset and update their hidden/cell state to distinguish the features from a small number of consecutive time frames (corresponding to a rare target event) which are noticeably different from the features from the rest of the acoustic signal (corresponding to the background). The proposed CRNN method has been previously shown to provide state-of-the-art accuracy in both real-life and synthetic SED datasets [9] and QMUL bird audio detection challenge 2017 [2]. We follow the similar CRNN architecture and procedure as in [9], with the exception that we train separate CRNNs for each class due to the combined single-class approach. In addition, we slightly adjust the training procedure according to the evaluation metric of the given SED task (see Section 3.2). This work has a companion website at [1].

The rest of the paper is organized as follows. The acoustic features and the proposed CRNN method is explained in Section 2. In Section 3, the acoustic material, evaluation metric and the eval-

[1] www.cs.tut.fi/~cakir/DCASE2017

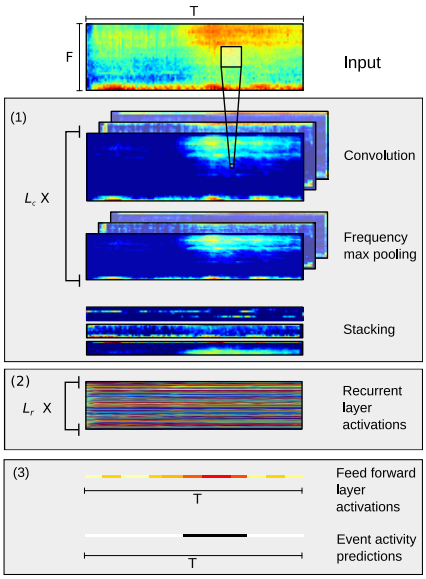


Figure 1: Overview of the proposed CRNN. (1): Multiple convolutional layers with max pooling in frequency axis, and stacking of the features over frequency axis (2): Gated recurrent layers, (3): feed-forward layer produces the event activity probabilities which are then binarized in evaluation/usage case.

uation results of the proposed method compared with the baseline methods is presented. Finally, our conclusions on this work are presented in Section 4.

## 2. METHOD

### 2.1. System Overview

The used SED approach consists of sound representation and frame-wise classification stages. In the sound representation stage, frame-level acoustic features are extracted for each time frame in the acoustic signal to obtain a feature matrix $\mathbf{X} \in \mathbb{R}^{F \times T}$, where $F \in \mathbb{N}$ is the number of features per frame and $T \in \mathbb{N}$ is the number of frames in the acoustic signal. In the classification stage, the task is to estimate the probabilities $p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})$ for target output vector $\mathbf{y} \in \mathbb{R}^T$, where $\mathbf{y}$ denotes the probability of the target event in each frame and $\boldsymbol{\theta}$ denotes the parameters of the classifier. Once the method is to be evaluated or utilized in a usage case, the event activity probabilities are typically binarized by thresholding, *e.g.* over a constant, to obtain binary event activity predictions $\hat{\mathbf{y}} \in \mathbb{R}^T$.

The classifier parameters $\boldsymbol{\theta}$ are trained by supervised learning, and the target outputs $\mathbf{y}$ are obtained from the onset-offset annotations of the sound event class. If the sound event class is present during frame $t$, then $\mathbf{y}_t$ will be set to 1, and 0 otherwise.

In this work, SED is conducted in combined single-class manner, so the stages below are repeated separately for each class.

### 2.2. Acoustic Features

The acoustic features used in this work are log mel-band energies, as they have been shown to provide good performance on SED with deep neural networks [2, 6, 9]. Each audio sample is divided into 40 ms frames with 50% overlap and 40 log mel-band energy features are extracted from the magnitude spectrum of each frame. Each feature is then normalized independently to zero mean and unit standard deviation by using statistics calculated from the training data.

### 2.3. CRNN Architecture

The CRNN architecture used in this work consists of three main blocks: (1) convolution block, (2) recurrent block, and (3) classification block. The illustration of the architecture is given in Figure 1. The input for the CRNN are the acoustic features (log mel-band energies). In the convolution block, the input is fed to $L_c$ consecutive convolutional layers with linear activation functions. Each convolutional layer is followed by batch normalization per feature map [13], a rectified linear unit (ReLU) activation function, a dropout layer [14], and a frequency domain max-pooling layer. At the end of the convolutional block, the extracted features over the CNN feature maps are stacked along the frequency axis.

Convolutional layers provide robustness to frequency shifts in the input features due to shared weight connections and max-pooling operation, and this is crucial to overcome the problem of intra-class acoustic variability for SED. However, as it has been shown previously in other works [9, 15], convolutional layers perform the the best when the filter size is small, and this means the temporal context used in these layers is very short (typically less than two hundred miliseconds).

In the recurrent block, these stacked features are fed to $L_r$ GRU layers where tanh and hard sigmoid activation functions are used for update and reset gates, respectively. Each recurrent layer produces outputs for each frame by using both the features extracted by the convolutional layers (or the previous recurrent layers) and the previous frame activations as input. Dropout is applied on both the inputs and the hidden state outputs of the recurrent layer [16].

GRU layers control the information flow through a gated unit structure. For frame $t$, the total activation of GRU layer is a linear interpolation of previous activation $h_{t-1}$ and the candidate activation $\hat{h}_t$ as

$$h_t = u_t \cdot h_{t-1} + (1 - u_t) \cdot \hat{h}_t \tag{1}$$

where $u_t$ denotes the update gate. Candidate activation $\hat{h}_t$ is a function of $h_{t-1}$, the GRU layer's input $x_t$ and the reset gate $r_t$. GRU activation is mainly controlled by reset gate when the GRU layer's input $x_t$ is significantly different than in previous frames. When reset gate is closed ($r_t = 0$), the candidate activation does not include any contribution from $h_{t-1}$. Fast response to the changes in the input and the previous activation information is crucial for high performance in rare SED, where the task is to detect a small of consecutive time frames where target event is present.

In the classification block, a feed-forward layer of single unit with sigmoid activation function is used as the classification layer. While computing the output of the classification layer, the same weight and bias values are used over the recurrent layer outputs for each frame. The contributions of GRU's previous and candidate activations to the classification output, namely $c_{t-1}$ and $\hat{c}_t$, can be computed as

$$\begin{aligned} c_{t-1} &= w \odot (u_t \cdot h_{t-1}) \\ \hat{c}_t &= w \odot ((1 - u_t) \cdot \hat{h}_t) \end{aligned} \tag{2}$$

Table 1: CRNN hyper-parameters for each target class.

| Hyper-parameters | Baby cry | Glass break | Gun shot |
|---|---|---|---|
| $L_c$ | 3 | 3 | 3 |
| pool size | (5,4,2) | (5,4,2) | (5,4,2) |
| $L_r$ | 1 | 3 | 1 |
| # filters/units | 96 | 160 | 32 |
| # Parameters | 520K | 1750K | 59K |

where $w$ is the weight vector that connects GRU layer and the classification layer, and $\odot$ denotes element-wise multiplication. The outputs of the classification layer are regarded as the presence probabilities of the target class in each frame of the audio sample.

If the model is to be evaluated or utilized in a usage case, the presence probabilities are binarized with a constant threshold of 0.5 to get the binary presence predictions. These predictions are further post-processed with a median filter of length 540 ms.

## 3. EVALUATION

### 3.1. Acoustic Material

For the acoustic material, DCASE2017 challenge dataset has been used and detailed information on the dataset can be found in Section 4 of [17]. The dataset consists of samples from 15 different everyday acoustic scenes (park, home, street, cafe, train etc.), some of which are mixed with isolated recordings from at most one of the three different target sound event classes: baby crying, glass breaking and gun shot. The isolated recordings are divided into segments based on the signal energy levels, and the segments relevant to the target class are selected by a human annotator. Mixing is done by adding a segment to the 30-second long background acoustic scene sample with a random time offset. The mean duration of the isolated target sound event recordings is below 2.25 seconds for all three classes and each isolated event is present at most once for each mixed sample, making them active for only a short period of time (hence the task name rare sound event detection).

For the development set, 2973 training, 298 validation and 1496 test samples (4767 total) are generated through the code repository provided as a part of the DCASE challenge [18]. Although the probability of including isolated recordings in each mixed sample is set to 0.5 as default in the code provided by the challenge, we increase the probability of including target events from default 0.5 to 0.99 for training and validation samples. This change increases the percentage of the frames labeled as including a target event from 5% to 8% in the training data, which helps to ease the problem of data imbalance. This probability is kept at 0.5 for the test samples, as suggested by the challenge organizers, to be able to compare the development set results over the same conditions with other participants. In the evaluation set, the training and validation samples of the development set are combined into a single training set, test samples are used as the validation set, and the system is evaluated against an unseen set of 1500 samples (500 for each target class).

### 3.2. Procedure and Final Configuration

The CRNN is trained using Adam method for gradient based optimization [19]. Cross-entropy is used as the loss function. The network is trained for a maximum of 200 epochs. After each epoch of training, validation set is evaluated for the event-based error rate (see Section 3.4) and the model at the epoch with the lowest error rate is saved in the memory. This way, we aim to align the training procedure with the evaluation metric of this work. If the error rate does not decrease for 25 consecutive epochs, the training is stopped and the last saved model is selected as the final model.

In order to decide the architecture to be used in the evaluation, we run a hyper-parameter grid search and pick the architecture with the lowest event-based error rate on the test set of the development data. The fixed hyper-parameters for each experiment is as follows. We use 5-by-5 size feature maps in convolutional layers, and dropout with probability 0.25 for both convolutional and recurrent layers. The grid search covers the number of convolutional feature maps (filters) / RNN hidden units (both are set to the same value) {32, 96, 160}; the number of recurrent layers {1, 2, 3, 4}; and the number of CNN layers {1, 2, 3 ,4} with the following frequency max-pool sizes after each convolutional layer {(8), (4, 2), (2, 2, 2), (5, 2, 2), (5, 4, 2), (5, 2, 2, 1), (5, 2, 2, 2)}. The best performing CRNN hyper-parameters for each target class are listed in Table 1.

### 3.3. Baseline

In this work, we compare the performance of CRNN with two baseline methods using deep learning with the same input features. The first baseline method is a deep FNN with two hidden layers of 50 units, which is also the official baseline method for the challenge. The input features differ slightly in the sense that the extracted 40 log mel-band energy features are concatenated for five consecutive frames to gather temporal context, creating a feature vector with 200 entries. The second baseline method is the CNN. While selecting the CNN architectures to be used in evaluation, a very similar grid search procedure has been applied as explained in Section 3.2, the only difference being that the recurrent layers of the CRNN are replaced with the feed-forward layers to obtain CNN architecture.

### 3.4. Evaluation Metric

The official evaluation metric used in DCASE2017 challenge task 2 is the event-based error rate (ER) with onset tolerance of 500 ms. ER is the sum of insertion, deletion and substitution rates. ER is calculated as explained in detail in [20].

### 3.5. Results

The models used in the evaluation (hence the challenge submission) have been selected as the following. As a part of hyper-parameter grid search, 84 experiments have been run on development data for CNN and CRNN each. The evaluation models are then selected based on ER on test set of development data. We present four different CRNN methods for the rare SED challenge which are labeled as:

- CRNN-1: the architecture with the lowest ER on average over three classes. This model also happens to have the lowest ER on the "baby cry" class, and its parameters are given in the corresponding column of Table 1.
- CRNN-2: the ensemble of the seven best architectures with the lowest ER on average over three classes.
- CRNN-3: the architecture with the lowest ER for each of the three classes. The parameters for each of the architectures are given in Table 1.

Table 2: Event-based error rate for the baseline FNN and CNN methods and the proposed CRNN on the test set of evaluation data. Method indices are explained in Section 3.5.

| Method | Evaluation | | | |
|---|---|---|---|---|
| | Baby cry | Glass break | Gun shot | Average |
| FNN | 0.80 | 0.38 | 0.73 | 0.64 |
| CNN-1 | 0.46 | 0.13 | 0.58 | 0.39 |
| CNN-2 | 0.38 | 0.15 | 0.53 | 0.35 |
| CNN-3 | 0.46 | 0.14 | 0.55 | 0.39 |
| CNN-4 | 0.42 | 0.14 | 0.53 | 0.36 |
| CRNN-1 | 0.27 | **0.07** | **0.20** | 0.18 |
| CRNN-2 | **0.18** | 0.10 | 0.23 | **0.17** |
| CRNN-3 | 0.27 | 0.14 | 0.47 | 0.29 |
| CRNN-4 | 0.21 | 0.11 | 0.24 | 0.19 |

- CRNN-4: the ensemble of seven best architectures with the lowest ER for each class.
- CNN methods have been obtained in the same fashion to CRNN methods as explained above.

The ensemble method is conducted as follows. Among the seven selected architectures, if four or more predict that the target class is not present in a given sample, then the final decision on the sample is that the target class is not present. Otherwise, the onset and offset values of the target class are selected as the median of the predicted onset and offset values for the sample. This ensemble method is used in order to get more reliable predictions over the onset and offset values and to filter the outlier predictions among the architectures with lowest ER.

The event-based ER results for the proposed and baseline methods have been presented in Table 2. CRNNs clearly provide better performance compared to both baseline methods for all target classes. In addition, by utilizing ensemble methods for both CNN and CRNN, the performance can be further improved, however this comes with an increased computational cost due to running several architectures in parallel. With the experiments for CRNN-1 method, we aimed to show if it is possible to find a single architecture that performs well for all three classes. For the development data, CRNN-1 provides comparable performance (0.16 vs 0.14 ER) with CRNN-3, where the best architecture is selected for each target class. For the evaluation data, as presented in Table 2, CRNN-1 performs even better than CRNN-3.

Regardless of the method, highest performance is obtained for glass break. Although the best performing architectures for each class differ significantly in the number of parameters (see Table 1), the median number of parameters among the seven best architectures are 687K, 806K and 774K for baby cry, glass break and gun shot, respectively. Therefore it is not possible to draw any direct conclusions on the relationship between the target class and the best performing architecture size.

### 3.6. An insight on GRU layer of CRNN

A case-study demonstration of the effect of GRU layers on the CRNN outputs is given in Figure 2. The CRNN architecture used to create this illustration consists of three convolutional layers and one GRU layer with 32 filters/units each, followed by a single unit classification layer. In panel (a), we can see that multiple GRU units respond to the change of input features around 4.5 second

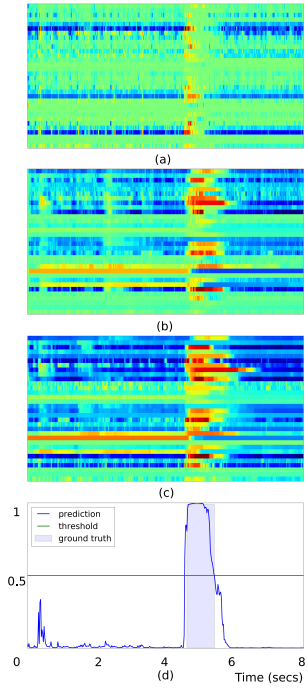


Figure 2: (a): contribution of current (candidate) activation $\hat{c}_t$, (b): contribution of previous activation $c_{t-1}$, (c): total contribution of GRU layer to the classification activation; (d): event activity probabilities vs. time for the first eight seconds of sample *devtest_babycry_001_1128b63726e9ed59ddc1bb944b3f22ce.wav*.

mark and trigger the candidate activation, as the target event starts to appear in the audio signal. After that, the GRU contribution is mainly controlled by the previous activations while the target event is still present, as shown in panels (b) and (c). Finally, the CRNN produces almost perfect detection of onset and offset for the given target event, as shown in panel (d).

## 4. CONCLUSIONS

In this paper, CRNN has been proposed for rare SED. CRNN has provided significantly improved performance over FNNs and CNNs for every target sound event class in DCASE 2017 challenge dataset. It is shown that the performance can further be improved using ensemble methods. For future work, improved ways to incorporate the evaluation metric into training procedure as the objective function can be considered. For instance, instead of aiming to directly match the target output and the predicted output for each frame, the objective function can be calculated over a window of frames, especially for the case when the onset and offset times can be tolerated to a certain degree.

## 5. REFERENCES


[1] D. Stowell and D. Clayton, "Acoustic event detection for multiple overlapping similar sources," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2015, pp. 1–5.

[2] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," in *European Signal Processing Conference (EUSIPCO)*, 2017, accepted.

[3] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Reliable detection of audio events in highly noisy environments," *Pattern Recognition Letters*, vol. 65, pp. 22–28, 2015.

[4] Y. Wang, L. Neves, and F. Metze, "Audio-based multimedia event detection using deep recurrent neural networks," in *2016 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2742–2746.

[5] I. Choi, K. Kwon, S. H. Bae, and N. S. Kim, "DNN-based sound event detection with exemplar-based approach for noise reduction," DCASE2016 Challenge, Tech. Rep., September 2016.

[6] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi-label deep neural networks," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015.

[7] T. Lidy and A. Schindler, "CQT-based convolutional neural networks for audio scene classification and domestic audio tagging," DCASE2016 Challenge, Tech. Rep., September 2016.

[8] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *2016 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 6440–6444.

[9] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[10] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[11] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," *Advances in intelligent computing*, pp. 878–887, 2005.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 448–456.

[14] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] Y. Gal, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in neural information processing systems*, 2016.

[17] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.

[18] T. Heittola. (2016) Dcase2017 baseline system. [Online]. Available: github.com/TUT-ARG/DCASE2017-baseline-system

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

# Publication VI

Emre Çakır, Ezgi Ozan, Tuomas Virtanen. "Filterbank Learning for Deep Neural Network Based Polyphonic Sound Event Detection", in *Proceedings of the International Joint Conference on Neural Networks (IJCNN).* Vancouver, Canada, July 2016, pp. 3399-3406.

©2016 International Neural Network Society (INNS). Reprinted, with permission, from E. Çakır, E. Ozan, and T. Virtanen, Filterbank Learning for Deep Neural Network Based Polyphonic Sound Event Detection, Proceedings of the International Joint Conference on Neural Networks, July 2016.

# Filterbank Learning for Deep Neural Network Based Polyphonic Sound Event Detection

Emre Cakir, Ezgi Can Ozan, Tuomas Virtanen

***Abstract*— Deep learning techniques such as deep feedforward neural networks and deep convolutional neural networks have recently been shown to improve the performance in sound event detection compared to traditional methods such as Gaussian mixture models. One of the key factors of this improvement is the capability of deep architectures to automatically learn higher levels of acoustic features in each layer. In this work, we aim to combine the feature learning capabilities of deep architectures with the empirical knowledge of human perception. We use the first layer of a deep neural network to learn a mapping from a high-resolution magnitude spectrum to smaller amount of frequency bands, which effectively learns a filterbank for the sound event detection task. We initialize the first hidden layer weights to match with the perceptually motivated mel filterbank magnitude response. We also integrate this initialization scheme with context windowing by using an appropriately constrained deep convolutional neural network. The proposed method does not only result with better detection accuracy, but also provides insight on the frequencies deemed essential for better discrimination of given sound events.**

## I. Introduction

A sound event is an audio segment that can be labeled as a distinctive concept in an audio signal. Some examples of sound events can be given as dog bark, car horn, footsteps etc. Automatic detection of sound events has recently drawn a surging interest especially in accordance with the invention of modern machine learning techniques. Sound event detection has applications in smart homes [1], scene recognition for mobile robots [2] and surveillance in living environments [3], [4].

An audio analysis system can be divided into two stages as *sound representation* and *classification* of consecutive time frames. In order to extract the relevant information, the audio signals are often transformed into a higher level representation. This representation is obtained by the *acoustic features*. In audio related classification and detection tasks such as speech recognition, music classification, acoustic scene classification and sound event detection, there are certain conventional, hand-crafted acoustic features that have proved to perform in a satisfactory level. These features are commonly extracted from the time-frequency representations of the audio signals, *i.e.*, spectrograms. Some of the conventional acoustic features can be listed as spectral energy, zero-crossing rate [5], (log) mel band energy [6] and mel-frequency cepstral coefficients (MFCC) [7]. The spectrogram of an audio signal can also be regarded as an image, which makes it possible to apply and extend hand-crafted image feature extraction methods on sound event detection. Consequently, local spectrogram features [8], histogram of gradients [9], [10] and Gabor filterbank features [4] have been proposed recently for sound event detection.

The evidence from auditory psychophysics proves that humans perceive sound signals along a nonlinear scale in frequency domain [11]. Human ear is more sensitive to the changes in the lower frequencies than in the higher frequencies. Consequently, acoustic features are often chosen by using nonlinear scales (log frequency scale, mel scale, bark scale etc.) to give better correspondence with human perception. Mel scale is a perceptual scale in which the pitches are adjusted by the listeners so that the successive pitches are perceived to have equal distance among the scale [12]. In sound related classification tasks, acoustic features that utilize the mel scale have been found to provide robust sound representation and have become standard features. Mel band energy and especially MFCCs can be given as examples for such features [6], [7]. On the other hand, the empirical nature of mel scale leads to several hand-crafted mel formulas [12], [13], [14], [15], [16], all of which provide only an approximation to the real human perception. There are also certain views that mel scale needs to be readdressed due to biased experiments, namely the original experiments did not take into account the hysteresis, *i.e.*, human perception varying when the frequencies are listened in the ascending order than in descending order [17]. To sum up, mel scale does not provide a perfect fit for human perception. On the other hand, the goal in polyphonic sound event detection is not modeling perfectly the human perception, but to obtain high detection accuracy. Nevertheless these two goals are closely related to each other, and mel scale representation can be a good starting point for acoustic features in a sound event detection task.

For the classification stage, the traditional approaches in sound event detection and other audio related classification tasks have been Gaussian mixture modeling (GMM) [4], Hidden Markov modeling (HMM) [18] and Support Vector Machines (SVM) [19]. Recently, deep neural networks (DNN), which are neural networks with multiple hidden layers, have been shown to give better results in sound event detection [6], [20], [21], [22], [23].

Deep learning architectures have opened an alternative path to hand-crafted data representation methods. In a deep architecture, with each hidden layer, the input is transformed to a higher level representation which is to be classified at the

*The research leading to these results has received funding from the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND. Computing resources of *Taito* supercluster from Finnish Grid Infrastructure has been used in this work.

output layer. The layer parameters such as weights and biases are generally initialized with small random values. While backpropagating the error derivatives during the training stage, these weights and biases are updated and higher level features are obtained to present a better model for the target output, *i.e.*, features are *learned* through the input data. Due to their high expressional capability, deep classifiers do not rely on the high level hand-crafted representations of the *raw* data as their input and they are able to express the highly nonlinear relationship between the raw data and the target output. This led to a trend of using raw data (pixel values for an image [24] or magnitude/power spectrum for audio [25], [26], [27] or even raw audio [28]) as input and deep learning methods as the classifier in machine learning tasks.

In our recent work [6], we found that for the given polyphonic environmental sound recordings, the optimal multi-label detection method was using mel band energies as acoustic features and DNN with two hidden layers. DNNs with mel band energy features outperformed the traditional MFCC feature with GMM-HMM classifier method by a huge margin. The ability of DNNs to use subsets of their hidden units to detect several events simultaneously makes them a suitable choice for polyphonic sound event detection tasks, where multiple events are occuring simultaneously [6], [25]. DNNs also tend to perform better with lower level acoustic features (such as mel band energies) compared to traditional higher level features (such as MFCCs) [29]. This can be explained with the fact that MFCCs are obtained by applying discrete cosine transform (DCT) on mel band energy features to decorrelate these features, but DNNs are already very powerful in modeling data correlation and do not necessarily require a DCT step [30].

In this work, we propose to integrate the empirically obtained human perception information with feature learning capabilities of the deep architectures to learn a new filterbank. This ad-hoc filterbank is initialized with a human perception based method and tuned with deep learning techniques. We use the high-resolution magnitude spectrum of an audio signal (which can be regarded as low level representation) as the input for deep feedforward and deep convolutional neural networks (CNN). Instead of initializing all the weights and biases with small random values, we initialize the first hidden layer weights of the network as the coefficients of a human-perception based filterbank magnitude response, namely mel filterbank. During training, the first hidden layer weights, *i.e.*, the filterbank magnitude response is updated to provide better discrimination over the target sound events. The trained network does not only provide better detection accuracy over completely randomly initialized networks, but also, depending on the updates made on the first hidden layer weights during training, it gives an idea of which frequency bins are deemed more essential for discrimination in a given sound event detection task.

The organization of this paper is as follows. Polyphonic sound event detection task is explained in detail in Section II. The proposed mel scale filterbank and mel scale weight initialization technique for deep neural networks is proposed in Section III. Acoustic data and network settings used in the work is explained and an evaluation of the proposed methods is presented in Section IV. Finally, our conclusions on the subject is presented in Section V.

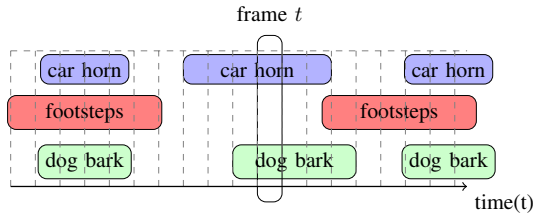


Fig. 1: Overlapping sound events in a recording from a realistic environment. Frame $t$ represents the short time frame from the part of the audio signal where only car horn and dog bark events are present.

## II. Sound Event Detection

The aim of sound event detection is to temporally locate the sound events and assign a class label to each event present in an audio signal. In a real-life situation, there can be multiple sound events present simultaneously, *i.e.*, the audio signal can be polyphonic at a certain time. An example of this is illustrated in Figure 1, where an audio signal is labeled with the sound events happening in time domain. Sound event detection can be formulated as a multi-label classification problem in which the temporal locations of sound events are obtained by doing binary classification of the activity of each sound event class over consecutive time frames.

In the sound representation stage, the audio signal $x(t)$ is first divided into $N$ time frames by using *e.g.* Hamming window of length 40 ms and 50% overlap. Then, the magnitude spectrum matrix $\mathbf{X}$ for the audio signal $x(t)$ is obtained by taking the absolute value of the Fast Fourier transform (FFT) of the short time frames. The traditional approach is to compute further higher level representations from $\mathbf{X}_{m,n}$, where $m$ and $n$ are the frequency bin and frame indices. Instead, we add another hidden layer of the deep architecture and initialize that layer's weights with mel filterbank magnitude response to automatically learn the higher level representations. This technique is explained in detail in Section III.

The task in the classification stage is to model the nonlinear relationship between the magnitude spectrum $\mathbf{X}$ and the target outputs $\mathbf{Y} \in \mathbb{R}^{K \times N}$ as

$$\mathbf{X} \rightarrow \alpha(\cdot) \rightarrow \mathbf{Y} \tag{1}$$

where $\alpha$ is the classifier model, $\mathbf{Y}$ is the binary matrix encoding the present events in frame $n$ and $K$ is the pre-determined number of events. If the event $k$ is present in a frame $n$, $\mathbf{Y}_{k,n}$ is set to 1 and otherwise 0. In multi-label sound event detection, the pre-determined events can occur

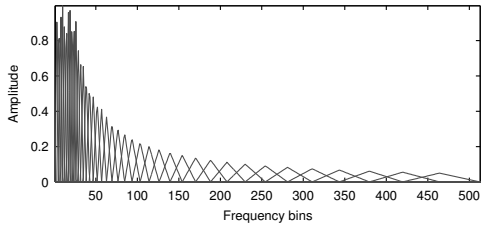


Fig. 2: Mel filterbank magnitude response.

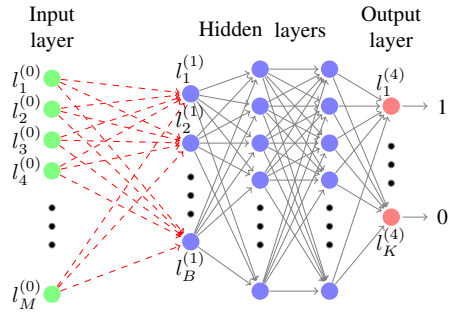


Fig. 3: Symbolic representation of the DNN topology with mel scale weight initialization. The dashed red weight connections between $l^{(0)}$ and $l^{(1)}$ are initialized with mel filterbank magnitude response $\mathbf{F}_{b,m}$ where $b \in [1, ...B]$ and $m \in [1, ...M]$.

in any different combination at a time. So, a vector of length $K$ is required to encode the target outputs $\mathbf{Y} \in \mathbb{R}^{K \times N}$.

The classifier model $\alpha$ is trained using annotated material. The start and end times of each event in the audio signal $x(t)$ is annotated and used to obtain the binary target outputs $\mathbf{Y}$. The annotation procedure is illustrated in Figure 1 and explained in more detail in Section IV-A.

## III. Method

In this work, we combine the following findings:

- Mel scale approximates the human hearing perceptually better than a linear frequency scale [12],
- New improvements on DNN architectures and learning are needed to push the features even further back to the raw levels of acoustic measurements [30].

We use magnitude spectrum of short time frames as input for a DNN and initialize the first hidden layer weights of a DNN with mel scale filterbank magnitude response. This is done to integrate the empirical human perception knowledge in the automatic feature learning. In order to introduce the temporal information encoded in the consecutive time frames, we use appropriately constrained CNN with context window input. We obtain the binary detection output by regarding the network outputs as posterior probabilities and thresholding these probabilities with a constant $\phi$.

### A. Mel scale filterbank

Mel scale is a perceptually motivated scale that is designed so that the intervals consist of equal perceptual pitch increments. The reference point is often chosen as setting 1000 mels to 1 kHz. The relation between mel scale frequency $mel$ and frequency $f$ (Hz) is commonly defined with the following formula [15]:

$$mel = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2}$$

First, the lowest and highest edges of the frequency range $[0, \frac{Fs}{2}]$ are converted into mel frequencies according to (2), where $Fs$ is the sampling rate. Then, the mel range is equally spaced into $B$ mel bands. Mel scale filterbank magnitude response $\mathbf{F}_{b,m}$ is designed according to this mel scale, where $b$ and $m$ are the mel filter and frequency bin indices, respectively. The magnitude response of the filterbank is approximated using triangular band-pass filters. For each mel filter, the triangular band-pass filter magnitude response $\mathbf{F}_{b,m}$ is selected in the range $[0, 1]$, where the response is 0 outside a band, and from band start rises linearly to peak 1, from which it linearly decays to 0 at band end. The slope is larger at the lower frequencies where the mel bands contain fewer frequency bins, and vice versa. Then, the magnitude response for each filter is scaled so that the magnitude sum for each filter is approximately constant. In this work, the resource code in [31] is used to obtain the mel scale filterbank magnitude response explained above and visualized in Figure 2.

Mel band magnitudes $\mathbf{E}$ are computed from the magnitude spectrum $\mathbf{X}$ and mel scale filterbank magnitude response $\mathbf{F}$ as

$$\mathbf{E}_{b,n} = \sum_{m=1}^{M} \mathbf{F}_{b,m} \mathbf{X}_{m,n} \tag{3}$$

where $M$ is the half of the number of FFT bins in the magnitude spectrum. In this work, 1024 point DFT is used, therefore $M = 1024/2 + 1 = 513$. Negative frequency terms of $\mathbf{X}$ are discarded because the FFT is computed for real input and therefore $\mathbf{X}$ is Hermitian-symmetric, where the negative frequency terms consist of complex conjugate values of the positive frequency terms.

### B. DNN with mel scale weight initialization

As the input for the DNN, we use the first half of the magnitude spectrum, which is a lower level representation compared to traditional acoustic features such as MFCCs. As in [6], DNN structure consists of two hidden layers before the output layer. The difference with the architecture in [6] is that we add another hidden layer $l^{(1)}$ of $B$ hidden units at the front part of the DNN architecture, *i.e.*, between the input layer and the first hidden layer. For a given input vector $\mathbf{x}$ (where $\mathbf{x} = \mathbf{X}_{:,n}$, time frame index $n$ is omitted for simplicity), the input layer $l^{(0)}$ is set to $\mathbf{x}$ and the hidden

neuron outputs $\mathbf{z}_i^{(1)}$ for the layer $l^{(1)}$ are computed as

$$\mathbf{z}_i^{(1)} = \theta(\sum_{j=1}^{M} \mathbf{W}_{i,j}^{(1)} \mathbf{x}_j) \qquad (i = 1, 2, ...B) \tag{4}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{B \times M}$ are the hidden neuron weights and $\theta$ is the activation function. The output layer outputs $\mathbf{z}^{(4)} \in [0, 1]$ are treated as the posterior probability $\tilde{y}$ for each event class. Selecting $\theta$ as rectified linear unit will result in the same structure in computing mel band magnitudes as in Eq. 3 and computing feedforward neural network outputs as in Eq. 4. Instead of initializing the weights $\mathbf{W}^{(1)}$ with small random values, we use mel filterbank magnitude response $\mathbf{F}_{b,m}$, so that this layer will learn a filterbank that provides better discrimination of the given sound events. The process is illustrated in Figure 3. For the output activations, logistic sigmoid function is used to obtain multi-label outputs.

Mel scale filterbank magnitude response $\mathbf{F}_{b,m}$ is a sparse matrix due to the fact that mel bands span only a few frequency bins in the linear scale, especially for the lower frequencies. On the other hand, randomly initialized weights of the rest of the architecture, *i.e.*, weights for layers $l^{(2)}$, $l^{(3)}$ and $l^{(4)}$ are sampled from a uniform distribution $U(-\Delta, \Delta)$, where $\Delta$ is calculated from the randomized initialization method proposed in [32]. The random weights drawn from a uniform distribution $U(-\Delta, \Delta)$ have zero mean by definition and their variance can be calculated as $\sigma^2 = \frac{1}{12}(-\Delta - \Delta)^2$. Using $\mathbf{F}_{b,m}$ directly to initialize the weights $\mathbf{W}^{(1)}$ would result in a mismatch in terms of mean and variance with the rest of the architecture, which may slow down the learning. Therefore, the mean of $\mathbf{F}_{b,m}$ coefficients is subtracted and the variance is matched with the uniformly distributed weights before initialization. In order to avoid some neurons having the same initial weights as zero and introducing an undesired symmetry to initialization after mean and variance matching, the zero weights of $\mathbf{W}^{(1)}$ are replaced with small random values. This initial mel filterbank response is illustrated in Figure 6(a).

### C. DNN with context windowing

Context windowing, *i.e.*, expanding the input instance from the features of a single frame to a neighbourhood of frames, has been shown to improve the performance significantly in sound event detection with a DNN classifier [29]. In context windowing for DNN, the input $\mathbf{X}_{:,n}$ is concatenated with $\frac{C-1}{2}$ preceding and $\frac{C-1}{2}$ succeeding frame features to form context window $\hat{\mathbf{X}}_{:,n}$ as

$$\hat{\mathbf{X}}_{:,n}^T = \left[ \mathbf{X}_{:,n-\frac{C-1}{2}}^T \cdots \mathbf{X}_{:,n+\frac{C-1}{2}}^T \right] \tag{5}$$

where superscript $T$ represents the the transpose.

While using DNNs, context windowing can be done by concatenating input frame features in single dimension. However, this approach can be deemed inefficient, as the information that some of the features actually come from consecutive time frames is discarded while presenting the input to DNN. In addition, when the number of input features is high, the number of parameters to be learned between the input and the first hidden layers increase accordingly. Moreover, combining context windowing with mel scale weight initialization in a DNN architecture would result in separate learned filterbanks for each frame, unless a weight sharing technique is used. Therefore, in this work, context windowing for DNN input instances is only used in the baseline mel band magnitude feature experiments. The experiments involving mel scale weight initialization and context windowing are conducted with CNN architectures.

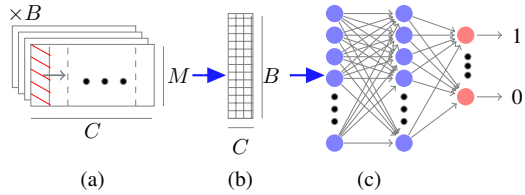


Fig. 4: An illustration of the CNN topology with mel scale weight initialization. Cascaded windows represent the $B$ feature maps and the crossed section represents the convolutional filter shape $M \times 1$.

### D. CNN with mel scale weight initialization and context windowing

CNNs can be regarded as the extensions of DNNs that can also learn the spatial and/or temporal information encoded in the multi-dimensional input features. CNNs have three properties that can be used to explain their difference with DNNs: local receptive field, weight sharing and pooling. Local receptive field means that each hidden neuron in a CNN layer is not connected to all the neurons in the previous layer, but only a local region of the input neurons to the convolutional layer. Multiple local fields with the same shape can be defined to extract different features in each region by working parallel to each other, and the weights and biases of each of these local fields define a feature map. The local region is shifted through all the input neurons and the same local region weights and biases are applied through the whole input space, *i.e.*, the weights are shared throughout the input. Finally, a pooling layer is often added after the convolutional layer to subsample the output and reduce the translational variations on the feature maps.

In this work, we aim to benefit from the temporal information encoded in the consecutive time frames (by using context windowing) and at the same time keep the filterbank structure in the first hidden layer of the deep architecture. In order to combine context windowing with mel scale weight initialization in a deep architecture, the weights in the first hidden layer are initialized with mel filterbank magnitude response and shared over the input frame features in the context window. This structure essentially becomes a constrained version of CNN using context windows as input.

The context window $\hat{\mathcal{X}} \in \mathbb{R}^{M \times C \times N}$ is formed as

$$\hat{\mathcal{X}}_n = \begin{bmatrix} \mathbf{X}_{1,n-\frac{C-1}{2}} & \mathbf{X}_{1,n-\frac{C+1}{2}} & \dots & \mathbf{X}_{1,n} & \dots & \mathbf{X}_{1,n+\frac{C-1}{2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{X}_{M,n-\frac{C-1}{2}} & \mathbf{X}_{M,n-\frac{C+1}{2}} & \dots & \mathbf{X}_{M,n} & \dots & \mathbf{X}_{M,n+\frac{C-1}{2}} \end{bmatrix} \quad (6)$$

where $\hat{\mathcal{X}}_n = \hat{\mathcal{X}}_{:,:,n}$.

After context windowing, each input instance for CNN can be regarded as the spectrogram for $C$ frames, as shown in Figure 4(a). The target output for each input instance $\hat{\mathcal{X}}_n$ will be the same as the target output for the center frame of the context window, *i.e.*, $\hat{\mathbf{Y}}_{:,n} = \mathbf{Y}_{:,n}$. The number of feature maps in the convolutional layer is fixed to $B$ and for each feature map, the convolutional filter shape is fixed to $M \times 1$. Given the context window $\hat{\mathbf{X}}$ (where $\mathbf{X} = \mathcal{X}_{:,:,n}$, time frame index $n$ is omitted for simplicity), the output of the convolutional layer $\mathbf{Z} \in \mathbb{R}^{C \times B}$ is calculated as

$$\mathbf{Z}_{c,b} = \theta(\sum_{m=1}^{M} \mathbf{W}_{b,m} \hat{\mathbf{X}}_{m,c}) \qquad (c = 1, 2, ...C) \quad (7)$$

where $\mathbf{W}_{b,:}$ represents the weights for the $b^{th}$ feature map. The complete CNN topology corresponds to a single convolutional layer without pooling connected to two fully connected hidden layers before the output layer as illustrated in Figure 4(c).

We implement the mel scale weight initialization in the way that we introduce $B$ feature maps with dimensions $M \times 1$, and initialize the weights $\mathbf{W}_{b,m}$ with mel scale filterbank magnitude response $\mathbf{F}_{b,m}$. This way, each feature map is initialized with the coefficients of the corresponding mel band and filterbank learning is introduced to the deep architecture in the convolutional layer. As the name suggests, each *feature map* learns a different higher level *feature* and the learning includes making updates on the standard mel filterbank magnitude response.

### E. Decision Thresholding

In a neural network based sound event detection system, logistic sigmoid output of the network can be treated as the posterior probability $\tilde{y}$ for each event class for each time frame, and the posterior probabilities of the sound events produced by a deep learning classifier should be converted into a binary detection format. In a real-life sound event detection task, the number of positive detections in a frame $n$ is not constant and it is in the range of $[0, K]$. In order to obtain multiple binary detection outputs in a single frame, posterior probabilities for each sound event is thresholded with a constant $\phi$ (in this work, $\phi = 0.5$ is used to set an unbiased probability threshold).

## IV. Evaluation

### A. Acoustic Data

We evaluate the performance of the proposed method with synthetic material that consists of sequences of occasionally overlapping sound events. There are 9 sound event classes that have been investigated in this work: alarm clock, baby crying, cat meowing, dog barking, door bell, footsteps, glass shattering, music and speech. The samples for these classes except music and speech are collected from *stockmusic* [1]. The total length of these samples is around 4270 seconds, with an average length of 17 seconds per sample. In addition, the speech samples are the clean samples from the training set (all speakers) of the 2nd CHiME challenge [33] and the music samples are 15-20 second randomly selected excerpts from music tracks with various genres [34]. The samples for each class are distributed randomly as 60 % in training set, 20 % in validation set and 20% in test set. In order to simulate the polyphony in real-life recordings, the collected samples are mixed in 15 minute mixtures in the following way. Among the samples collected from *stockmusic*, the samples having the same track are grouped together in either training, validation or test sets to avoid that samples recorded in the same environment would be easily recognized by the factors other than their sound characteristics. For each mixture, randomly selected samples from the event classes are added to the mixture by introducing a random time delay, adding a sample to the mixture and repeating this until the end of the mixture. By doing this for each class, a mixture with time-varying polyphony is obtained. The polyphony is roughly controlled with the range of the random time delay, as illustrated in Figure 5. The ground truth activity for each mixture is obtained by automatically annotating (labeling) the frames in the mixture where the sample from the corresponding event class is used. In order to avoid mislabeling of possible silent segments in the beginning and the end of the samples, mean RMS energy of the sample is calculated in short frames, and then, compared to this mean RMS energy, the frames in the beginning and end of the sample without enough RMS energy are left unlabeled. A total of 50 mixtures obtained this way for the whole dataset. Among the annotated frames, the polyphony percentage of the test set is given in Table I.

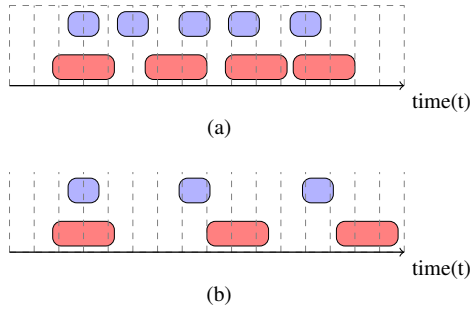


Fig. 5: The polyphony of the mixture is controlled with the length of the time delay in between samples: (a) high polyphony, (b) low polyphony.

[1] http://www.stockmusic.com

TABLE I: Polyphony level versus data amount percentage for the annotated frames in the test set.

| Polyphony | 1 | 2 | 3 |
|---|---|---|---|
| Percentage | 90.9 | 8.5 | 0.5 |

### B. Settings

In order to obtain the magnitude spectrum, the mixtures are divided into 40 ms time frames with 50% overlap. Hamming window is applied for each frame to reduce the boundary effect. Magnitude spectrum for each time frame is calculated by taking the absolute value of 1024-point FFT. The baseline model uses 40 mel band energies and two hidden layers for the DNN [6]. Consequently, our DNN method uses a fully-connected hidden layer with 40 neurons and for CNN, this hidden layer is replaced with convolutional layer with 40 feature maps. For both DNN and CNN methods, the first hidden layer is followed by 2 hidden layers of 200 neurons. During the training with stochastic gradient descent algorithm, learning rate is set to 0.1 and Nesterov accelerated momentum with value 0.2 is used. These network hyper-parameters are selected according to grid search over the validation set. Constant thresholding of the network outputs with $\phi = 0.5$ is used to get the binary outputs. Each experiment has been repeated ten times and the results are presented as the average accuracy over the experiments. This is done to take into account the accuracy changes due to random initialization of the network parameters. The experiments are conducted based on Keras neural networks library [35].

### C. Evaluation Metric

The evaluation metric for this work is frame-wise multi-label F1 score. Correct, wrong and missed detections of events are obtained for each input instance by comparing the estimated binary outputs to the target outputs. Then, precision and recall is calculated from the total number of correct, wrong and missed detections of events throughout the whole test set. F1 score is calculated as the harmonic mean of precision and recall.

### D. Overall Results

DNN and CNN architectures with different features, first hidden layer weight initialization methods and context window lengths has been experimented and the accuracy results are presented in Table II. Regardless of the context window length $C$, deep architectures trained with magnitude spectrum features outperform the mel band magnitude features. This can be seen as a promising step on using lower level features as input and automatically learning the higher level features through deep learning in sound related tasks. Moreover, CNNs whose first layer weights are initialized with mel filterbank magnitude response generally outperform the randomly initialized CNNs.

In addition to accuracy benefit, mel filterbank weight initialization also has an *understandability* benefit. Randomly initialized weights have an implicit final structure since

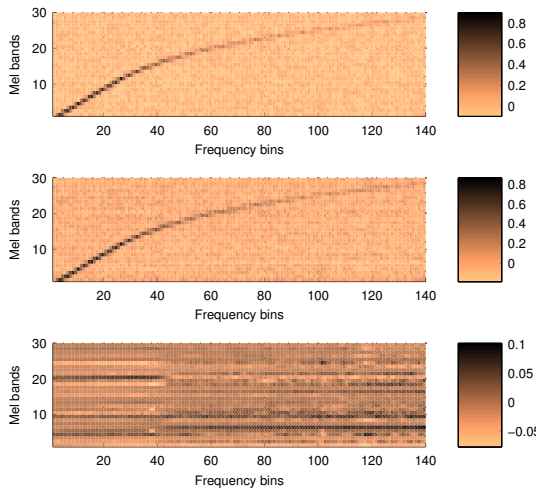


Fig. 6: (a): initial mel filterbank response, (b): final mel filterbank response, (c): difference between (a) and (b).

they map a very complex nonlinear function starting from random coefficients for this function. On the other hand, mel initialized CNNs provide understandable weights in the first hidden layer, where one can deduct the essential frequency bins for the particular sound event detection task. The initial and final mel filterbank responses for $C = 11$ is visualized in Figure 6. The final filterbank clearly preserves the nonlinear mel scale structure. Moreover, when we look at the difference between initial and final mel filterbank responses as given in Figure 6(c), we notice that the frequency bins in some mel bands are emphasized by decreasing the weights of the frequency bins in the same filter, but outside the mel band. This points to the fact that the frequencies in the given mel band play an important role in the detection of the sound events.

Consistent with our observations in [29], introducing time information encoded in successive frames, *i.e.*, using context windows rather than individual frames, provides great boost in accuracy. The accuracy nevertheless converges at around $C = 11$, which corresponds to 240 ms of audio signal with the given settings. Using the appropriately constrained CNN allows us to introduce context windowing to mel filterbank weight initialization. Consequently, the best detection accuracy (marked with bold in Table II) is obtained when these two methods are combined.

### E. Class-wise Results

The detection accuracy for each sound event class is presented in Table III. Context window length for each method in Table III is selected as $C = 11$, for which the accuracy is highest as presented in Table II. In sound event detection, the input usually consists of unstructured sounds, which makes it hard to both model the sound events and interpret the accuracy of the proposed methods for each sound event

TABLE II: Detection accuracy as a function of acoustic feature, network architecture and context window length $C$. *MEL*: mel band energy, *FFT*: magnitude spectrum, *random init*: complete random initialization of network weights, *mel init*: mel filterbank weight initialization in the first hidden layer weights, *$\mu$ and $\sigma$ match*: mean and variance matching.

| Feature | Architecture | C=1 | C=3 | C=5 | C=7 | C=9 | C=11 | C=13 |
|---|---|---|---|---|---|---|---|---|
| MEL | DNN random init | 69.3 | 75.3 | 77.1 | 77.9 | 77.9 | 78.9 | 78.9 |
| FFT | CNN random init | 71.4 | 76.1 | 78.2 | 79.4 | 80.6 | 81.2 | 80.7 |
| FFT | CNN mel init | 71.5 | 75.8 | 78.5 | 80.2 | 81.2 | 80.8 | 80.8 |
| FFT | CNN mel init ($\mu$ and $\sigma$ match) | 71.3 | 76.2 | 78.4 | 79.7 | 80.9 | **81.8** | 81.3 |

TABLE III: Detection accuracy as a function of acoustic feature & network architecture and sound event classes for context window length $C = 11$.

| Feature & Architecture | alarm clock | baby cry | cat meow | dog bark | door bell | footsteps | glass shatter | music | speech |
|---|---|---|---|---|---|---|---|---|---|
| MEL & DNN random init | 61.0 | 77.0 | 79.1 | 69.1 | 66.0 | 76.0 | 75.0 | 89.6 | 88.0 |
| FFT & CNN random init | 70.8 | 76.9 | 78.8 | 74.8 | 64.5 | 78.9 | 81.3 | 90.5 | 89.1 |
| FFT & CNN mel init | 65.0 | 77.6 | 82.2 | 74.4 | 57.7 | 77.7 | 81.7 | 90.0 | 89.3 |
| FFT & CNN mel init ($\mu$&$\sigma$ match) | 74.3 | 76.5 | 81.8 | 74.9 | 69.6 | 75.9 | 82.0 | 90.3 | 88.8 |

separately. However, for the *more* structured classes such as alarm clock and door bell (which often consist of multiple sinusoids recorded simultaneously), mel filterbank weight initialization provides a significant boost in accuracy. For the classes such as baby crying, footsteps, music and speech the accuracy difference between the methods seems negligible.

## V. Conclusions

In this work, we aim to combine the automatic feature learning capabilities of the deep learning architectures with the empirically obtained human perception knowledge. Instead of using the traditional, perceptually motivated mel band magnitudes as input features, we use magnitude spectrum features and initialize the first hidden layer weights with mel filterbank magnitude response, which essentially corresponds to computing the mel band magnitudes through the first hidden layer outputs. During learning process, the filterbank is updated to provide better discrimination over sound events. The proposed method does not only improve the detection accuracy over randomly initialized networks, but also provide *interpretable* weights in the first hidden layer, which can be used to deduct the important frequency bins in the detection of the given sound events.

## References


[1] D. Hollosi, J. Schröder, S. Goetze, and J. Appell, "Voice activity detection driven acoustic event classification for monitoring in smart homes," in *3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010)*. IEEE, 2010, pp. 1–5.

[2] S. Chu, S. Narayanan, C. Kuo, and M. Mataric, "Where am I? Scene recognition for mobile robots using audio features," in *IEEE Int. Conf. Multimedia and Expo (ICME)*, 2006, pp. 885–888.

[3] A. Harma, M. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in *IEEE Int. Conf. Multimedia and Expo (ICME)*, 2005, pp. 4–pp.

[4] J. Geiger and K. Helwani, "Improving event detection for audio surveillance using Gabor filterbank features," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2015.

[5] J. T. Geiger, B. Schuller, and G. Rigoll, "Large-scale audio feature extraction and SVM for acoustic scene classification," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2013)*, 2013, pp. 1–4.

[6] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multilabel deep neural networks," in *Int. Joint Conf. on Neural Networks (IJCNN)*, 2015, pp. 1–7.

[7] J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[8] J. Dennis, H. Tran, and E. 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.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE, 2005, pp. 886–893.

[10] V. Bisot, S. Essid, and G. Richard, "HOG and subband power distribution image features for acoustic scene classification," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2015.

[11] J. B. Allen, "Cochlear modeling," *IEEE ASSP Magazine*, vol. 2, no. 1, pp. 3–29, 1985.

[12] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[13] S. S. Stevens and J. Volkmann, "The relation of pitch to frequency: A revised scale," *The American Journal of Psychology*, pp. 329–353, 1940.

[14] J. Makhoul and L. Cosell, "LPCW: An LPC vocoder with linear predictive spectral warping," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 1976, pp. 466–469.

[15] D. O'shaughnessy, *Speech communication: human and machine*. Universities press, 1987.

[16] D. D. Greenwood, "A cochlear frequency-position function for several species 29 years later," *The Journal of the Acoustical Society of America*, vol. 87, no. 6, pp. 2592–2605, 1990.

[17] D. Greenwood, "The mel scale's disqualifying bias and a consistency of pitch-difference equisections in 1956 with equal cochlear distances and equal frequency ratios," *Hearing research*, vol. 103, no. 1-2, pp. 199–224, 1997.

[18] T. Heittola, A. Mesaros, T. Virtanen, and M. Gabbouj, "Supervised model training for overlapping sound events based on unsupervised source separation," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013, pp. 8677–8681.

[19] D. Sadlier, N. E. O'Connor *et al.*, "Event detection in field sports video using audio-visual features and a support vector machine," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1225–1233, 2005.

[20] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2015, pp. 1–6.

[21] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2015, pp. 559–563.

[22] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic

events using deep neural networks," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2014.
[23] M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani, "Exploiting spectro-temporal locality in deep learning based acoustic event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, pp. 1–12, 2015.
[24] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
[25] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. Sainath, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
[26] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014.
[27] T. N. Sainath, B. Kingsbury, A. Mohamed, and B. Ramabhadran, "Learning filter banks within a deep neural network framework," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2013, pp. 297–302.
[28] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2015.
[29] E. Cakir, "Multilabel sound event classification with neural networks," Master's thesis, Tampere University of Technology, Finland, 2014.
[30] L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, and J. Williams, "Recent advances in deep learning for speech research at Microsoft," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8604–8608.
[31] D. P. W. Ellis, "PLP and RASTA (and MFCC, and inversion) in Matlab," 2005, online web resource. [Online]. Available: http://www.ee.columbia.edu/ dpwe/resources/matlab/rastamat/
[32] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.
[33] E. Vincent *et al.*, "The second chimespeech separation and recognition challenge: Datasets, tasks and baselines," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2013, pp. 126–130.
[34] T. Heittola, "Automatic classification of music signals," Master's thesis, Department of Information Technology, Tampere University Of Technology, 2004.
[35] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

# Publication VII

Emre Çakır, Tuomas Virtanen. "End-to-End Polyphonic Sound Event Detection Using Convolutional Recurrent Neural Networks with Learned Time-Frequency Representation Input", in *Proceedings of the International Joint Conference on Neural Networks (IJCNN).* Rio de Janeiro, Brazil, July 2018, pp. 2412-2418.

©2018 IEEE. Reprinted, with permission, from E. Çakır, and Tuomas Virtanen, End-to-End Polyphonic Sound Event Detection Using Convolutional Recurrent Neural Networks with Learned Time-Frequency Representation Input, Proceedings of the International Joint Conference on Neural Networks, July 2018.

# End-to-End Polyphonic Sound Event Detection Using Convolutional Recurrent Neural Networks with Learned Time-Frequency Representation Input

Emre Çakır
*Tampere University of Technology, Finland*
emre.cakir@tut.fi

Tuomas Virtanen
*Tampere University of Technology, Finland*
tuomas.virtanen@tut.fi

***Abstract*—Sound event detection systems typically consist of two stages: extracting hand-crafted features from the raw audio waveform, and learning a mapping between these features and the target sound events using a classifier. Recently, the focus of sound event detection research has been mostly shifted to the latter stage using standard features such as mel spectrogram as the input for classifiers such as deep neural networks. In this work, we utilize end-to-end approach and propose to combine these two stages in a single deep neural network classifier. The feature extraction over the raw waveform is conducted by a feedforward layer block, whose parameters are initialized to extract the time-frequency representations. The feature extraction parameters are updated during training, resulting with a representation that is optimized for the specific task. This feature extraction block is followed by (and jointly trained with) a convolutional recurrent network, which has recently given state-of-the-art results in many sound recognition tasks. The proposed system does not outperform a convolutional recurrent network with fixed hand-crafted features. The final magnitude spectrum characteristics of the feature extraction block parameters indicate that the most relevant information for the given task is contained in 0 - 3 kHz frequency range, and this is also supported by the empirical results on the SED performance.**

***Index Terms*—neural networks, convolutional recurrent neural networks, feature learning, end-to-end**

## I. Introduction

Sound event detection (SED) deals with the automatic identification of the sound events, *i.e.*, sound segments that can be labeled as a distinctive concept in an audio signal. The aim of SED is to detect the onset and offset times for each sound event in an audio recording and associate a label with each of these events. At any given time instance, there can be either a single or multiple sound events present in the sound signal. The task of detecting a single event at a given time is called monophonic SED, and the task of detecting multiple sound events is called polyphonic SED. In recent years, SED has been proposed and utilized in various application areas including audio surveillance [1], urban sound analysis [2], multimedia event detection [3] and smart home devices [4].

The research leading to these results has received funding from the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND. The authors wish to acknowledge CSC IT Center for Science, Finland, for providing computational resources.

SED has traditionally been approached as a two-stage problem: first, a time-frequency representation of the raw audio signal is extracted, then a classifier is used to learn the mapping between this representation and the target sound events. For the first stage, magnitude spectrograms, and human perception based methods such as mel spectrograms and mel frequency cepstral coefficients (MFCC) have been the most popular choices among SED researchers, and they have been used in a great portion of the submissions for the two recent SED challenges [5], [6]. For the second stage, deep learning methods such as convolutional and recurrent neural networks have recently been dominating the field with state-of-the-art performances [7]–[9].

Using time-frequency representations are beneficial in the following ways. Compared to raw audio signal in time domain, frequency domain content matches better with the semantic information about sounds. In addition, the representation is 2-D, which makes the vast research on classifiers on image-based recognition tasks applicable to SED. Also, they are often more robust to noisy environments than raw audio signals (as the noise and the target sources can occupy different regions in the frequency domain), and the obtained performance is often better than using the raw audio signals as input to the second stage. On the other hand, especially for human perception based representations, it can be argued that these representations utilize domain knowledge to discard some information from the data, which could have been otherwise useful given the optimal classifier method.

### A. Related Work

Recently, classifiers with high expression capabilities such as deep neural networks have been utilized to learn directly from raw representations in several areas of machine learning. For instance, in image recognition, since the deep learning methods have been found to be highly effective with the works such as AlexNet [10], hand-crafted image features have been mostly replaced with raw pixel values as the inputs for the classifiers. For speech recognition, similar performance have been obtained for raw audio and log mel spectrograms in using convolutional, long-short term memory deep neural network (CLDNN) classifiers [11]. For music genre recognition, raw audio input for a CNN gives close performance to mel spectro-

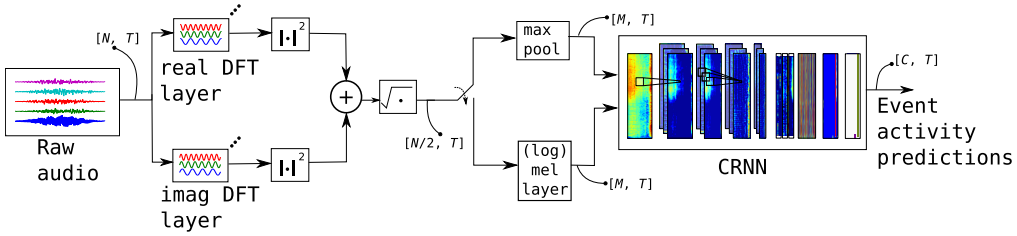


Fig. 1. Method framework. The method output shape in various stages of the framework is given in brackets.

grams [12]. Both [11] and [12] claim that when the magnitude spectra of the filter weights of the first convolutional layers are calculated and then visualized with the order of lowest dominant frequency to the highest, the resulting scale resembles the perception based scales such as mel and gammatone. For speech emotion recognition, a CLDNN classifier similar to [11] with raw audio input outperforms the standard hand-crafted features in the field, and it provides on-par performance with the state-of-the-art on a baseline dataset [13].

However, when it comes to SED, hand-crafted time-frequency representations are still found to be more effective than raw audio signals as the classifier input. In [14], raw audio input performs considerably worse than concatenated magnitude and phase spectrogram features. In [15], deep gated recurrent unit (GRU) classifier with raw audio input ranks poorly compared to time-frequency representation based methods in DCASE2017 challenge sub-task on real-life SED [6]. Most likely due to the poor performance, the research on the end-to-end methods for SED has recently been very limited, and only two out of 200 submissions have used raw audio as classifier input (with low success) in DCASE2017 SED challenge [6]. As an attempt to move towards lower level input representations for SED, in [16], magnitude spectrogram has been used an input to a deep neural network whose first layer weights were initialized with mel filterbank coefficients.

### B. Contributions of this work

In this work, we propose to use convolutional recurrent neural networks (CRNN) with learned time-frequency representation inputs for end-to-end SED. The most common time-frequency representations consist of applying some vector multiplications and basic math operations (such as sum, square and log) over raw audio signals divided into short time frames. This can be implemented in the form of a neural network layer, and the benefit is that the parameters used in the vector multiplications can be updated during network training to optimize the classifier performance for the given SED task. In this work, we investigate this approach by implementing magnitude spectrogram and (log) mel spectrogram extraction in the form of a feature extraction layer block, whose parameters can be adapted to produce an optimized time-frequency representation for the given task. We then compare the adapted parameters with the initial parameters to gain insight on the neural network optimization process for the feature extraction block. To our knowledge, this is the first work to integrate and utilize domain knowledge into a deep neural network classifier in order to conduct end-to-end SED. The main differences between this work and the authors' earlier work on filterbank learning [16] are the input representation (raw audio vs. magnitude spectrogram), spectral domain feature extraction block using neural network layers and the classifier (CRNN vs. FNN and CNN).

## II. Method

The input $\mathbf{X} \in \mathbb{R}^{N \times T}$ consists of $T$ frames of raw audio waveforms sampled of $N$ samples with sampling rate $F_s$, and Hamming window with $N$ samples is applied to each frame. Initially (*i.e.* before the network training), the output of the feature extraction block is either max pooled magnitude spectrogram, mel spectrogram or log mel spectrogram. The method framework is illustrated in Figure 1.

### A. Feature Extraction block

The input $\mathbf{X}$ to the feature extraction block is fed through two parallel feedforward layers, $l^{\text{re}}$ and $l^{\text{im}}$, each with $N$ neurons with linear activation function and no bias. The weights of these two layers, namely $\mathbf{W}^{\text{re}} \in \mathbb{R}^{\frac{N}{2} \times N}$ and $\mathbf{W}^{\text{im}} \in \mathbb{R}^{\frac{N}{2} \times N}$, are initialized so that the outputs of these layers for each frame $\mathbf{X}_{:,t}$ $(t = 1, ...T)$ would correspond to the real and the imaginary parts of the discrete Fourier transform (DFT):

$$
\begin{aligned}
\mathbf{F}_{k,t} &= \sum_{n=0}^{N-1} \mathbf{X}_{n,t}[\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)] \\
\mathbf{W}^{\text{re}}_{k,n} &\leftarrow \cos(2\pi kn/N) \\
\mathbf{W}^{\text{im}}_{k,n} &\leftarrow \sin(2\pi kn/N) \\
\mathbf{Z}^{\text{re}}_{k,t} &= \sum_{n=0}^{N-1} \mathbf{W}^{\text{re}}_{k,n} \mathbf{X}_{n,t} \\
\mathbf{Z}^{\text{im}}_{k,t} &= \sum_{n=0}^{N-1} \mathbf{W}^{\text{im}}_{k,n} \mathbf{X}_{n,t}
\end{aligned}
\tag{1}
$$

for $k = 0, 1, ..., \frac{N}{2} - 1$ and $n = 0, ..., N - 1$, where $\mathbf{Z}$ is the weighted output for each feedforward layer. The reason for taking only the first half of the DFT bins is that the raw audio waveform input $\mathbf{X}$ is purely real, resulting with a symmetric magnitude spectrum. Each weight vector $\mathbf{W}_{k,:}$ can be deemed

as an individual sinusoidal filter. For both $l^{\mathrm{re}}$ and $l^{\mathrm{im}}$, the outputs given the input $\mathbf{X}$ is calculated using the same weights $\mathbf{W}^{\mathrm{re}}$ and $\mathbf{W}^{\mathrm{im}}$ for each of the $T$ frames. Both layers are followed by a square operation, the outputs of the layers are summed, and finally a square root operator results with the magnitude spectrogram $\mathbf{S} \in \mathbb{R}^{\frac{N}{2} \times T}$:

$$\mathbf{S}_{k,t} = |\mathbf{F}_{k,t}| = \sqrt{(\mathbf{Z}^{\mathrm{re}}_{k,t})^2 + (\mathbf{Z}^{\mathrm{im}}_{k,t})^2} \tag{2}$$

At this stage, $\mathbf{S}$ can be directly fed as input to a CRNN classifier, or it can be further processed to obtain $M$ (log) mel spectrogram using a feedforward layer with $M$ neurons, rectified linear unit (ReLU) activations and no bias:

$$\mathbf{Z}^{\mathrm{mel}}_{m,t} = \max(0, \sum_{k=0}^{N/2-1} \mathbf{W}^{\mathrm{mel}}_{m,k} \mathbf{S}_{k,t}) \tag{3}$$

for $m = 0, 1, ...M-1$. The weights $\mathbf{W}^{\mathrm{mel}}$ of this layer is initialized with the mel filterbank coefficients in the similar manner with [16] and log compression is used in part of the experiments as

$$\mathbf{Z}^{\mathrm{logmel}} = \log(\mathbf{Z}^{\mathrm{mel}} + \epsilon) \tag{4}$$

where $\epsilon = 0.001$ is used to avoid numerical errors. The parameters $\mathbf{W}^{\mathrm{mel}}$ are obtained from *Librosa* [17] package and the center frequencies for each mel band are calculated using O'Shaughnessy's formula [18]. For the experiments where this layer is utilized, the weights $\mathbf{W}^{\mathrm{re}}$ and $\mathbf{W}^{\mathrm{im}}$ are kept fixed, as explained in Table I.

In our experiments while using $\mathbf{S}$ directly as the input for CRNN, we observed that when the number of features for $\mathbf{S}$ is dropped from $\frac{N}{2}$ to $M$ by using max-pooling in frequency domain, the computation time is substantially reduced with very limited decrease in accuracy. Hence, we followed this approach when the mel feature layer is omitted.

### B. Convolutional Recurrent block

Following the same approach with [8], the CRNN block consists of three parts:

1) convolutional layers with ReLU activations and non-overlapping pooling over frequency axis
2) gated recurrent unit (GRU) [19] layers, and
3) a single feedforward layer with $C$ units and sigmoid activation, where $C$ is the number of target event classes.

The output of the feature extraction block, *i.e.*, a sequence of feature vectors, is fed to the convolutional layers and the activations from the filters of the last convolutional layer are stacked over the frequency axis and fed to the GRU layers. For each frame, GRU layer activations are calculated using both the current frame input and the previous frame outputs. Finally, the GRU layer activations are fed to the fully-connected layer. The output of this final layer is treated as the event activity probability for each event. The aim of the network learning is to get the estimated frame-level class-wise event activity probabilities as close as to their binary target outputs, where target output is 1 if an event class is present in a given frame, and 0 vice versa. In the usage case, the estimated frame-level event activity probabilities are thresholded with 0.5 to obtain binary event activity predictions. More detailed explanation about CRNN block can be found in [8].

TABLE I
A TABLE SHOWING WHICH WEIGHT MATRICES ARE LEARNED FOR EACH EXPERIMENT. ✓ STANDS FOR LEARNED, ✗ STANDS FOR FIXED, AND - STANDS FOR NOT UTILIZED IN THE EXPERIMENT.

| Learned? | $\mathbf{W}^{\mathrm{re}}$ | $\mathbf{W}^{\mathrm{im}}$ | $\mathbf{W}^{\mathrm{mel}}$ |
|---|---|---|---|
| DFT learned | ✓ | ✓ | - |
| Mel learned | ✗ | ✗ | ✓ |
| Log mel learned | ✗ | ✗ | ✓ |

The network is trained with back-propagation through time using Adam optimizer [20] with learning rate $10^{-3}$, binary cross-entropy as the loss function and for maximum 300 epochs. In order to reduce overfitting of the model, early stopping was used to stop training if the validation data frame-level F1 score did not improve for 65 epochs. For regularization, batch normalization [21] was employed in convolutional layers and dropout [22] with rate 0.25 was employed in convolutional and recurrent layers. Keras deep learning library [23] was used to implement the network.

## III. Evaluation

### A. Dataset

The dataset used in this work is called *TUT-SED Synthetic 2016*. It is a publicly available polyphonic SED dataset, which consists of synthetic mixtures created by mixing isolated sound events from 16 sound event classes. Polyphonic mixtures were created by mixing 994 sound event samples with the sampling rate 44.1 kHz. From the 100 mixtures created, 60% are used for training, 20% for testing and 20% for validation. The total length of the data is 566 minutes. Different instances of the sound events are used to synthesize the training, validation and test partitions. Mixtures were created by randomly selecting event instance and from it, randomly, a segment of length 3-15 seconds. Mixtures do not contain any additional background noise. Dataset creation procedure explanation and metadata can be found in the web page [1] hosting the dataset.

### B. Evaluation Metrics and Experimental Setup

The evaluation metrics used in this work are frame-level F1 score and error rate. F1 score is the harmonic mean of precision and recall, and error rate is the sum of the rate of insertions, substitutions and deletions. Both metrics are calculated in the same manner with [8] and they are explained in more detail in [24].

The input $\mathbf{X}$ to the feature extraction block consists of a sequence of 40 ms length frames with 50% overlap. The number of frames in the sequence is $T = 256$ which corresponds to 2.56 seconds of raw audio. The audio signals have

[1] http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/tut-sed-synthetic-2016

TABLE II
FRAME-LEVEL F1 SCORE $F1_{\text{frm}}$ AND ERROR RATE $ER_{\text{frm}}$ RESULTS FOR DIFFERENT TIME-FREQUENCY REPRESENTATION METHODS AND SAMPLING RATES. "DFT" STANDS FOR MAGNITUDE SPECTROGRAM USING LINEAR FREQUENCY SCALE, "MEL" STANDS FOR MEL SPECTROGRAM, "FIXED" AND "LEARNED" STANDS FOR WHETHER THE WEIGHTS OF THE FEATURE EXTRACTION BLOCK ARE KEPT FIXED OR UPDATED DURING TRAINING.

| Method | $F1_{\text{frm}}$ | $ER_{\text{frm}}$ |
|---|---|---|
| DFT 8 kHz fixed | 60.8±0.8 | 0.55±0.01 |
| DFT 8 kHz learned | 60.8±0.8 | 0.55±0.01 |
| Mel 8 kHz fixed | 60.8±0.9 | 0.55±0.01 |
| Mel 8 kHz learned | 61.0±0.8 | 0.56±0.01 |
| Log mel 8 kHz fixed | 63.1±0.6 | 0.52±0.01 |
| Log mel 8 kHz learned | 58.6±1.6 | 0.56±0.01 |
| DFT 16 kHz fixed | 61.9±0.9 | 0.54±0.01 |
| DFT 16 kHz learned | 60.1±1.7 | 0.58±0.03 |
| Mel 16 kHz fixed | 62.3±0.7 | 0.54±0.01 |
| Mel 16 kHz learned | 60.6±0.9 | 0.57±0.02 |
| Log mel 16 kHz fixed | 65.8±1.4 | 0.50±0.01 |
| Log mel 16 kHz learned | 59.9±1.3 | 0.56±0.01 |
| DFT 24 kHz learned | 58.1±1.6 | 0.59±0.03 |
| Log mel 44.1 kHz fixed [8] | 66.4±0.6 | 0.48±0.01 |

been resampled from the original rate of 44.1 kHz to 8, 16 and 24 kHz in different experiments, which corresponds to $N = 160$, 320, and 480 features for each frame, respectively. This is done both to investigate the effect of discarding the information from higher frequencies, and also to reduce the memory requirements to be able to run experiments with a decent sized network and batch size. At the max pooling (or mel) layer of the feature extraction block, the number of features is set to $M = 40$.

In order to find the optimal network hyper-parameters, a grid search was performed, and the hyper-parameter set resulting with the best frame-level F1 score on the validation data was used in the evaluation. The grid search consists of every possible combination of the following hyper-parameters: the number of convolutional filters / recurrent hidden units (the same amount for both) {96, 256}; the number of recurrent layers {1, 2, 3}; and the number of convolutional layers {1, 2, 3 ,4} with the following frequency max pooling arrangements after each convolutional layer {(4), (2, 2), (4, 2), (8, 5), (2, 2, 2), (5, 4, 2), (2, 2, 2, 1), (5, 2, 2, 2)}. Here, the numbers denote the number of features at each max pooling step; e.g., the configuration (5, 4, 2) pools the original 40 features in a single feature in three stages: 40→8→2→1. This grid search process is repeated for every experiment setup in Table II (except the last experiment, where a similar grid search has been performed earlier for that work).

After finding the optimal hyper-parameters, each experiment is run ten times with different random seeds to reflect the effect of random weight initialization in convolutional recurrent block of the proposed system. The mean and the standard deviation (given after ±) of these experiments are provided.

### C. Results

The effect of feature extraction with learned parameters have been investigated and compared with the fixed feature extraction parameters in Table II. For both frame-level F1 score and error rate metrics, experiments with fixed feature extraction parameters often outperform the learned feature extraction methods in their corresponding sampling rates. In addition, the experiments with fixed parameters benefit from the increased sampling rate, whereas the performance does not improve for learned feature extraction parameters with higher sampling rates. One should also note that the F1 score using both learned and fixed parameters with 8 kHz sampling rate is 60.8%. Although there is some drop in performance from the highest F1 score of 66.4% at 44.1 kHz, it is still remarkable performance considering that about 82% of the frequency domain content of the original raw audio signal is discarded in the resampling process from 44.1 kHz to 8 kHz. This emphasizes the importance of low frequency components for the given SED task. Since the computational load due to high amount of data in the raw audio representations is one of the concerns for end-to-end SED systems, it can be considered to apply a similar resampling process for the end-to-end SED methods in the future.

In order to investigate how the original parameters of the feature extraction block have been modified during the training, the magnitude spectrum peak, *i.e.* the maximum value of the magnitude spectrum, of the trained weights for $\mathbf{W}_k^{\text{re}}$, $\mathbf{W}_k^{\text{im}}$, and $\mathbf{W}_k^{\text{re}} + i \cdot \mathbf{W}_k^{\text{im}}$ are calculated for each filter $k$. Without network training, these weights represent sinusoid signals, therefore the magnitude spectrum of each filter is equal to a single impulse at the center frequency of the filter, whose amplitude equals to the number of filters. At the end of the training, the peak of the magnitude spectrum for each filter stays at the center frequency of the filter, while the amplitude of the peak is either increased or decreased to a certain degree. In order to visualize the change in the peak amplitude, the peak amplitude positioned at the center frequency for each filter after training is given in Figure 2. The same analysis is repeated for different experiments using raw audio inputs with different sampling rates (8 kHz, 16 kHz and 24 kHz) as input to their feature extraction block which initially calculates the pooled magnitude spectrogram. The magnitude spectrum peaks for each experiment is scaled with the number of filters for visualization purposes, and therefore each peak is equal to 1 before the training. The three observations that can be made from Figure 2 is

- Although each of these three systems have different CRNN architectures (grid search for each system results with different hyper-parameter set) and their raw audio input is sampled with different rates, the magnitude

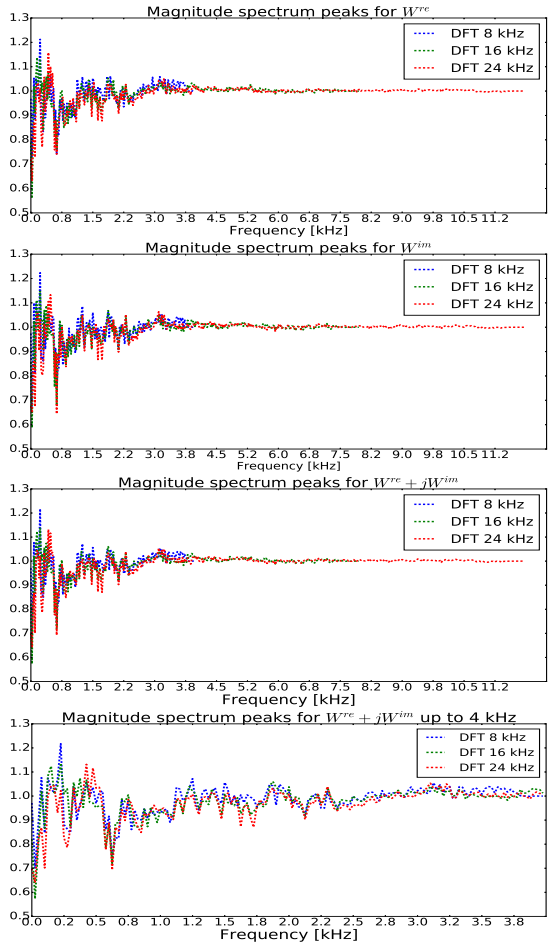


Fig. 2. Magnitude spectrum peaks for the real ($\mathbf{W}^{\mathrm{re}}$) and imaginary ($\mathbf{W}^{\mathrm{im}}$) DFT layer filters after the training. The amplitude of the peak for each filter is positioned at the center frequency of the corresponding filter, resulting with a line plot covering the whole frequency range for the experiment with given sampling rate.

spectrum peaks possess very similar characteristics. For all three experiments, the peaks are modified the most for the frequencies below around 3 kHz, and there is little to no change in peak amplitudes after 4 kHz. This may indicate that the most relevant information for the given SED task is in 0-4 kHz region. Although the authors cannot conclude this, it is empirically supported to a certain degree with the results presented in Table II. Even though the amount of data from the raw audio input sampled with 44.1 kHz is substantially reduced by resampling with 8 and 16 kHz, the performance drop is limited.

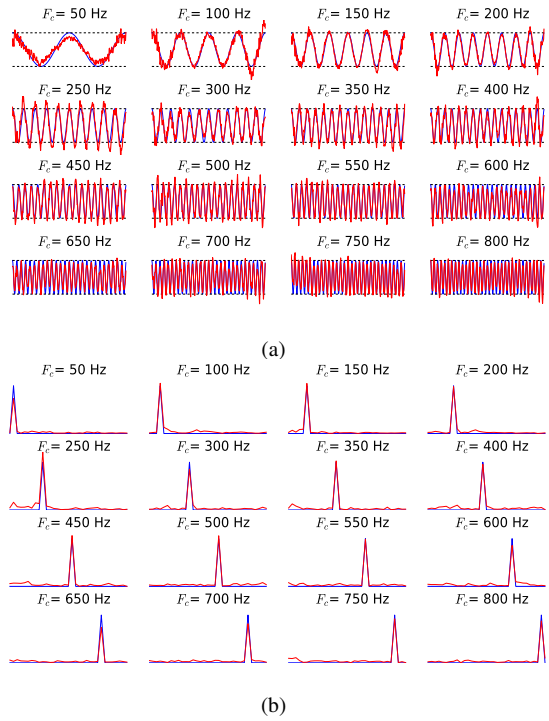


Fig. 3. (a): Real ($\mathbf{W}^{\mathrm{re}}$) DFT layer filters with different center frequencies $F_c$ and (b): their magnitude spectra. Blue plot represents initial values, and red plot represents the values after the training. Horizontal dashed lines at -1 and 1 mark the initial maximum and minimum values for the filters.

- For all three experiments, the change in the magnitude spectrum peaks is not monotonic in the frequency axis. Some of the peaks in the low frequency range are boosted, but there are also other peaks in the same frequency region that are significantly suppressed. This implies a different optimal time-frequency representation than both magnitude and mel spectrogram. One should also bear in mind that this learned representation is task-specific, and the same approach for other classification tasks may lead to a different ad-hoc magnitude spectrum peak distribution.
- Although they represent different branches of the feature extraction block (and therefore are updated with different gradients), the magnitude spectrum peaks of $\mathbf{W}^{\mathrm{re}}$ and $\mathbf{W}^{\mathrm{im}}$ are modified in a very similar manner at the end of the training.

The learned filters with the center frequencies up to 800 Hz with the sampling rate 8 kHz and their magnitude spectra have been visualized in Figure 3. The neural network training process seemingly do not result with a shift in the center frequencies of the filter. On the other hand, it should be noted that in addition to the peak at the center frequency, the

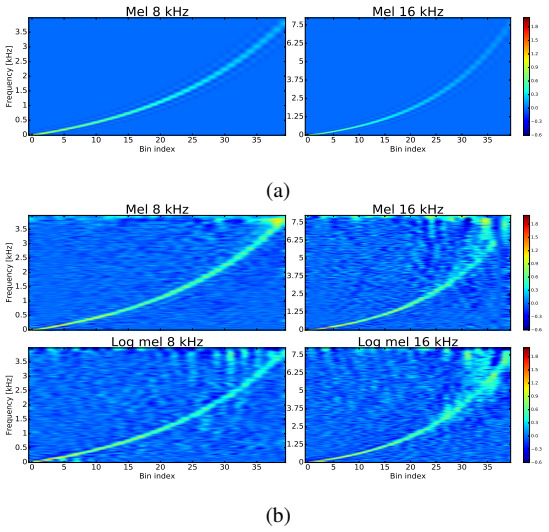


Fig. 4. (a): Initial, and (b): learned mel filterbank responses.

magnitude spectrum of each filter consists of other components with smaller amplitude values spread over the frequency range, which reflects that the pure sinusoid property of the filters are lost.

For the experiments where the mel layer is utilized, the learned mel filterbank responses are visualized in Figure 4. One common point among the responses is that the filterbank parameters covering lower frequency range have been emphasized. The learned filterbank response that is the most resembling its initial response belongs to mel layer with 8 kHz sampling rate, which also performs the best among these four experiments with 61% F1 score, as presented in Table II. For the response of both mel and log mel layers with sampling rate 16 kHz, the parameters covering higher frequency range have been emphasized, and the filter bandwidths for higher frequencies have been increased. However this does not result with an improved performance, as these experiments provide 60.6% and 59.9% F1 score, respectively.

## IV. Conclusion

In this work, we propose to conduct end-to-end polyphonic SED using learned time-frequency representations as input to a CRNN classifier. The classifier is fed by a neural network layer block, whose parameters are initialized to extract common time-frequency representation methods over raw audio signals. These parameters are then updated through the training process for the given SED task. The performance of this method is slightly lower than directly using common time-frequency representations as input. During the network training, regardless of the input sampling rate and the neural network configuration, the magnitude response of the feature extraction block parameters have been significantly altered for the lower frequencies (below 4 kHz), and stayed mostly the same for higher frequencies.

## References


[1] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Reliable detection of audio events in highly noisy environments," *Pattern Recognition Letters*, vol. 65, pp. 22–28, 2015.

[2] J. P. Bello, C. Mydlarz, and J. Salamon, "Sound analysis in smart cities," in *Computational Analysis of Sound Scenes and Events*. Springer, 2018, pp. 373–397.

[3] Y. Wang, L. Neves, and F. Metze, "Audio-based multimedia event detection using deep recurrent neural networks," in *2016 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2742–2746.

[4] S. Krstulović, "Audio event recognition in the smart home," in *Computational Analysis of Sound Scenes and Events*. Springer, 2018, pp. 335–371.

[5] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 379–393, 2018.

[6] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017.

[7] H. Lim, J. Park, and Y. Han, "Rare sound event detection using 1D convolutional recurrent neural networks," DCASE2017 Challenge, Tech. Rep., September 2017.

[8] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[9] S. Adavanne, G. Parascandolo, P. Pertilä, T. Heittola, and T. Virtanen, "Sound event detection in multichannel audio using spatial and harmonic features," DCASE2016 Challenge, Tech. Rep., September 2016.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform CLDNNs," in *Proc. Interspeech*, 2015.

[12] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6964–6968.

[13] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5200–5204.

[14] L. Hertel, H. Phan, and A. Mertins, "Comparing time and frequency domain for audio event recognition using deep learning," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 3407–3411.

[15] Y. Hou and S. Li, "Sound event detection in real life audio using multimodel system," DCASE2017 Challenge, Tech. Rep., September 2017.

[16] E. Cakir, E. C. Ozan, and T. Virtanen, "Filterbank learning for deep neural network based polyphonic sound event detection," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 3399–3406.

[17] B. McFee, M. McVicar, C. Raffel, D. Liang, O. Nieto, E. Battenberg, J. Moore, D. Ellis, R. Yamamoto, R. Bittner, D. Repetto, P. Viktorin, J. F. Santos, and A. Holovaty, "librosa: 0.4.1," Oct. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.32193

[18] D. O'shaughnessy, *Speech communication: human and machine*. Universities press, 1987.

[19] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv:1412.6980 [cs.LG]*, 2014.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.
[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," in *Journal of Machine Learning Research (JMLR)*, 2014.
[23] F. Chollet, "Keras," github.com/fchollet/keras, 2015.
[24] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.