UNIVERSITY
OF OSLO

Master's thesis

Raw Audio End-to-End Deep Learning Architectures for Sound Event Detection

Arvid Falch

Music, Communication and Technology
30 ECTS study points

Department of Musicology
Faculty of Humanities

Spring 2023

Arvid Falch

Raw Audio End-to-End Deep Learning Architectures for Sound Event Detection

Abstract

This thesis proposes deep learning architectures for sound event detection that aims to work fully end-to-end, working with raw audio as input, which can be directly compared to models using fixed graphical time-frequency representations as input. The primary objective is to assess the effectiveness of employing raw audio input in comparison to the conventional fixed graphical time-frequency representations. To achieve this, pairs of similar models based on convolutional recurrent neural networks commonly utilized in sound event detection, are trained using either raw audio or fixed graphical time-frequency representations to enable a comprehensive comparison.

The findings reveal that the proposed deep learning architectures, operating on raw audio input, can achieve comparable performance to models based on fixed graphical time-frequency representations in sound event detection. Moreover, in specific applications where high temporal resolution is of importance, the architectures utilizing raw audio input showcase superior performance when compared to their fixed graphical counterparts. This finding highlights the potential of raw audio end-to-end deep learning architectures in capturing fine-grained temporal information critical for accurate sound event detection.

Contents

Preface

0.1 Acknowledgements

This thesis would never have been possible without the tuition, guidance and inspiring nature of the Music, Communication and Technology Master's program at the University of Oslo. In particular I would like to thank the head of the program and my supervisor Stefano Fasciani for all the knowledge, patience and well worded critique over the last two years.

I would like to thank my employer Squarehead and the research and development team led by Ines Hafizovic for the inspiring discussions and work in acoustic detection that led to my interest in the field of sound event detection.

In a world where GPU computational resources are getting more expensive, I am forever grateful to the Norwegian Ministry of Education and Research and our educational system for providing free GPU computational resources for researchers. And thank you to Riccardo Simionato from the Department of Musicology for providing additional GPU resources for this thesis.

I would also like to thank my fellow students at the Music, Communication and Technology Master's program for pushing each other towards excellence, and for all the fun we managed to have while struggling through our projects.

Thank you to mum, dad, my brother and sister for all the support.

And finally, undertaking a Master program full time as an adult with two kids would never have been possible without the endless love and support of my better half, Cecilie. Thank you for everything.

Sanna and Leander, this one's for you.

Kolsås 11.5 2023
Arvid Falch

Chapter 1

Introduction

In the title of his paper (1993), William W. Gaver asks the question "How do we hear in the world?" and follows up with:

"Everyday listening is the experience of hearing events in the world rather than sounds per se."

As humans we are constantly surrounded by accoustic stimuli which contain information about our surroundings, and help us navigate our daily lives. We are able to distinguish simultaneous sounds, and quickly discern which event the sound originated from, as well as from which direction and how far away the event occurred. When sound waves caused by a physical event propagate through the air and hit our ears, they contain a wealth of information.

Unfolding over time, the amplitude changes of the waves constitute frequencies and amplitude envelopes which the brain can understand as meaningful information about our surroundings. As Gaver points out, in *everyday listening* humans automatically interprets the sounds as events in the world rather than consciously considering the acoustic properties of the sound itself. He contrasts this process to *musical listening*, where we are more concerned with the actual frequencies, timbre, and amplitude of the sound rather than the events causing them.

Finally Gaver develops his main question into two fundamental sub-questions. "What do we hear", and "How do we hear it?".

1.1 Machine Listening

Machine listening is a broad and loose term describing algorithms which analyse digital audio signals and retrieve information from the signal. It can be described as an attempt to replicate parts or all of the processes of our human auditory perception with a computer. Early attempts were limited and could be used only in very specific audio-related applications, but as the algorithms have become more sophisticated, we can start to think of them in the context of Gaver's questions. How do computers "hear" in the world?

In early attempts of *machine listening*, the temporal nature of sound quickly proved to be challenging. Sound, except for the amplitude at any given moment, must always be considered as a sequence of information. The computational cost and complexity of analysing a sequence of digital audio over time was high, mainly because of how dense we have to sample the audio signal to retrieve all of the information in a digital format. Creating algorithms to analyse a dense sequential digital signal was complex and ineffective and quickly led the way towards the use of audio feature extraction techniques.

Feature extraction is the process of identifying and selecting the most relevant and informative characteristics or features of raw data in order to transform it into a new representation that is more suitable for analysis. An interesting development driven by feature extraction occurred in various fields within *machine listening* that suddenly gave a very surprising answer to Gaver's reformulated question on how computers "hear".

1.2 Machine Learning

The rise of Machine Learning (ML) during the 2000's and 2010's resulted in major accomplishments within almost every field of natural sciences, and *machine listening* was just as influenced by this new paradigm as every other field of science. The development of Artificial Neural Networks (ANNs) and deep learning models capable of reaching human performance levels in various applications have had (and still have) an enormous impact on society and the sciences. Inspired by the success of *computer vision* research using deep learning models, it quickly became clear the models used for analysing images could be used for audio-related applications as well. By feeding the models graphical time-frequency representations of the audio signal, the models became sophisticated enough for computers to classify a wide range of sounds, and in specific applications to an extent answer Gaver's first sub-question, *what do we hear*. The answer (although simplified) to the next sub-question *how do we hear* is of key interest to the subject of this thesis. The computers do not "hear" sound, they "see" it.

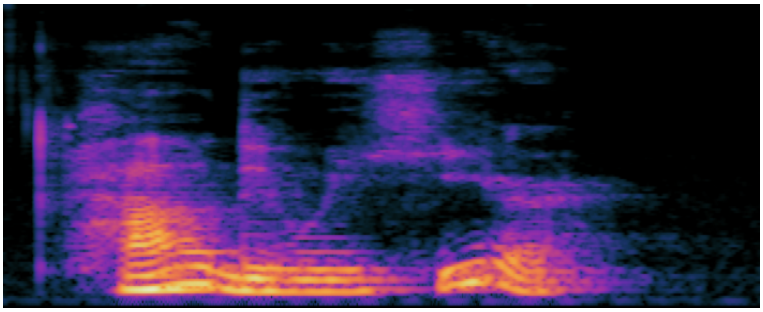


Figure 1.1: A graphical time-frequency representation of the author's voice asking "How do we hear?"

It might be considered speculative to use perceptual terms such as hearing and seeing when describing computer processing, and therefore we should be careful when using such terminology in computer sciences in general. However, in our attempts to better understand the processes of ML algorithms, it is crucial that we understand all the constituents and consequences of the data that we use to train the algorithms. Is the graphical time-frequency representation of sound sufficient when we move towards developing more general purpose machine listening systems?

1.3 Thesis Proposal

Machine listening makes up a very broad field of research, both within the "everyday listening" and the "musical listening" categories of Gaver. The latter spawned various Music Information Retrieval (MIR) applications and have drastically changed our abilities to process and manipulate audio in the digital format. Within the former, a wide range of acoustic information analysis applications have been developed. These

will be covered in chapter 2. This thesis focuses on the specific field of polyphonic Sound Event Detection (SED), which aims to answer Gaver's first sub-question, thereby being able to get information about events in an environment purely through acoustic properties.

To date, using raw audio as input for deep learning models doing SED is not common approach. Until recently, the computational cost of deep learning models using raw audio as the input was prohibitive, while computer vision models showed such promising results and lower computational complexity also in audio-related applications. However, the increase in performance of these models in SED has stagnated in recent years (Chan & Chin, 2020), and state of the art SED systems are still far behind the performance levels achieved in computer vision. This thesis proposes the use of alternative deep learning model architectures based on raw audio input for SED.

For a final time allowing ourselves to use human terms to describe computer processes, inspired by Gaver we ask the question *does it matter whether a computer "hears" or "sees" sound?*

1.3.1 Scope and limitations

The aim of this thesis is to experiment with novel deep architectures capable of learning a meaningful representation directly from raw audio as input and compare them to models using fixed time-frequency representation log mel-spectrograms as input, the latter which is by far the most popular choice of input to SED systems.

To achieve this goal within the time frame of this Master thesis, the models must be limited in size and complexity, and the dataset size manageable. The aim is therefore not to beat any current state-of-the-art performance within SED, but to conduct an experiment where comparable models using different input can be analysed to answer the research questions. Compared to modern SED systems the models will be notably smaller and less complex, which is also true for the amount of data used in training.

1.3.2 Research Questions

This thesis proposes the following research questions within the field of SED.
RQ 1: How does the performance of a deep learning model that works strictly on raw audio as input compare to the performance of a baseline SED system using fixed graphical time-frequency representation of audio as input?
RQ 2: Does working with raw audio as input to a deep learning model provide any new insights into the challenges of SED?
RQ 3: Are there limitations in 2D Convolutional neural networks used in SED which can impact future general purpose machine listening systems?

1.4 Motivation

In a time where highly advanced ML systems such as the natural language processing model ChatGPT and the image generating Stable Diffusion models have caught the public's attention, artificial intelligence is quite likely the greatest scientific breakthrough so far in the 21st century. The public's reaction ranges from utter amazement to pure fear of being displaced by robots, and almost every level of human existence will to some extent be impacted by artificial intelligence in the future. Understanding this new tool humanity has at its disposal is crucial in order to develop applications that can

be beneficial for us. Speech recognition ML algorithms like openAI's Whisper (Radford et al., 2022) have reached an impressive level of performance, but there are still many problems to solve in the use of ML in audio analysis applications. Sound is an important part of the human experience, and therefore ML algorithms capable of understanding sound on a human level can provide huge benefits in a variety of applications.

1.5 Contributions

The main contributions of this thesis is:

- Proposal of a deep learning architecture for sound event detection which can learn a meaningful representation directly from raw audio, and are directly comparable to graphical time-frequency representations of audio as input.
- Questioning the traditional choice of graphical time-frequency representations as input in SED.
- Proposing fully 1D Convolutional Neural Networks for SED.

Chapter 2

Background

2.1 Sound Event Detection

Human auditory perception enables us to process and understand highly complex *auditory scenes*. Within these scenes we can segregate separate sound sources and direct our attention towards them, and we can infer properties of our environment through our auditory perception. We can understand complex sequences of sound to be a single sound event instance, for example when we hear a bird singing or a person walking on gravel. Creating a system capable of reproducing some of these capabilities, we are, given an audio file (both real time monitoring and off-line processing) simply trying to discern *what* is happening and *when* it is happening.

Polyphonic Sound Event Detection (SED) systems aim to classify one or more sound events in a sound scene, and at the same time predict the start and ending time of each sound event. The latter is often referred to as segmentation. Therefore, SED comprehends two tasks: classification and segmentation. Using multichannel audio we can expand on SED systems and localise the sound event in space as well, which is referred to as Sound Event Detection and Localization (SEDL). Both SED and SEDL differ from speech recognition and music information retrieval systems as it is not concerned with the aesthetic qualities of sound or the linguistic information, but purely being concerned with the identification of sound sources.

SED is important in various real-world applications such as healthcare, surveillance, smart homes, wildlife monitoring and video indexing. In the broader field of computer based audio analysis SED is closely related with research in audio scene classification, audio anomaly detection and bioacoustic event detection.

2.1.1 Specific issues in sound event detection

There are various specific issues which are important for our understanding of SED.

- Our environment is filled with sound, and the sound waves coming from different sound events mix as they propagate towards our sensors, be it a microphone or our ears. This makes the task of SED polyphonic, as multiple sound events can happen simultaneously in time and the system must be able to discretise between them. On many occasions where the distance between a sound source and the sensor is significant, the sound pressure level of the sound event can be lower than other sounds occurring closer to the sensor, increasing the difficulty of polyphonic SED.

- The amount of possible sounds that can exist in a real environment is unlimited, which differs from speech recognition which can focus on a set number of phonemes. In SED, the possible number of different sound classes is therefore infinite and each system must be constructed to solve a specific task and decide upon a predefined set of target sound classes. At the same time the system must be prepared for the unlimited amount of possible non-target sound events which it will be exposed to, which leads to the problem of how to accurately represent this vast category of "everything else". A SED system's ability to not trigger a detection of a non-target sound event (i.e. low False Positive (FP) rate) can be important. Recent findings have shown that the performance of state-of-the-art SED systems decreases significantly when tested only with non-target sound events (Ronchini & Serizel, 2022), implying that even if they work well for very specific application, they are limited when seen in the context of general purpose machine listening systems.

- ML SED systems require huge datasets of audio to be annotated by humans, which are time consuming to assemble. The onset and offset times of the sound events must be precisely labelled, and the human annotator has to be sure that a sound event belongs to the labelled class. This requires audio that is either recorded in controlled environments, or audio accompanying video files where the source of the sound event can be verified. For a human to label a sound event purely with the audio information can be notoriously hard. As an example of how we easily we can mix up sound sources, the sound engineer Tasos Frantzolas in (Frantzolas, 2009) shows a film clip of a scene depicting a rainy scene, where the audio which sounds like rain is actually a recording of bacon frying in a pan.

2.2 Machine Learning

Machine Learning (ML) is a field within Artificial Intelligence (AI). AI is a general field encompassing both ML and deep learning, and can be described as *the effort to automate intellectual tasks normally performed by humans* (Chollet, 2021). ML is a subfield within AI, and can be defined as "the study of algorithms that allow computer programs to automatically improve through experience" (Mitchell et al., 2007). ML algorithms are mathematical models which through analysis of data (experience) are capable of doing inference based on patterns in previously unseen data.

In traditional programming, a program is made up of a set of explicit rules provided by the programmer, enabling the program to provide an answer to a given input. In ML, the system is trained based on input data to come up with the best set of rules in order to successfully transform the input data to the desired output. In order to achieve this, the ML system has to learn a useful *representation* of the input data that brings us closer to the desired output.

During training, the ML system searches for all possible representations given the algorithm and the input data, which we can think of as the system's *hypothesis space*. A learning algorithm is then applied in order to determine the distance between the model output and the desired output. Given the distance, a feedback signal is sent through the system adjusting the algorithm responsible for learning the best representation.

ML algorithms can be divided into three broad categories: unsupervised learning, supervised learning and reinforcement learning algorithms. These categories differ in how we define the expected or desired output of the system.

In unsupervised learning, the desired output is not given explicitly, and the system is trained to find patterns, structure, and relationships within the data based on statistical techniques. Common techniques used are clustering, dimensionality reduction, and anomaly detection.

In supervised learning, we provide a *ground truth* explicitly stating the desired output during training. In classification tasks this is referred to as labelled data, meaning that the model is trained on a pair of input data and labels as input and output respectively. In regression tasks the label is replaced by a target value. Audio classification tasks which this thesis focuses on are mainly based on supervised ML algorithms.

Reinforcement learning are ML algorithms based on reward/punishment input from humans or a fixed environment, enabling the algorithm to learn the desired behaviour.

As ML algorithms became more complex and our computational abilities increased, the term deep learning became popular to describe a certain kind of ML systems.

Deep learning can be defined as stacking multiple ML algorithms (usually referred to as *layers*). Each layer enables the system to learn an increasingly meaningful representation of the data, thereby expanding the system's hypothesis space. Deep learning ML systems are traditionally referred to as *models*, and what layers and number of parameters they consist of are referred to as the model architecture.

2.2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are the broad term describing all models based on layers consisting of neurons. The most basic implementation of this is called a *feedforward* or *dense* layer. Each *feedforward* layer consists of one or more neurons. A neuron is a function made up of weights, bias and an activation function, and is connected to all the neurons in the layer before (which could be the input layer or another layer of neurons). Each connection is given a weight, and the output of a prior neuron is then multiplied by the weight of the connection, which has the bias of the receiving neuron added to its value. The result is run through the activation function, which decides the output of the neuron. The output of the neuron is connected to all the neurons in the next layer, and the process is repeated. The input layer is representing our input data, and each neuron in the input data represents one scalar value of our input data. Input data neurons do not have weights and activation functions. Middle layers in ANNs are called *hidden layers*, and each *hidden layer* can consist of any number of neurons. Finally the output layer must correspond to our desired output. In a binary classification task, the output layer would consist of one neuron outputting either 0 or 1. In a regression task, we can remove the activation function for the last layer resulting in the neuron outputting a scalar value, or in some cases we can have multiple neurons outputting a vector. More complicated tasks such as *multiclass* and *multilabel* classification problems are described in detail in section 2.3.

2.2.2 Convolutional Neural Networks

Feedforward layers are limited to learning global patterns of their input data. The proposal of Convolutional Neural Networks (CNNs) was a huge step forward. Convolutional layers are able to learn local patterns of input data, which greatly improves the network's ability to learn a meaningful representation of complex data. CNNs[1] were

[1]The term CNN customary refers to models using 2D convolution, as in computer vision models. To specify the dimensionality of the convolutional layers one can use 1D CNNs to specify 1 dimensional convolution, which will be covered in section 3.2.

proposed as early as in 1989 by LeCun et al. (1989) who trained a CNN to classify handwritten digits. However it's popularity never took hold in the field of computer vision, until the CNN AlexNet in 2012 clearly outperformed traditional ANNs in various image classification challenges (Krizhevsky et al., 2017).

Instead of fully connected neurons, a convolutional layer is made up of filters, each consisting of multiple weights whose number is given by the kernel size. The shape of the filters is a vector for 1D convolution, and a matrix for 2D convolution.

During the convolution operation, the filter is slid across the input data. Each position of the filter is the filter's current *local receptive field*. The dot product between the filter weights and the input values within the local receptive field is computed to produce a single output value. By connecting each neuron in the layer to a different local receptive field, the network is able to learn hierarchical representations of the input data, with lower [2] layers detecting simple features and higher layers detecting more complex features that are composed of combinations of lower-level features.

The size of the local receptive field is determined by the size of the filter and the stride of the convolution operation, which specifies the distance between adjacent filter positions. A larger filter size and smaller stride will result in a larger local receptive field, while a smaller filter size and larger stride will result in a smaller local receptive field.

A key feature of the CNNs is that the patterns learned (ie the representation of the data) by each filter are translation-invariant. Due to the local receptive field, a pattern learned in the lower left corner of an image could be applied to input data where the pattern occurs in the top left corner, or if it is flipped. Whereas a *feedforward* model would have to learn a pattern from scratch if it appeared in a new location. For computer vision, this is vital, as our visual world is translation-invariant.

Another key feature of CNNs is how the filters emphasise that all data points that fall within the local receptive field have some significant connection to each other. In computer vision this is again true, as pixels that lie close to each other either belong to the same object or create a border between objects, which is just as important.

Finally, both *Feedforward* ANNs and CNNs have no memory of prior events, each input is processed independently. This is a severe limitation when working on sequential data like time series. In order to work with sequential data such networks must condense the whole sequence into a single data point, which can prove detrimental given a long sequence.

2.2.3 Recursive Neural Networks

For sequential data, where knowledge of former data matters in predicting the current input data, Recursive Neural Networks (RNNs) have been a common technique. In its most simple form, RNNs consist of recursive layers, where the output of a current timestep is fed back into the neurons recursively at the next time step and concatenated or multiplied with the input for the next timestep. Note that RNNs iterate over each element of a sequence, and during this process the RNN neurons maintain a state based on all the sequence elements so far in the iterative process. At the start of a new sequence, the RNN neuron re-initialises its state. (In the case of the Simple RNN, this simply means looking purely at the input and not receiving any values from the former step).

[2] deep learning models are described rather counter intuitively from bottom to top, meaning lower layers are the first layers in the model

However, this very simplified RNN (referred to as Simple RNN) is not capable of differentiating between important and less important information, and long-term dependencies prove impossible to learn in practice. This is called the *vanishing gradient problem*, meaning that important information from further back in the sequence will vanish quickly, thereby causing the memory of the RNNs to be too short on complex sequences.

A more complex version is the Long Short-Term Memory (LSTM) layer, proposed in 1997 (Hochreiter & Schmidhuber, 1997). By adding additional weights and activation functions in the neuron, the state of the neuron is given more flexibility in which to emphasise each new recursive input. An additional track of data flow maintaining just the state (called the *cell state*) runs parallel to the usual input and recursive input track (the recursive input track is called the *hidden state*). This new track containing the *cell state* can be envisioned as a conveyor belt. Information from the current sequence *can* be brought onto the conveyor belt (through what is referred to as the *update gate*, just as information already on the belt *can* be dropped at any point in the sequence (the *forget gate*). As this process is also made up by learnable weights, an LSTM cell (since there are multiple operations going on we refer to it as a cell rather than a neuron) can learn what information to keep and what to discard, enabling it to learn long term dependencies of a sequence.

A new, simplified version of the LSTM cell was proposed in (Cho et al., 2014), called the Gated Recurrent Unit (GRU). It consists of less weights than the LSTM as it combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, but has shown to perform equally well on multiple tasks.

LSTM and GRU layers can be set up as bidirectional layers. The original layer is joined by a mirror layer, which operates on the sequence backwards. The output of the two layers are joined either through multiplication or concatenation.

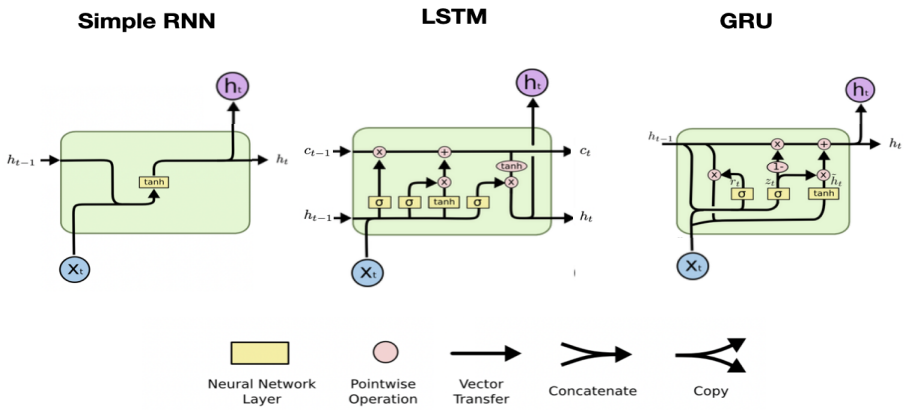


Figure 2.1: Anatomy of RNN cells, illustration from (Olah, 2019) where x is input, h is output and c is cell state, and $_t$ denotes the timestep.

2.2.4 Transformers

Transformers were introduced in 2017 (Vaswani et al., 2017) and focuses on multi-head attention neurons which are capable of storing attention vectors to a great number of learned tokens. It has revolutionised Natural Language Processing (NLP) applications in ML, leading to Large Language Models (LLMs) like GPT. In automatic speech

recognition, openAIs Whisper (Radford et al., 2022) represents the state-of-the-art using a Transformer architecture. However, Transformers are out of the scope of this thesis, so the mechanisms behind them will not be explained here. Many SED models utilise attention based mechanisms similar to those found in Transformers, but the embedding process (known as Tokenization in NLP) is still on a very experimental stage for SED, disqualifying it as a useful tool for answering the research questions of this thesis.

2.3 Sound Event Detection and Machine Learning

2.3.1 Multi-class multi-label classification

SED can be described as a *multi-class multi-label classification* task. There are more than two possible classes (in contrast to Binary Classification problems), and the labels can contain multiple events (correct labels) simultaneously. As explained in section 2.2.1 the output layer of a model (usually a *feedforward* layer) must correspond to the desired output. In a binary classification task, the final layer needs only one neuron which outputs either 0 or 1. For a multi-class problem, the final layer must have a number of neurons equal to the number of classes, and usually the *softmax* activation function (Equation 2.1) then provides the probability of each class summed to 1. In a multi-label problem, the *sigmoid* activation function (Equation 2.2) is used. It provides individual probabilities for each class, enabling the prediction of multiple classes at each instance of the input data.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{2.1}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

2.3.2 Audio Analysis and ML

Automatic Speech Recognition (ASR) makes up by far the largest body of research in audio analysis research, and many audio analysis breakthroughs in ML have had it's origin in ASR. Presented with audio of a voice speaking, the goal of ASR is to transform this data into written text. In 2012, researchers from four industrial leading companies (Hinton et al., 2012) outlined the history of ML approaches for ASR, which is a good starting point for understanding the background of SED and ML. The paper summarises the breakthroughs in deep learning on the task of phonetic classification for ASR before the introduction of CNNs, where RNNs, Support Vector Machines (SVMs) and Gaussian Mixture models were prevalent. With the success of CNNs in computer vision from 2012, audio analysis research adapted the principles of computer vision models on graphical time-frequency representations of sound and quickly reported huge gains in performance.

2.3.3 Convolutional recursive neural networks in SED

ANNs and more significantly CNNs can be applied to audio classification tasks with great success, but as they lack the ability to process sufficiently long sequences of data, once we add the task of segmenting events, ie provide us with the onset and offset of event detections, they are no longer sufficient. For this task we need models which are capable of learning long term dependencies in sequential data.

Pure RNN networks have been attempted both in SED (Parascandolo et al., 2016) and Automatic Speech Recognition (ASR), but were less capable of learning a high level

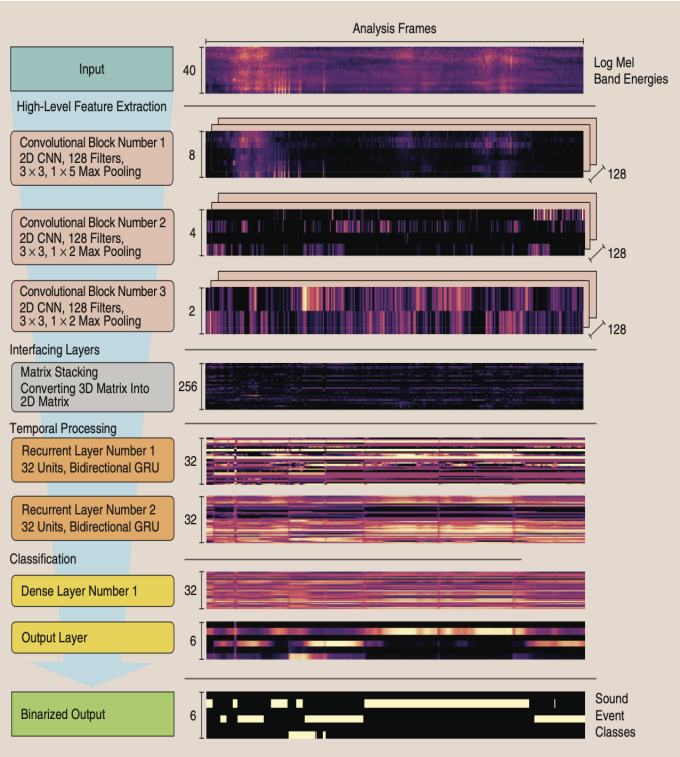


Figure 2.2: Basic implementation of the CRNN suggested by (Çakir et al., 2017). Illustration from (Mesaros et al., 2021).

representation of data, which forced the use of more drastic feature extraction methods which limited the models capability of generalising (Çakir et al., 2017).

Based on results from research in ASR (Sainath et al., 2015) and music genre classification (Choi et al., 2017), Cakir et al. proposed a model in 2017 consisting of a block of CNN layers followed by RNN layers, calling it a Convolutional Recurrent Neural Network (CRNN) (Çakir et al., 2017). Drawing on the power of the CNN to find meaningful high level representations of the data before it is fed sequentially to the RNN which is able to output predictions for each timestep of the sequence, and keeping a memory of former events, the CRNN vastly outperformed earlier attempts at SED using pure RNNs or other ML algorithms.

The CRNN architecture (Figure 2.2) proposed a change in the way normal CNNs are applied, as the CNN layers did dimensionality reduction only over the frequency axis, leaving the size of the temporal axis unchanged. In a CNN processing an image, both the x and the y axis are gradually reduced in size. Since visual data is translation-invariant, the x and y axis represent two homogeneous spatial dimensions and dimensionality reduction can be applied equally to both axes. However, when working with sound in a time-frequency representation, it is more appropriate for the temporal axis to keep the same resolution as the input data, while the frequency axis gradually undergoes a dimensionality reduction as high level feature representations are learned by the filters in the successive layers. This was a major step forward for SED, and most SED systems have used a version of CRNNs since then.

2.3.4 Detection and Classification of Acoustic Scenes and Events Community

The Detection and Classification of Acoustic Scenes and Events (DCASE) Community[3] was established in 2013 by researchers working within SED and various other computational auditory scene and event analysis methods. Providing yearly challenges in a variety of tasks, many of the improvements in SED have been conceived in the context of DCASE. One of its many contributions to SED has been the establishment of openly available datasets both for developing and evaluating SED systems, which enables researchers to compare the performance of their systems to each other, and in many cases removes the burden of annotating large bodies of data for a SED research project.

Sound event detection and separation in domestic environments

Since 2017, the Task 4 of the annual DCASE challenge has focused on SED in domestic environments. The focus of the challenge is to explore how we can use weakly annotated data (which is much more accessible) together with small strongly labelled datasets to train SED systems. In SED, a strongly labelled dataset contains the onset, offset and class label of each sound event, while a weakly labelled dataset is just the presence of a sound event class label in a sound clip, without its temporal position. The strongly labelled dataset originating from this task will be further described in chapter 4, as it is the dataset used to train and evaluate the deep learning models of this thesis.

2.4 Audio representations in Sound Event Detection

Sound sampled in a digital wavefile poses a problem when used as input for any ML system. The original waveform of a sound can be too dense and rich for deep learning models to learn a meaningful representation of the data (Natsiou & O'Leary, 2022). This complexity also increases training time and computational cost. This has led developers of ML audio applications to work with various feature extraction methods in order to condense the raw audio wavefile into a more suitable data format. This section will explain the different representations of sound used in ML applications, with a specific focus on graphical time-frequency representations used in SED systems.

2.4.1 Raw audio

The term raw audio is used to refer to audio encoded using Pulse Code Modulation (PCM). A continuous waveform is digitally sampled in time and amplitude. The result is a sequence of numbers, each number representing the amplitude value sampled at a given time step. The amount of time steps is determined by the sample rate which is being used. The sample rate is chosen according to the Nyquist-Shannon theorem (Shannon, 1949), which states that a sampled signal can only preserve all the information up to half the sample rate used. The sample rate is written in Hertz, such that a digital audio file of 1 second duration sampled at 44.1 kHz will consist of 44100 samples. The resolution of the sample values, which also affects the quality of a digital signal, is given by the bit rate the signal is stored as. 8 bits result in 256 possible discrete values, 16 bits provide 65536 possible values, and 24 bits provide 16.8 million values. CD quality audio is traditionally sampled in 44.1 kHz sample rate with 16 bit. To reduce computational

[3] https://dcase.community/

cost in ML audio applications, it is common to use sample rates ranging from 8 kHz to 24 kHz, while keeping a 16 bit resolution.

2.4.2 Acoustic features

Acoustic features are compressed, descriptive representations of aspects of a sound. Historically in ML, before deep learning and especially CNNs was able to handle more complex data, acoustic features usually covered one aspect such as fundamental frequency, spectral features or energy levels. Such acoustic features can be combined in order to cover more aspects of the sound, but are still too generic and often not successful in describing specific content of a sound (Virtanen et al., 2018). A more complex acoustic feature frequently used in ASR and audio analysis even to this day is the Mel-Frequency Cepstral Coefficient (MFCC). The MFCC captures the spectral envelope in frequency bins in overlapping segments, preserving more complex information of various aspects of the sound. However, as computational resources grew, more complex acoustic features emerged as the most popular input in ML audio applications.

2.4.3 Graphical time-frequency representations

The majority of SED systems have adopted Machine Learning (ML) techniques derived from the field of computer vision, even though visual data and audio data have significant differences. To date, these SED systems regard audio through its graphical time-frequency representation.

Spectrograms

The spectrogram is a time-frequency graphical representation of sound. By applying the Short Time Fourier Transform (STFT) to overlapping segments of the waveform and discarding the complex numbers, the magnitude of the Fast Fourier Transform (FFT) is calculated for each segment over a number of frequency bins. Short-Time Fourier Transform (STFT) of the input signal x:

$$STFT[n,k] = \left| \sum_{m=0}^{W-1} w[m]x[nm+k]e^{-j2\pi mk/W} \right|, \tag{2.3}$$

where n is the frame index, k is the frequency bin index, $w[m]$ is the mth sample of the window function of length W, and $j = \sqrt{-1}$.

This results in a time-frequency representation. The FFT size denotes the size in samples of each segment. The frequency resolution of each frequency bin is given by

$$N\_bins = FFTsize/2, \tag{2.4}$$

$$FR = Fmax/N\_bins, \tag{2.5}$$

where FR is the frequency resolution of the spectrogram and $Fmax$ is the sample rate divided by 2. There is an inherent trade-off between frequency and time resolution when employing the Short-Time Fourier Transform (STFT). As the FFT size decreases, the time resolution improves, enabling a finer depiction of temporal changes. However, this comes at the cost of diminished frequency resolution.

Logarithmically scaled mel-spectrograms

In the logarithmically scaled mel-spectrogram, referred to as a *log mel-spectrogram*, the linear frequency scale of the spectrogram is converted to a logarithmic scale, and then the logarithmic frequency axis is mapped to the mel scale. The mel scale is a non-linear scale that approximates the way humans perceive pitch (Pedersen, 1965). The relation between the mel scale (f) and the Hertz scale f can be expressed as in (Fant, 1968):

$$Mel(f) = \frac{1000}{log2} log(1 + \frac{f}{1000}) \tag{2.6}$$

It has become hugely popular in ML audio applications as the log mel-spectrograms condenses the frequency content in the highest frequency register resulting in an emphasis on important frequency regions corresponding to the human perception in the graphical time-frequency representation (see Figure 2.3).

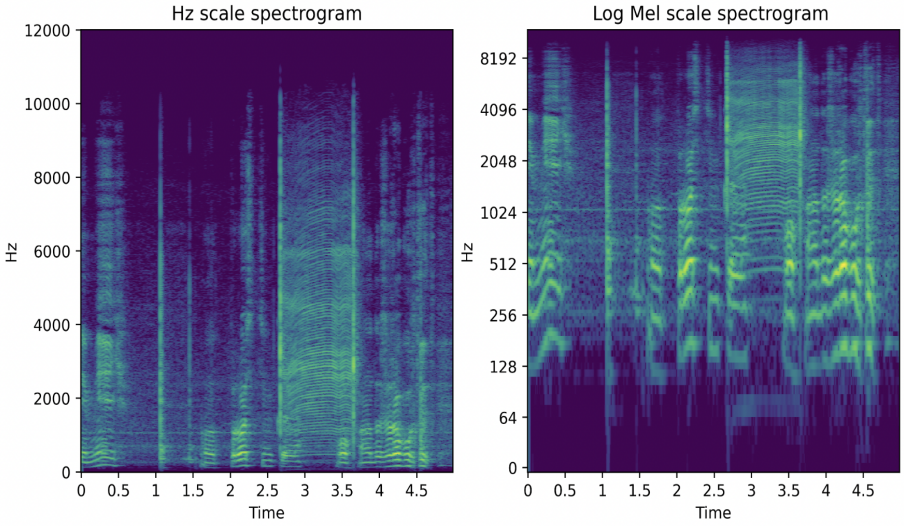


Figure 2.3: Normal Hz scale spectrogram vs log mel-spectrogram on a segment of audio containing speech. Notice how important frequency regions (lower and middle register) are emphasised in the log mel-spectrogram.

Wavelet transform

Another time-frequency graphical representation of sound is the scalogram, which are obtained through the continuous wavelet transform (CWT). The CWT is based on a family of functions called wavelets, which are essentially small wave-like oscillations that are localised in time and frequency. Scalograms computed with the CWT allows for a more flexible frequency resolution, where the higher frequencies are given a higher resolution than the lower frequencies (Chen et al., 2018), (Copiaco et al., 2021).

The CWT of a signal x(t) is defined as the convolution of the signal with a scaled and shifted version of a wavelet function:

$$CWT_{\psi}(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t)\psi^*(\frac{t-b}{a})dt, \tag{2.7}$$

where ψ^* is the complex conjugate of the wavelet function ψ, a and b are scaling and translation parameters, respectively, that control the size and position of the wavelet.

The CWT is computed for a range of scales and positions to obtain a time-frequency representation of the signal.

2.4.4 Representation learning

Deep learning models provides us with a framework of layers which learns successively more high-level feature representations of the data (LeCun et al., 2015), usually resulting in some form of dimensionality reduction as the data is processed deeper into the model. If a model is trained on a large enough dataset, it can be adjusted and tuned to generalise well on other tasks than it was trained for. This is called transfer learning, were an existing model trained on a large dataset is further trained (fine tuned) on a new task, which can be made up by a much smaller dataset. If we think of a CNN image recognition model trained to classify hundreds of different classes, we can substitute the last fully connected layer with a new random initialised layer adhering to the new task (ie number of neurons = number of classes for the new task), and train it on a small dataset consisting of examples of the new classes. Using openly available large image recognition models like ResNet and AlexNet, adjusting the model architecture, and then fine tune the model to classify sound sources based on log mel-spectrograms, is an example of transfer learning which saves developers a lot of time.

More conceptually, a feature representation obtained from any layer inside a specific deep learning model can be used as input to other deep learning models designed to solve different tasks in the same domain. This obtained representation is referred to as a learned feature representation or a latent space representation.

Encoder-Decoder approach

As a deep learning model attempts to learn a meaningful representation of the input data, a certain type of deep learning model can be used to create a latent space representation of the input data, which can then be fed to another model as the input data. This is done by creating an *encoder-decoder model*. The encoder part of the model is responsible for creating a latent space representation of the input signal in any desired shape and dimensionality, while the decoder is trained to reconstruct the original signal based purely on the latent space representation. If successfully trained, we can assert that the latent space representation contains enough information about the input signal since the decoder is capable of recreating the signal based on it.

The encoder-decoder does not require any labelled data as it is trained with the same signal as input and output.

After the model is trained, the decoder part can be removed and the encoder can now be used as the first part of any model architecture, responsible for taking the input signal and transforming it to a latent space representation which is more suitable for the deep learning model and the task it is trained to perform. The weights in the layers responsible for creating the latent space representation can either be frozen or trainable.

However, any deep learning model architecture can be tasked to produce a latent space representation of the input signal. Knowing that a certain architecture works in an encoder-decoder format enables us to integrate the encoder architecture in any deep learning model and train it in end-to-end fashion, ie not pre-train it using the input as ouput.

In the case of the latent space representation learned in an end-to-end fashion in a SED system, it is simpler to refer to it as a learned audio representation. Examples of a

log mel-spectrogram and the corresponding learned audio representation from raw audio can be seen in Figure 2.4.

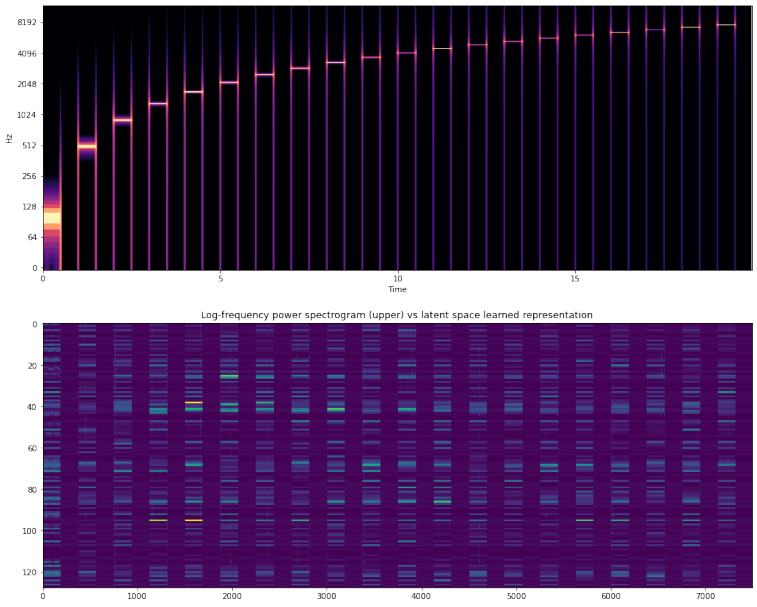


Figure 2.4: Example of a log mel-spectrogram and the corresponding learned audio representation from raw audio

2.5 Performance Metrics for Sound Event Detection

Evaluating the performance of a polyphonic SED system can be challenging. The model outputs a probability for the presence of a sound event for each class, at very short intervals (2 to 200 ms). This interval is called *the analysis frame*. As the output of each neuron in the last layer is a probability value between 0 and 1, a threshold value has to be defined (usually 0.5), and any output above the threshold value is considered a detection. The system's detection can then be compared to the ground truth, which is the annotations of the evaluation data set.

2.5.1 Post processing

Comparing the results of the system's output and the ground truth at short temporal resolutions can be penalised by small timing errors and human annotation errors. Therefore, the system output will usually be post processed before it is evaluated. The post processing could be mean, max or median value over a set number of analysis frames for each class, resulting in the system output's final temporal resolution. For example, some of the models used in this thesis generates a prediction every 2.6 ms, but by applying a post processing step of taking the mean value over 150 ms we can smooth the system's output (Figure 2.5 and Figure 2.6). Lin et al (2019) proposed to use individual time resolutions of smoothing for each class, applying longer temporal smoothing windows over audio classes which mainly consists of long sound events, and shorter windows for short sound events. Post processing can also involve using individual detection thresholds for each class, based on evaluation analysis.

After the post processing step, we use different proposed strategies to evaluate the predictions as described in the next subsections.

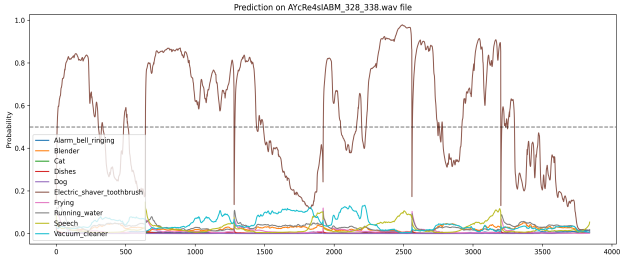


Figure 2.5: Example model predictions before post processing

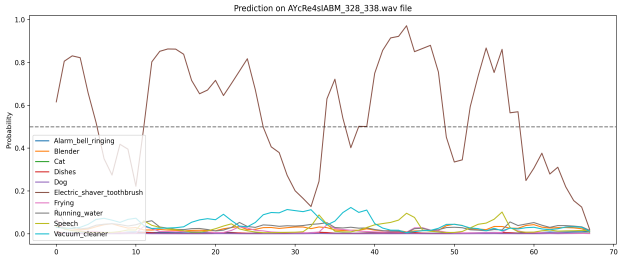


Figure 2.6: Example model predictions after mean post processing over 150 ms

2.5.2 Analysis frame based evaluation

By comparing each analysis frame with the ground truth before applying any post-processing, the frame level performance metrics can be obtained. It was the main evaluation strategy for most SED systems until segment based and event based evaluations became more popular from 2017.

2.5.3 Segment based evaluation

In segment based evaluation, the system's output and the ground truth are considered on a fixed temporal grid with a coarser temporal resolution than the system output, where both the system's output and the ground truth has been temporarily quantized. The result of the comparison is a matrix of *n_classes* and *n_segments* where each instance is either a True Positive (TP), True Negative (TN), False Positive (FP) or False Negative (FN). In the DCASE Task 4 evaluations the segment length is traditionally set to 1 second. Segment based evaluation does not provide insight on the temporal precision of a SED system at higher resolutions than the segment length, and in applications where the system needs to detect sound events quickly and precisely, it would not be the right evaluation strategy.

2.5.4 Event based evaluation

Event based evaluation compares the system's output and the evaluation data annotation as lists of sound events with onset and offset times. The latter is called the ground truth, and each event in the ground truth is referred to as the reference event. The performance is determined by checking how well the detected sound event corresponds to the reference event. As the onset and offset is given in a very high temporal resolution, a collar of a fixed length (typically 200 ms) is applied to the reference event onset. As long as the

onset of the detected event is within this collar it is accepted as a correct onset. For the offset a tolerance value corresponding to a percentage of the reference event length is used (typically 20% - 50%).

- TP's are detected events with a temporal position which overlap an event in the ground truth reference event list within the set collar and tolerance.
- FP's are detected events which have no correspondence to an event of the same label in the ground truth, or when the detected event does not overlap within the set collar and tolerance.
- FN's are events in the ground truth which have no correspondence to events of the same label in the detected output.
- There are no meaningful TN's in event based evaluation, as the list of reference events do not contain non-target sound events.

Event based evaluation can penalise detections of multiple sound events if the ground truth only contains one reference event, as described below in the section PSDS.

2.5.5 F1 score

The main metric used by SED systems for both segment based and event based evaluation is the F1-score. By counting the correct and erroneous predictions as TP, FP, TN and FN we can calculate the F1 score (F1) based on Precision (P) and Recall (R).

$$P = \frac{TP}{TP + FP} \tag{2.8}$$

$$R = \frac{TP}{TP + FN} \tag{2.9}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.10}$$

The class-wise F1 score is calculated individually for each class, and the Macro-Average F1 Score is calculated as the average of the individual classes.

2.5.6 Error Rate

The Error Score (ER) is also widely used. The ER consists of the Substitution Error (S), Insertions (I) and Deletions (D). S is all occurrences of both FPs and FNs, while the remaining FPs unaccounted for in S are the remaining insertions I, and the FNs unaccounted for by S make up the deletions D. The Error Rate is then given by

$$ER = \frac{S + D + I}{N} \tag{2.11}$$

where N is the number of reference segments/events.

In the case of a ground truth event of a cat meowing, where the system detects a dog barking, this is counted as S. It is both the FP of a dog bark detected, and a FN of a cat meowing being undetected. In the case of a system detecting a dog when there is no target event in the ground truth, this is counted among the I. When the system does not detect any sound event while the ground truth contains a sound event this would be counted in D.

2.5.7 Accuracy

Accuracy is only calculated as a segment based metric.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.12}$$

2.5.8 Polyphonic Sound Detection Score

The recently proposed Polyphonic Sound Detection Score (PSDS) (Bilen et al., 2020) is a more flexible event based evaluation metric. The PSDS framework allows to evaluate different scenarios for a SED system, and takes into account the following specific problems of SED evaluations:

- **Operating point dependency:** The decision threshold, i.e. the threshold applied to the system output for deciding whether the system outputs a detection or not, has an impact on the performance metrics. A system evaluated with different decision thresholds might produce different results in regards to the classic metrics of F1 and ER. In other words the problem of detecting sound events and the problem of finding the right operating point for the system is mixed together.
- **Definition of sound events:** Event-based metrics rely on the collar and tolerance as constraints for the onset and offset times of the predicted output compared to the ground truth events. As described above in section 2.5.4, these timings may be subjective to human labellers (and human users of a SED system). If a dog barks three times in quick succession, should it be labelled as one or three sound events? Does a short pause between words signify a new sound event or not? When is the onset time when a car is approaching from far away?
- **Cross-triggers:** Cross-triggers (CTs) are the subset of FPs which match another labelled class. Detecting a dog barking when the ground truth is a cat meowing (in case of the dog a FP) is different from detecting a dog when there is no corresponding sound event of any label in the ground truth (also a FP considering the dog class). Distinguishing CTs from other FPs provides deeper insight into the SED system performance. Considering the SED system from a machine listening perspective, acoustically similar classes should have a higher degree of CTs than acoustically very different classes.

To handle the above-mentioned challenges of SED evaluation, the PSDS offers a more robust way to calculate the FPs, TPs and finally the subset of FPs that falls under the category of CTs than the event based evaluation. To be more robust and put less emphasis on onset and offset times, the PSDS score relies on percentages of intersection between detected events and ground truth. The three criterions listed below can be varied to meet the requirements of the specific application of the system. The PSDS algorithm is a stepwise process, decided by the parameters Detection Tolerance Criterion, Ground Truth intersection Criterion and Cross-Trigger Tolerance Criterion. All three parameters can be varied to test the system in different scenarios:

1. The Detection Tolerance Criterion (DTC) calculates the relevance of the intersection between detections and the ground truth. If the result is greater than DTC, it is considered a valid detection and is further processed by the next step. Events that are beneath the DTC will be counted as FPs and passed on to step 3 in order to determine whether it is a CT.

2. The Ground Truth intersection Criterion (GTC) determines if the detected event that passed the DTC has enough intersection with the ground truth based on the GTC. Detected events that pass this step are counted as TP's.

3. Cross-Trigger Tolerance Criterion (CTTC): calculates the subset of FP's that also corresponds to a sound event containing another class.

Given the TPs and FPs according to the DTC and GTC, The PSDS calculates the true positive rate (recall) against the False Positive Rate (FPR) over a range of operating points, which is the same as threshold values described in the post processing section.

$$FPR = \frac{FP}{FP + TN} \tag{2.13}$$

By calculating scores over a wide range of operating points (50 by default), PSDS isolates the challenge of system tuning (i.e. finding the right operating point) from the system's performance. The PSDS framework also provides insight on the optimal threshold value for each class, which can be applied as a part of the post processing stage in practical use. When these scores are visualised, the result is the metric Area Under the Curve (AUC), which is the probability that a system will rank a randomly chosen positive instance higher than a randomly chosen negative one. An example can be seen in Figure 2.7.

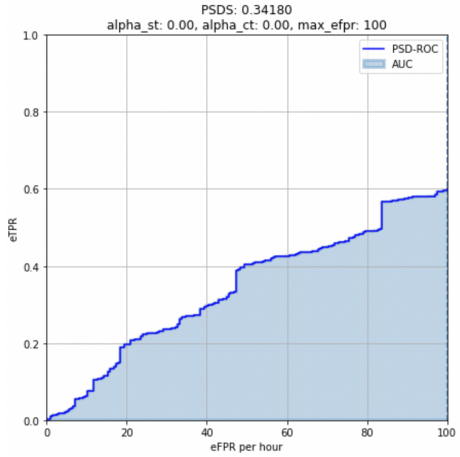


Figure 2.7: PSDS score AUC, the performance for the various operating points follow the x axis.

Chapter 3

Related work

In this chapter related work will be reviewed and explained, focusing on deep learning models using raw audio as input and notable work on graphical time-frequency representations in deep learning.

3.1 Raw audio input in SED

In 2018, a year after the proposal of the CRNN for SED applications by Cakir et al. (2017), an end-to-end CRNN using raw audio as input was suggested (Çakir & Virtanen, 2018). The model was trained to learn a time-frequency representation of raw audio. To accomplish this, the model consisted of a feature extraction block prior to the CRNN model. The feature extraction block was initialised as two layers with a set of weights producing both the real and the complex values of the Short Time Fourier Transform, and in some of the experiments also added the logarithmic scaling and the mel-frequency bank to the output of the feature extraction block. The model was trained in an end-to-end fashion, aiming to learn even more meaningful representations of the audio data. The experiment compared different strategies, sometimes fixing some of the weights in the feature extraction block and sometimes allowing all the weights to be trainable. However, the final results showed slightly worse performance than directly using time-frequency representations as input. One interesting finding was that the magnitude response for the lower frequencies (< 4 kHz) was significantly altered in the learned representation compared to a fixed time-frequency representation, while the higher frequencies (> 4 kHz) was not significantly altered during training.

After this paper, researchers were mainly focusing on using fixed time-frequency representation as input to SED systems. A quick review of the last two years of the SED challenge in DCASE shows that every system (approximately 120 SED systems) uses some version of log mel-spectrograms or log mel-energies as input. The review done by Messaros et al. on the DCASE challenge 2017 reported the same findings (Mesaros et al., 2018).

In 2022, Nam et al. investigated a new attention-based block as the frontend to a CRNN (Nam et al., 2022). They questioned the 2D Convolutional layer's ability to take into account the fact that the frequency axis in a spectrogram is not translation-invariant, while an image is translation-invariant on both x and y axes. In their experiments an attention-based frontend block was applied in order to achieve frequency dynamic convolution, before the data was further processed by a CRNN. The attention layers use attention to decide the optimal parameters for the filter in the following convolutional layer, in other words the filter weights are constantly slightly changed in order to deal

with different frequency registers. Their experiments showed a significant improvement in performance compared to a CRNN using fixed time-frequency representations directly. However the models in this paper still used log mel-spectrograms as input, and did not reattempt to work directly on raw audio.

Copiaco et al. (Copiaco et al., 2021) performed a study in 2021 comparing log mel-spectrograms with scalograms obtained from the constant wavelet transform on domestic multi-channel sound event detection. They found that the scalogram showed improved performances compared to the log mel-spectrogram as input to pre-trained computer vision models like AlexNet and ImageNet (Krizhevsky et al., 2017).

Wyse presented a paper (Wyse, 2017) which analyses computer vision CNNs when used on spectrograms. The same arguments as presented in section 2.2.2 and (Nam et al., 2022) were discussed. The 2D convolution filters treats the x and y axis as homogenous, due to the translation-invariance of visual data. However in a graphical time-frequency representations of audio this is true only when an object is shifted in the time dimension (x axis), but is not true when shifted in the frequency dimension (y axis). A sound object shifted in the frequency axis changes pitch, which changes the properties significantly.

The second element discussed in (Wyse, 2017) is the local receptive field of 2D CNNs, which emphasise that all data points that fall within the local receptive field have some significant connection to each other. Contrary to images and the visual world, this is not necessarily true for graphical time-frequency representations of audio. A sound object can consist of multiple frequencies spread apart over the frequency axis. Also, sound is transparent, so multiple sound objects could be represented within the same local receptive field of a 2D convolution filter.

3.2 Raw audio input for other classification tasks

Computer vision models apply 2D convolution in their architectures, and therefore some data transformation must be applied when working on 1D signals. In order to avoid this data transformation, which usually implies the use of hand-crafted feature extraction, Kiranyaz et al. (2015) proposed a network using 1D convolution for electrocardiogram (ECG) monitoring for medical use, and showed that even small 1D CNNs were able to handle both the feature representation learning and the classification task. In their paper, a CNN with 3 1D convolutional layers was proposed that beat the state-of-the-art in ECG monitoring at the time.

A survey of 1D CNNs (Kiranyaz et al., 2021) details how 1D CNNs, although a newer phenomena than 2D CNNs, have become very adept at working with 1D signals, and widely used in medical applications, as well as f.ex. vibration monitoring in structures and machines. They argue that 1D CNNs can learn the feature representation and classification with much fewer layers than a 2D CNN and is ideal for real-time use in devices with lower computational resources.

In the field of environmental sound classification, 1D CNNs working with raw audio have been attempted with great success. Environmental audio classification is a pure classification problem without the localisation in time of the sound event, where an audio file of any length is classified as containing a target class or not.

In (Dai et al., 2017) a 1D CNN with 35+ 1D convolutional layers using raw audio as input was proposed, whitch performed at the level of 2D CNNs using fixed time-frequency representations. However the size of the CNN was huge, comparable to computer vision models using 2D CNNs. In 2019 it was proposed a smaller 1D CNN model working with raw audio as input whitch matched the state-of-the-art in environmental

sound classification (Abdoli et al., 2019) at the time. The argument supporting the 1D convolutional architecture in audio analysis is that models learn to extract time-frequency information from the raw audio signal using large filters (kernel size of 128 or 64) resulting in a large local receptive field in the early layers of the model, with the following layers having progressively smaller local receptive fields in order to learn higher-level representations of the audio signal. However as their system was not tasked with localisation in time, their models could reduce the dimensionality of the temporal axis freely. When working with a 1D audio signal in a 1D CNN, the temporal axis is the only meaningful axis to apply dimensionality reduction. Dimensionality reduction is crucial for CNN to learn increasingly high-level features. When using 1D CNNs in a SED system, this poses a problem as the temporal resolution cannot be reduced beyond the expected temporal precision of the system.

SoundDet (He et al., 2021) was suggested as a framework using raw audio as input for SELD applications in 2021. The SoundDet block capable of learning a meaningful representation from raw audio started with a layer with 256 rectangular band-pass filters (each filter was a Finite Impulse Response filter with 481 coefficients), however the trainable parameters of each filter was just the lower frequency cutoff and the higher frequency cutoff. With a low number of trainable parameters the computational burden was reduced for training the system compared to using convolutional layers with a kernel size of 481. The output of the band-pass layer was fed to a 1D convolutional encoder-decoder consisting of seven layers with small receptive fields (kernel size of 3 and strides of 2), gradually reducing the temporal dimension through the strides until a frame level representation was obtained equalling 100 ms of audio for each frame. The output of this first block was then fed to two separate blocks responsible for the event detection as in a classic SED problem and the localization of the sound source respectively. SoundDet showed promising results, outperforming the baseline SED system it was compared to.

3.3 Raw audio input in other applications

In 2016 Google Deepmind presented WaveNet (Oord et al., 2016), a generative model for raw audio. Applied to text-to-speak applications it clearly outperformed the state-of-the-art. WaveNet is an autoregressive deep learning model trained on raw audio waveforms, utilising dilated 1D convolutional layers to predict the next sample of a sequence. Dilated convolution is convolution where the filter is applied over an area larger than the kernel size by skipping input values in certain steps. This makes the model capable of taking a large segment of audio as input and increase the local receptive field of filters without greatly increasing the computational cost by using large kernel sizes (see Figure 3.1). The architecture and ideas behind WaveNet could be applied to generate all sorts of

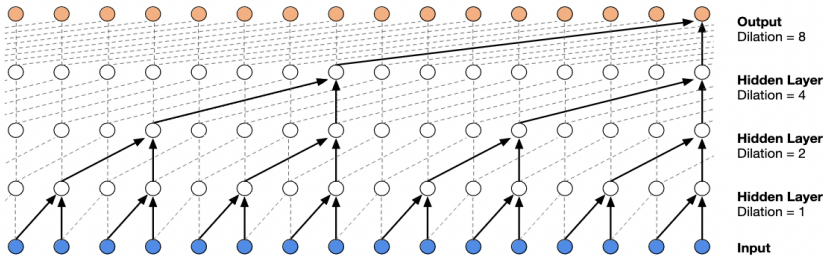


Figure 3.1: Illustration of four dilated convolutional layers, from (Oord et al., 2016)

audio, not just speech, and was quickly inspiring work in other fields than text-to-speech applications.

In the field of analog audio effect modelling, Ramirez et al. (2019) tried similar approaches. They came up with a simpler model which was able to generate audio modelling the behaviour of an analog effect when given dry audio (ie a signal not effected by the audio effect) as input, which at the time outperformed state-of-the-art in analog effect modeling and WaveNet.

Their model CAFx used a block of two 1D convolutional layers (the last one applying depthwise convolution) and a max pooling layer to create the learned audio representation which was then fed to a RNN. Finally a block of deconvolutional layers acted as a decoder to create the audio waveform. The two convolutional layers in the first block were made up of 128 filters each, the first layer had a kernel size of 64 and the second layer 128. CAFx followed an encoder-decoder approach, and was also pre-trained as a pure encoder-decoder before it was tasked to learn the analog effect behaviour. This learned audio representation was much smaller and more efficient than prior work, and will be explained in detail in chapter 4.

3.4 Summary

Raw audio input has been attempted, but largely discarded in SED. In other fields raw audio input has been used with great success, and the question is whether SED could benefit from retrying this approach? Cakir and Virtanen's approach in (Çakir & Virtanen, 2018) may have been flawed by the use of half it's feature learning block handling the complex numbers of the STFT. The complex numbers resulting from the STFT are used whenever a signal needs to be brought back to the time domain through the inverse fourier transform, however the information in the complex numbers describe the phase of the signal which does not contain much useful information in the case of audio classification.

Raw audio input is computationally more expensive than log mel-spectrograms. A square log mel-spectrogram with 128 bins and 128 timesteps results in 16384 unique data points. In a sample rate of 16 kHz this will be approximately equal to 1 second of raw audio. However the log mel-spectrogram could be a time-frequency representation of any length of sound, even though the time resolution will decrease the longer the time window represented is. Using log mel-spectrograms enables more audio to be fed into the model for the same amount of data points as for raw audio, whose size is decided by its length and sample rate.

Both (Nam et al., 2022) and (Wyse, 2017) mention the discrepancy between the 2D Convolutional operation and how the frequency axis in a time-frequency representation of sound is not translation invariant. The CRNNs designed for SED work around this through dimensionality reduction only on the frequency axis, however it remains to be explored whether this is a fair use of 2D CNNs, or if it is limiting the ability of the CNN to extract high-level features.

The local receptive field of CNNs implies that data points lying close to each other (within the receptive field) belong to each other. If one should translate this to time-frequency representations of audio, the local receptive fields of the first CNN layers would have to be much larger than what is traditionally applied. A sound event can be spread over a huge frequency spectrum, from a fundamental frequency in the low range (60 - 500 Hertz) to overtones in the very top range of the frequency spectrum.

Another key aspect is the log mel-spectrogram which is based on human perception, emphasising the same frequency regions as our auditory perception system. There are mammals with far more advanced auditory perception systems, and technology enables us to capture sound with much higher frequency ranges than we can hear. And not just the frequency ranges are affected, we can analyse sound in a much higher time resolution than our perceptions are capable of discerning. When developing machine listening systems, there might be applications where the analysis requires higher frequency content or finer time resolution. By using the log mel-spectrogram as input one can argue that we are limiting ourselves by the limits of the human auditory perception system, rather than using the extended resolution in both time and frequency enabled by technology.

1D CNNs are becoming increasingly successful in learning high-level representations of 1D data, and could also be computationally cheaper than 2D convolution. In many fields of audio modelling and audio analysis 1D CNNs represents the state-of-the-art, however they are yet to be applied to great extent in SED due to the following reasons: i) The argument that 1D convolution is a simpler computational operation than 2D convolution does not outweigh the fact that time-frequency representations can present the input audio to the model with less data than raw audio. ii) Temporal resolution matters in SED, and 1D CNNs are therefore not able to apply full-scale dimensionality reduction on the temporal axis, limiting their capabilities of learning high-level representations of the audio signal. iii) 2D CNNs have been used with success on graphical time-frequency representations, and research has therefore focused on improving other areas of SED models architecture. Modern day state-of-the-art SED systems apply many complex concepts such as ensemble models (Lin et al., 2019) and majority voting, mean-teacher models (Kim & Kim, 2021) and transformer blocks replacing the RNN (Miyazaki et al., 2020).

Chapter 4

Experimental Design

The objective of this thesis is to understand the extent to which novel deep architectures are capable of learning a meaningful representation directly from raw audio as input, and compare these with models using fixed time-frequency representation log mel-spectrograms as input. This chapter explains the model architectures and the evaluation strategies used, and the dataset used for training and evaluation. Models and code to replicate the experiments can be found at the accompanying Github site[1].

4.1 Key concepts

In order to investigate how deep learning models learn from raw audio as input, multiple pairs of models are trained and compared. The model pairs use either log mel-spectrograms or raw audio as input. The latter models have a block of layers in the beginning of the model responsible for learning a representation of the raw audio. The output from this block or a log mel-spectrogram is then fed into a CNN, responsible for extracting the high-level feature representation. The CNN output is fed sequentially into a RNN, and finally fully connected feedforward layers are responsible for predicting the probabilities for each class in the current analysis frame.

In order to investigate whether the 2D Convolution suffers from the non-translation invariant nature of sound, models where the CNN block consists of 1D convolutional layers have also been trained.

For simplicity, all models are presented with the same raw audio as input. The models which use the log mel-spectrogram as input utilise a custom layer constructed in Tensorflow which extracts the log mel-spectrogram as the first layer in the model. The log mel-spectrograms are created with an STFT size of 2048, 128 mel-frequency bins and a hop length similar to the analysis frame size. The models which use raw audio as input are constructed with a block replacing the log mel-spectrogram layer, responsible for learning the representation of the audio, detailed in section 4.2.1. The audio is ordered in segments of 40960 samples (approximately 1.8 seconds of audio) using a sample rate of 24 kHz.

Human hearing can discern gaps and changes in amplitude down to approximately 2.5 ms (Plack, 2018). It is an interesting question whether a deep learning model can emulate the same behaviour. SED systems traditionally use analysis frames ranging from 10 ms to 100 ms, but it would be interesting to see whether very small analysis frames can be used, and how the two different inputs will be affected by reducing the

[1] https://github.com/arvidfalch/RawAudio_e2e

analysis frame size. For comparing results, most of the main model architectures are trained on two different analysis frame sizes, 64 samples (approximately 2.6 ms of audio) or 256 samples (10 ms of audio) (See Table 4.4). The choice of analysis frame size is an important decision when designing a SED system, and this method allows us to compare two different analysis frame sizes.

4.1.1 Dataset

The dataset used for training is the strong labelled synthetic dataset from DCASE task 4 2021 (Serizel et al., 2020). The training dataset consist of 10k segments of 10 seconds referred to as sound scenes. Audio clips of varying length of both target sound events and non target sound events are randomly distributed among the sound scenes. The dataset is constructed using the Scaper library (Salamon et al., 2017) with the same parameters as for the DCASE 4 task from 2021.

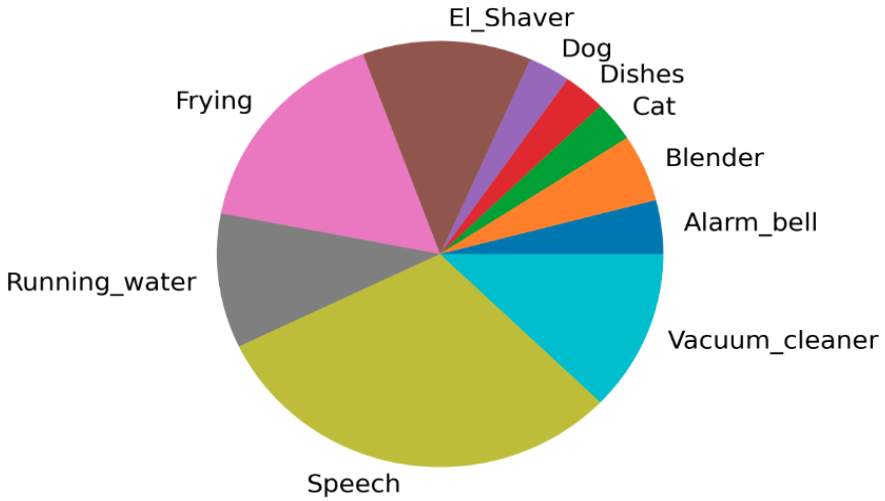


Figure 4.1: Training dataset class distribution

An additional 1000 segments are used as validation set during training. The evaluation dataset consists of 692 audio files of 10 seconds, and is the same as what is used as the synthesised public evaluation set in DCASE task 4 from 2018 (Chan & Chin, 2020).

The dataset consists of 10 target sound classes: alarm bell ringing, blender, cat, dishes, dog, electric toothbrush, frying, running water, speech and vacuum cleaner. The target audio event clips are taken from the Freesound Dataset (FSD) (Fonseca et al., 2017). The non-target sound events are divided into background sound events (considered as noise and subject to a SNR ratio to foreground events) and foreground events, which are treated similar to the target sound events.The foreground non-target audio events are also taken from the Freesound Dataset. The background non-target audio event clips are from the SINS dataset (Dekkers et al., 2017) and the TUT scenes 2016 development dataset (Mesaros et al., 2016b).

By counting the occurrence of each class for every analysis frame we can get a perspective of the class distribution in the training dataset. As can be seen in Figure 4.1, the class distribution is uneven, and we can expect the models to struggle to learn the classes that only make up less than 5% of the training data target sounds. The classes

alarm bell ringing, cat, dishes, dog and speech are considered short sound events, which in the case of all but speech correlates to the low representation in the training set.

Figure 4.1 does not take into account occurrences of non-target sounds or silence.

Since 2018, the DCASE task 4 has focused on the combination of strongly labelled datasets with weakly labelled datasets (Serizel et al., 2018), encouraging the research in semi-supervised learning tasks. The availability of strongly labelled datasets for SED was found to be poor, and therefore severely limiting the research in SED for purely supervised learning tasks after the 2017 DCASE challenge (Mesaros et al., 2019). The creation of the DCASE synthesised dataset with Scaper does present us with a decent sized strongly labelled dataset for such an experiment as this thesis, but is still considered too small to properly train a SED system achieving state-of-the-art performance.

4.2 Model architectures

4.2.1 Learnable Audio Representation block

The Learnable Audio Representation (LAR) block is the proposed architecture for allowing the deep learning models to work directly with audio. The LAR block is inspired by the works of Ramirez et al. (2019) modelling analog audio effect with deep learning models, where a version of this block was capable of encoding the signal into a RNN and then replicating the signal through a back end as explained in 3.3. Raw mono audio is used as input in segments of S samples, resulting in an input shape to the model of (S,1).

The input audio segment is fed through a 1D convolutional layer with 128 filters with kernel size of 64. The absolute value of the dot product is used as activation function after batch normalisation, resulting in an ouput shape of (S,128).

The output of the first convolutional layer is fed to a depthwise 1D convolutional layer consisting of 128 filters with kernel size of 128. Depthwise convolution only applies each filter to the corresponding filter output of the first layer, resulting in less weights and acting as a second parallel filter to the filter of the first layer. To introduce non-linearity to the LAR block the ReLU is used as activation function for the depthwise convolutional layer after batch normalisation has been applied. The ReLU activation function has also been shown to speed up the learning process of a neural network (Glorot et al., 2011).

$$Relu(z) = max(0, z) \tag{4.1}$$

The kernel size of 64 and 128 represents quite large filters and is computationally demanding. However by using large kernels the model should be able to learn audio features with longer time-dependencies like amplitude envelopes (Abdoli et al., 2019), (Dai et al., 2017).

The output of both the first and the second layer is concatenated (resulting in the shape (S, 256)), and then average pooled over the filter axis with a pool size of 2 to the shape (S, 128). This operation acts as a residual connection where also the pure output of the first layer is used, which should speed up the learning process and improve the independence of each layer (Szegedy et al., 2017).

Both convolutional layers utilise zero-padding on the edges to preserve the length of the input segment.

Finally the output of each filter is maxpooled over the temporal axis, using the desired analysis frame size F_{size} as pool size.

$$N_{frames} = S/F_{size} \tag{4.2}$$

This results in a shape of (N_{frames}, 128, 1), where the first axis is the temporal resolution and the second channel represents the filters. The log mel-spectrograms used for the models not built with the LAR block are computed to have the same output shape as the LAR block output.

Bias is disabled for the convolutional layers in the LAR block. The bias in a convolutional layer works the same way as for a feedforward layer, it is added to the result of the convolutional operation of the input signal and weights.

$$y = w * x + b \tag{4.3}$$

where y is the output, w are the weights of the filter, x is the input signal and b is the bias. Each filter has an individual trainable bias value. Disabling bias makes it possible to reproduce each filter as a Finite Impulse Response (FIR) filter where the coefficients of each filter are the same as the weights in the layer. This enables the system to be used as a fixed filterbank after it is trained, where the filters are applied through multiplication in the frequency domain rather than convolution, speeding up the computational time of the LAR block when doing inference. Turning off the bias does reduce the hypothesis space of the LAR block, but preliminary tests did not show any drop in performance between models without bias and models with bias enabled. Equation 4.4 describes the output function for the convolutional layer without bias.

$$y = w * x \tag{4.4}$$

The major changes from (Ramirez & Reiss, 2019) architecture are:

- The residual connection with average pooling over the filter channels
- Bias disabled for both convolution layers
- ReLU as activation function for the Depthwise convolutional layer (Softplus was originally used)
- The input segments are 40960 samples long compared to 4096 in (Ramirez & Reiss, 2019).

Alternate LAR block architectures were attempted in preliminary tests, and a full experiment with two alternative LAR blocks was also conducted. This consisted of three individual depthwise convolutional layers without the residual connection of the first layer, and resulted in 3 channels in the output as the output of these three layers were concatenated over the channel axis (outputting a shape of (N_{frames}, 128, 3) after the max pooling layer. This model is referred to with the suffix **LAR_3**. The second of these models were trained purely with the hyperbolic tangent as activation function, in an attempt to keep the output of the filters even closer to an actual audio filterbank. This model is referred to with the suffix **LAR_3** *tanh*.

The filter weights of both convolutional layers in the LAR block are randomly initialised before training. In (Vecchiotti et al., 2019) and (Abdoli et al., 2019), both randomly initialised filter weights and filter weights initialised as gammatone filterbanks were attempted. (Abdoli et al., 2019) found an improval in models initialised as gammatone filterbanks, however (Vecchiotti et al., 2019) found the opposite to produce the best results. Preliminary tests did suggest a slight improvement when the weights of the LAR block convolutional layers were initialised as gammatone filterbanks, however to allow the LAR block to learn the audio representation from random initialisation would possibly shed more light on the process taking place in the LAR block.

The architecture of the Learnable Audio Representation block can be seen in Table 4.1.

Learnable Audio Representation (LAR) Block				
Layer type	Filters/ Hidden Units	Filtersize	Activation	Output Shape
Input	-	-	-	(40960, 1)
Convolution 1D	128	64	-	(40960, 128)
Batch Normalisation	-	-	-	(40960, 128)
Activation	-	-	Abs	(40960, 128)
Depthwise Convolution 1D	128	128	-	(40960,128)
Batch Normalisation	-	-	-	(40960, 128)
Activation	-	-	ReLU	(40960, 128)
Concatenation	-	-	-	(40960,256)
AveragePool channels	-	2	-	(40960, 128)
Maxpool	-	256	-	(160, 128)
Number of parameters: 25600				

Table 4.1: Learnable Audio Representation Block, where $S = 40960$, $F_{size} = 256$

4.2.2 2D Convolution CRNN

The 2D Convolution CRNN is the same model as described in (Adavanne et al., 2017: Mesaros et al., 2021: Adavanne et al., 2019), which can be considered a simple baseline SED system. Originally it takes as input a *log mel spectrogram*, uses 2D Convolution with pooling over the frequency axis, and has two bidirectional GRU layers as its recurrent part. The number of hidden units in the bidirectional GRU layers have been increased from 32 to 64 after preliminary tests [2]. The Bidirectional GRU layers use the hyperbolic tangent as activation function:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{4.5}$$

The two fully connected dense layers are applied after the RNN block, the first with 32 neurons and the hyperbolic tangent as activation function, and the final layer has 10 neurons representing each class and use the sigmoid activation function.

The 2D CRNN can easily be modified by applying the LAR block in the beginning of the model instead of the layer extracting log mel-spectrograms. In this way we can compare models which are similar in all aspects except for the LAR block.

Models using the 2D CRNN are referred to as **LOG2D** for the models using log mel-spectrogram input, and **RAW2D** for the models using raw audio as input.

The 2D CRNN architecture is fully described in Table 4.2, and a very similar implementation is illustrated in Figure 2.2.

[2]Original code for the baseline CRNN model can be found at https://github.com/sharathadavanne/sed-crnn/blob/master/sed.py

2D CRNN				
Layer type	Filters/ Hidden Units	Filtersize	Activation	Output Shape
Input	-	-	-	(40960, 1)
Log Mel-Spectrogram / LAR block	-	-	-	(160, 128, 1)
Convolution 2D	128	(3,3)	ReLU	(160,128,128)
Maxpool 2D	-	5	-	(160,25,128)
Convolution 2D	128	(3,3)	ReLU	(160,25,128)
Maxpool 2D	-	2	-	(160,12,128)
Convolution 2D	128	(3,3)	ReLU	(160,12,128)
Maxpool 2D	-	2	-	(160,6,128)
Stack filters	-	-	-	(160, 768)
BiGRU	64	-	tanh	(160,64)
BiGRU	64	-	tanh	(160,64)
Dense (Time distributed)	32	-	tanh	(160,32)
Dense (Time distributed)	10	-	sigmoid	(160,10)
Number of parameters: 619498 with log mel / 634498 with LAR block				

Table 4.2: 2D CRNN architecture (Batch normalisation and dropout layers have been excluded).

4.2.3 1D CRNN architectures

For the models trained on analysis frame size of 64, a model architecture utilising only 1D convolutional layers were designed. After the LAR block or the log mel-spectrogram, a CNN consisting of 3 1D convolutional layers, each with 512 filters with kernel size 3. No strides were used in order to preserve the temporal resolution, however a maxpooling operation with a pool size of 4 was used after each convolutional layer to bring the output shape of the filter axis to 128. ReLU and batch normalisation was also used in each of the three layers. As 1D CNNs operate on a 1D signal, the frequency axis of the log mel-spectrogram or LAR block is considered to be the channel axis, and the convolutional operation is applied independently for each channel. This was then fed into the same RNN architecture as the models with 2D convolutional layers. This architecture had 746k trainable parameters for the LAR block version and 721k parameters for the version using the log mel-spectrogram as input. The number of parameters are slightly higher than the 2D convolutional models, but not significantly bigger. These models are referred to as either **LOG1D** or **RAW1D**. An illustration of the main model architectures can be seen in Figure 4.2.

For the models trained on analysis frame size of 256, another 1D convolutional model was designed to resemble the architecture in (Abdoli et al., 2019) which had shown promising results in environmental audio classification. This model only works on raw audio, and does not use the LAR block architecture. Some changes had to be done in order for the model to be able to feed sequential data to the RNN, so the temporal dimensionality reduction had to be reduced compared to the model proposed by Abdoli et al. The RNN is the same as for all the other models.

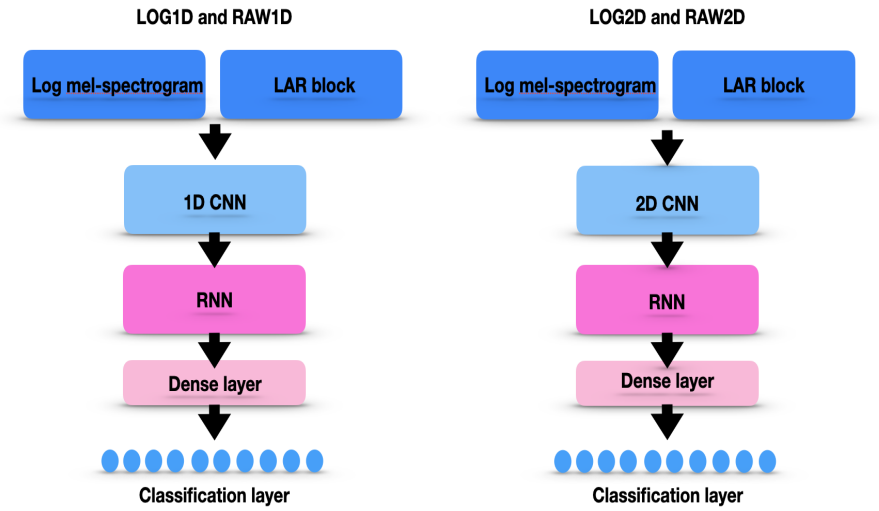


Figure 4.2: An illustration of the main model architectures

The main principle is to use filters with a large local receptive field in the first layer, followed by a large dimensionality reduction before the next layers, where decreasing kernel sizes are used to learn higher level features. The model is considerably larger than the other models used in the experiments (1.3 million parameters). The full architecture is shown in Figure 4.3. This model is referred to as **PURE1D**.

4.3 Main Experiments

The main experiment is the evaluation of all the models on the DCASE public evaluation set. The main metrics used are:

- **PSDS score**, set with a low Detection Tolerance Criterion (0.1), low Ground Truth intersection Criterion (0.1), a Cross Trigger Tolerance Criterion of 0.3, and a max false positive rate of 100 per hour. This is similar to the PSDS 2 scenario in (Chan & Chin, 2020) and (Ronchini & Serizel, 2022). This scenario is quite tolerant on the segmentation task, allowing for less precise onset and offset predictions. The PSDS score is calculated over 50 operating points linearly scaled from 0.01 to 0.99.
- **F1 segment score**, with a segment size of 1 second.
- **ER score**, calculated from the segment evaluation with segment size 1 second
- **F1 event score**, with an onset collar of 200 ms and an offset tolerance of 50%.
- **Macro-average F1 score**, The macro-average F1-score is the average of class-wise F1-scores, as opposed to micro-average where the number of true positives, etc. is counted across all classes (Mesaros et al., 2016a).

In order to understand better how the different SED models are learning the dataset, the class-wise F1 score is an F1 score calculated individually for each class. It is calculated based on the PSDS framework, according to the selected parameters described above for the PSDS score. However it is not calculated over multiple operating points, only for the selected threshold value 0.5.

1D CRNN (**PURE1D**)				
Layer type	Filters/ Hidden Units	Filtersize/ Poolsize	Strides	Output Shape
Input	-	-	-	(40960, 1)
Convolution 1D	128	64	2	(20480,128)
Maxpool 1D	-	8	4	(5119,128)
Convolution 1D	128	32	2	(2560,128)
Maxpool 1D	-	8	2	(1277,128)
Convolution 1D	128	16	2	(639,128)
Maxpool 1D	-	2	1	(638,128)
Convolution 1D	128	8	2	(319,128)
Convolution 1D	256	4	2	(160,128)
BiGRU	64	-	tanh	(160,32)
BiGRU	64	-	tanh	(160,32)
Dense (Time distributed)	32	-	tanh	(160,32)
Dense (Time distributed)	10	-	sigmoid	(160,10)
Number of parameters: 1,235,050				

Table 4.3: 1D CRNN architecture of **PURE1D** (Batch normalisation and dropout layers have been excluded, but they follow each convolution layer. ReLU was used as activation function for all convolution layers).

The output of all models have been post-processed by taking the mean value over a window of 150 ms. The detection threshold for F1 scores and ER scores are 0.5.

PSDS score, class-wise F1 score and Macro Average F1 score are extracted using the Python library *PSDS_eval* (Bilen et al., 2020) [3], while the rest of the metrics are extracted with the *sed_eval library* (Mesaros et al., 2016a) [4].

4.3.1 Preliminary experiments

Preliminary experiments were conducted training models on 10 % of the full data set for 100 epochs. These tests focused on the design and hyperparameter search for the learnable audio representation block, as the rest of the model architectures could use hyperparameters established in prior research.

4.3.2 Model training

For the main experiments, all models are trained for 40 epochs on the full dataset. This allows the conclusion of the experiments within the time constraints of the thesis on the GPU resources available. The models trained on raw audio also have to train the LAR block within the same number of epochs, which put these models at a slight disadvantage in the experiment compared to the models using log mel-spectrogram input. During preliminary tests, it was evident that the log mel-spectrogram models converged quicker, and reached convergence around 40 epochs.

The best performing model pair from the main experiment are trained for 70 epochs to analyse whether they improve or if additional epochs result in overfitting.

All the models can be seen in Table 4.4.

Model Name	F_size 64	F_size 256	40 epochs	70 epochs
LOG2D	✓	✓	✓	✓
RAW2D	✓	✓	✓	✓
LOG1D	✓	No	✓	No
RAW1D	✓	No	✓	No
PURE1D	No	✓	✓	No
RAW2D LAR_3	No	✓	✓	No
RAW2D LAR_3 *tanh*	No	✓	✓	No

Table 4.4: Overview of models trained, number of epochs and analysis frame sizes.

The loss function used is binary crossentropy, which is calculated as

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{4.6}$$

where M is the number of classes, y is the binary indicator (0 or 1) if class label c is the correct classification for observation o, and p is the predicted probability that o is of class c.

The optimizer used is Adam (Kingma & Ba, 2014). Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. A learning rate of 0.001 was used throughout the experiments.

[3] https://github.com/j-bernardi/psds_eval

[4] http://tut-arg.github.io/sed_eval/

Dropout is used as regularisation method for all the models models, to prevent them from overfitting (Srivastava et al., 2014). A dropout value between 0 and 1 gives the probability that a neuron will output 0. Dropout is applied after each convolutional layer and each of the recurrent layers. The recurrent layers also have recursive dropout. All models are trained with a dropout value of 0.5, similar to what is suggested in (Mesaros et al., 2021) and used in (Adavanne et al., 2017).

4.4 Additional Experiments

A SED system used in real life applications has to meet some conditions that are not covered by the main experiments of this thesis. This section explains two different experiments which are designed to further stress-test the SED models in this thesis. In a real life application, a SED system will handle audio recorded in physical locations on microphones and in conditions that might not resemble the training data audio. In the case of domestic sounds as in the DCASE task 4, we can imagine it being used in a home with large open rooms causing extra reverberation, or connected to a microphone which has a different filter response than the training data audio. If we think of SED systems designed for outdoors use, extra factors such as wind, insects, rain and previously unheard animal sounds will be prevalent.

4.4.1 Evaluation on non-target sounds

A question has been raised how SED systems are affected by detecting false positives when presented with only non-target sounds (Ronchini et al., 2021). They conclude that systems designed with strong emphasis on classification compared to segmentation performed better in such a case, and that the amount of non-target events in the training set corresponded with the SED systems ability not to detect false positives. Further work in (Ronchini & Serizel, 2022) showed how performance of state-of-the-art SED systems dropped when tested on an evaluation set consisting just of non-target sounds.

A dataset with audio from Soundly [5] created for this thesis are used to test the models on only non-target audio (available at the accompanying Github repository). The non-target dataset consists of 100 minutes of audio, with sound events from natural ambience, birds, traffic, vehicles, industrial sounds and ocean waves. No target sound events are present.

4.4.2 Evaluation on filtered audio

The evaluation dataset is filtered before it is fed into the models for prediction, in two separate tests.

- Low pass filter with a cutoff frequency of 500 Hz
- Hi pass filter with a cutoff frequency of 500 Hz

The low pass filtered evaluation audio will be subject to the biggest loss in information. The filters are constructed as FIR filters of order 99.

[5] https://getsoundly.com

Chapter 5

Results and analysis

5.1 Results

5.1.1 Training time

Various computational resources were used for this this project, including the shared ML Node 8 [1] consisting of 4 A-100 GPUs, and the Norwegian Research and Education Cloud service (NREC)[2] with two virtual machines with Tesla P40 with 12GB of GPU memory. Training times were consistent between the latter two, while the ML nodes were highly inconsistent in training time based on the amount of users and usually much slower than the NREC resources. Table 5.1 shows training time for completion of 1 epoch on the NREC virtual machine for models trained on analysis frame size of 256 samples. The smaller analysis frame size of 64 causes higher temporal resolution and therefore slows down the training as the RNN must operate on a much longer sequence. Training the LOG2D and RAW 2D for 70 epochs each on analysis frame size of 256 will take approximately 9 days on a similar setup as one of the NREC virtual machines, and considerable longer training times are to be expected for the other model architectures if one wants to replicate all the trained models of this thesis. A total of 60 days were used to train all the models for the main experiments.

Model	1 epoch training time
LOG2D	72 minutes
RAW2D	112 minutes
RAW2D_3channels	140 minutes

Table 5.1: Training times for completion of one epoch for selected models.

5.1.2 Main experiments

Analysis Frame Size 64

For the models trained on an analysis frame size (F_{size}) of 64 samples, the performance metrics in Table 5.2 show that the models using raw audio as input outperform its counterparts using log mel-spectrograms.

[1] Machine learning infrastructure (ML Nodes), University Centre for Information Technology, University Of Oslo, Norway.

[2] https://www.nrec.no/

Model Name	PSDS	F1_seg	ER	F1_event	Macro Avg. F1
LOG2D	0.2059	46.41%	0.87	7.41%	27.72%
RAW2D	**0.3191**	**48.14%**	**0.73**	**9.40%**	**31.96%**
LOG1D	0.1799	40.41%	0.92	7.25%	27.46%
RAW1D	0.2331	42.19%	0.94	6.83%	29.32%

Table 5.2: Performance metrics for 64 samples analysis frame size. Best scores in bold.

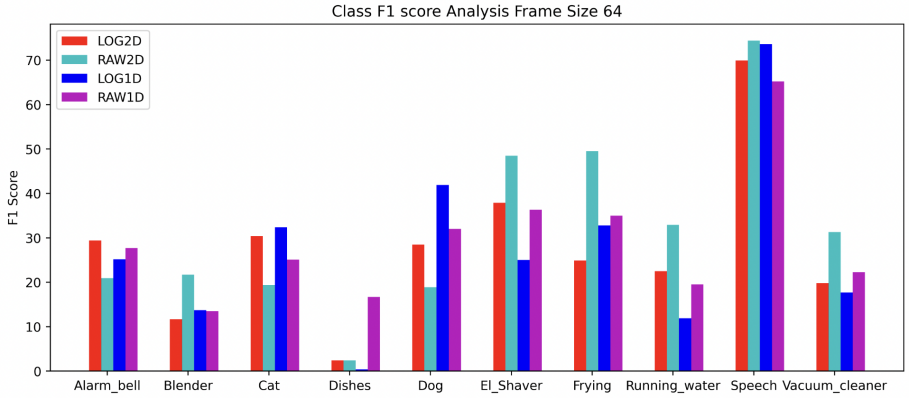


Figure 5.1: Class wise F1 Score for models trained on analysis frame size 64.

In terms of 1D CNN versus 2D CNN, RAW2D achieves better performance than RAW1D. LOG2D and LOG1D are harder to distinguish, with the Macro Average F1 score and the class F1 score (Figure 5.1) indicating that the two models are evenly matched. However LOG2D outperforms LOG1D in PSDS score and F1 segment score.

The low F1 event scores indicates that all the models struggle with precise onset detection, and analysis shows that the models often detects multiple short sound events in the case where there is only one long sound event, which according to the F1 event score counts as FPs.

As seen in Figure 5.1, all models perform best on *Speech* which has the biggest representation in the training dataset (Figure 4.1). RAW2D achieves the best class wise F1 score on classes *Electric shaver*, *Frying*, *Running water* and *Vacuum cleaner*, which are also well represented in the training dataset.

For the class *Dishes* which is not well represented in the training dataset (3%), RAW1D achieves the best score.

Standard deviation for the F1 Class score is seen in Table 5.3, where RAW1D achieves the most even performance among the classes.

Model Name	σ
LOG2D	17.00
RAW2D	19.53
LOG1D	19.13
RAW1D	13.97

Table 5.3: Standard deviation for F1 class score, analysis frame size 64

Model Name	PSDS	F1_seg	ER	F1_event	Macro Avg. F1
LOG2D	**0.3418**	**52.42%**	0.75	**9.74%**	**37.15%**
RAW2D	0.3265	47.72%	**0.72**	9.34%	30.69%
PURE1D	0.2496	41.07%	0.82	6.25%	29.92%

Table 5.4: Performance metrics for 256 samples analysis frame size. Best scores in bold.

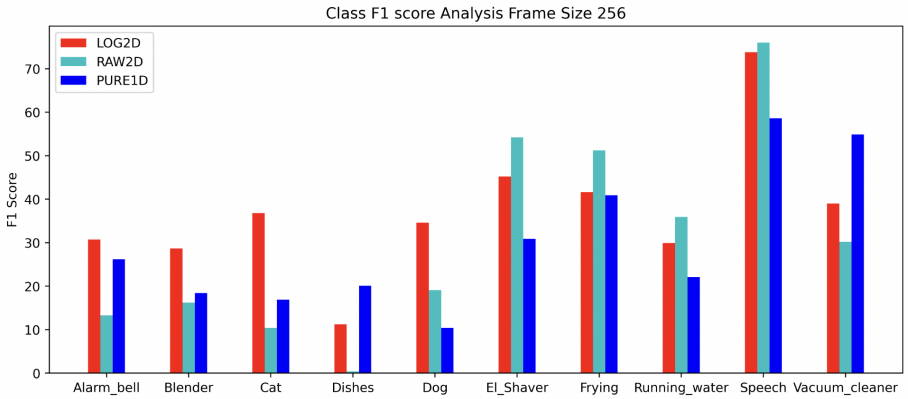


Figure 5.2: Class wise F1 Score for models trained on analysis frame size 256.

Analysis frame size 256

When we look at the performance metrics for the models using F_{size} of 256 samples (Table 5.4), PURE1D is clearly worse compared to the two other models.

Contrary to the F_{size} 64 experiment, this time LOG2D outperforms RAW2D except for the ER metric.

Analysing the class F1 score seen in Figure 5.2, it is clear that RAW2D performs best on the same classes as in the experiment for F_{size} 64, except for *Vacuum Cleaner* where PURE1D achieves the best performance. Table 5.5 shows the standard deviation for the F1 class score. LOG2D and PURE1D have a more even performance among the classes than RAW2D. PURE1D which does not use the LAR block seems to be less affected by the class distribution in the training dataset compared to RAW2D.

LOG2D improves the most compared to the F_{size} 64 experiment, while RAW2D seems to perform evenly between the two F_{size} experiments.

F1 event scores are again low, comparable to the F_{size} 64 experiment.

Model Name	σ
LOG2D	15.09
RAW2D	22.42
PURE1D	15.81

Table 5.5: Standard deviation for F1 class score, analysis frame size 256

70 epochs experiment

The two best models trained on F_{size} 256 was selected for this experiment, LOG2D and RAW2D. After training for 70 epochs, RAW2D improves while LOG2D performs worse, which might be due to overfitting.

Model Name	PSDS	F1_seg	ER	F1_event	Macro Avg. F1
LOG2D	0.2671	51.02%	0.82	9.67%	**36.82%**
RAW2D	**0.3571**	**51.65%**	**0.70**	**10.36%**	35.45%

Table 5.6: Performance metrics for models trained for 70 epochs. Best scores in bold.

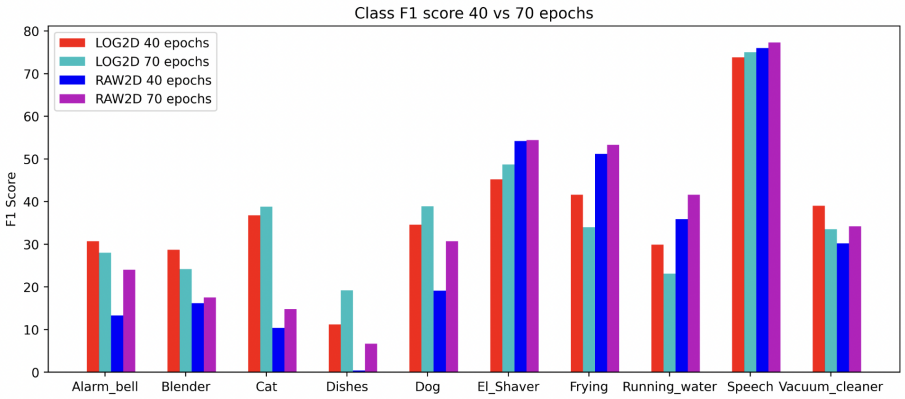


Figure 5.3: Class wise F1 score for models trained for 40 and 70 epochs

For LOG2D, The PSDS score drops from 0.3418 after 40 epochs to 0.2671 after 70 epochs. RAW2D improves, from a PSDS score of 0.3265 after 40 epochs to 0.3571 after 70 epochs (Table 5.6).

Figure 5.3 shows the class F1 score for models after 40 and 70 epochs. RAW2D does improve in every class from 40 to 70 epochs, but still struggles with the five least represented classes in the training dataset. LOG2D does improve in 5 classes but does worse on the other half of the classes.

Alternative LAR block

Two models were trained with alternative LAR blocks, using three separate dephtwise convolution layers concatenated over the channel axis. The first model RAW2D LAR_3 used the same activation functions as the original LAR block (Abs and ReLU), while the second model RAW2D LAR_3*tanh* used the hyperbolic tangent for all the convolutional layers. Trained for 40 epochs they both performed worse than the corresponding model using the traditional LAR block, the RAW2D. With a PSDS score of 0.2992 RAW2D LAR_3 was significantly better than RAW2D LAR_3*tanh* with a PSDS score of 0.2114.

5.1.3 Additional experiments results

Filtered audio evaluation

Low Pass Filtered (LPF) audio was clearly the most challenging, as it removes much more information than in the High Pass Filtered (HPF) audio. Models trained on F_{size} 64 suffered the highest performance drop when evaluated on LPF audio. The model using raw audio as input and trained on F_{size} 64 (RAW2D 64) reported the highest drop in performance. Table 5.7 shows the performance metrics for models evaluated on LPF audio.

The HPF evaluation audio improved the performance of both models trained on F_{size} 256, while both models trained on F_{size} 64 experienced a small drop in performance. LOG2D trained on F_{size} 256 evaluated on HPF audio scores the highest PSDS score

(0.3696) of all the experiments. HPF audio removes high energy content in the low frequency register which are often representing background noise, which can explain the improvement. Table 5.8 shows the performance metrics for models evaluated on HPF audio.

Low pass filtered audio				
Model name	PSDS		Macro F1	
RAW2D 256	0.110	-0.22	13.51%	-17.18
LOG2D 256	0.164	-0.17	18.27%	-18.88
RAW2D 64	0.059	-0.26	4.97%	-26.99
LOG2D 64	0.110	-0.10	13.94%	-13.78

Table 5.7: Performance metrics for evaluation on LPF audio with difference from main experiment.

High pass filtered audio				
Model name	PSDS		Macro F1	
RAW2D 256	0.3319	+0.01	31.55%	+0.86
LOG2D 256	0.3696	+0.03	38.24%	+1.09
RAW2D 64	0.3038	-0.02	30.82%	-1.14
LOG2D 64	0.1846	-0.02	26.4%	-1.32

Table 5.8: Performance metrics for evaluation on HPF audio with difference from main experiment.

Non-target sounds evaluations

An interesting observation from (Ronchini & Serizel, 2022) was that detections of the short audio event classes, defined as *Alarm bell ringing*, *Cat*, *Dishes*, *Dog* and *Speech* made up almost 90% of the false positives predicted when models were tested on a purely non-target dataset. In this experiment the findings were showing a very different tendency, where the long audio event classes (*Frying* etc.) made up over 90% of the system detections. The logical explanation is that the non-target dataset used in this experiment consists of mainly long audio events (vehicles, wind, waves, industry), indicating that the tendency is mainly due to the dataset.

Model	Events detected	Long	Short
RAW2D	915 events	862	53
LOG2D	1451 events	1326	125

Table 5.9: Number of events predicted on 100 minutes of Non-target audio

5.2 Analysis

5.2.1 Overall performance

The winning submission of DCASE 2022 challenge Task 4 (Ebbers & Haeb-Umbach, 2022) got a PSDS score (The PSDS 2 in DCASE) of 0.844 and a F1 event score of 59.8% on the public evaluation set. The baseline score to beat for 2022 was a PSDS of 0.536 and F1 event score of 40.1%. In comparison it is obvious that the models trained for the experiments in this thesis are not even close to reaching the same performance levels. Ebbers et al.'s system consisted of an ensemble model of 40 CRNNs and an additional 20 BiCRNNs to aid in the segmentation task. Each of the CRNNs used were significantly bigger than the models trained for this thesis, each consisting of 16 2D convolutional layers and two BiGRU layers with 256 hidden units each. These models were also trained on a much larger dataset, as weakly labelled data was used to great extent. Designing and training such a state-of-the-art SED system is clearly beyond the limitations set by the GPU resources and time available for this thesis.

It is also clear that the segmentation task is a difficult aspect of a SED system, mirrored in the low F1 event scores of the models. Modern day SED systems routinely uses separate models which are just trained to do the segmentation based on the output of a model responsible for tagging, ie detect the presence of a target sound event somewhere in the audio file.

5.2.2 1D CNN versus 2D CNN

Overall, the models using 2D convolutional layers in the CNN block do better than the models using 1D convolutional layers. It does seem like doing dimensionality reduction over the frequency axis is sufficient for the 2D CNN to extract high level features, and the 1D CNNs struggle because they are not allowed to do dimensionality reduction over the temporal axis. However, the LAR block itself is a pure 1D CNN, and is capable of both learning a meaningful representation of the data and to do a data transformation of the input signal into a 2D representation that can be understood by the 2D CNN block.

The LAR block does perform dimensionality reduction of the signal from it's original length to the number of analysis frames. In PURE1D the dimensionality reduction from original length in samples to the number of analysis frames is spread through the whole network, and although its performance is the worst for the F_{size} 256 models, it does seem to be less affected by the uneven class distribution. This might indicate that it is able to learn high level features more capable of generalising than RAW2D. The same tendency can be seen in F_{size} 64 experiment, where RAW1D scored the most even performance among the classes. Quite possibly the high level learned features of a 1D CNN acquired from the time-domain signal are more general and less specific for each class than 2D CNNs.

The dimensionality reduction in the LAR block is done in one operation in the maxpooling layer, which takes the maximum value of each filter over the F_{size}. The maxpooling layer in the models trained on F_{size} 256 therefore does a much more drastic dimensionality reduction (40906 to 160) than the models trained on F_{size} 64 (40960 to 640). However this does not seem to affect the performance, rather the F_{size} 256 improves compared to the F_{size} 64 models. This might be due to the fact that each analysis frame contains a learned representation of a bigger section of the original signal, which in turn benefits the high-level representation learning in the following CNN block. For

future work, LAR block designs which stacks multiple blocks each performing a smaller dimensionality reduction would have to be attempted in order to see whether this would improve the ability to generalise even on an unevenly distributed dataset.

It was clear from the results of the 70 epochs experiment that RAW2D needed more time to train than LOG2D. Presumably this would also hold true for RAW1D and PURE1D, which could both be severely at a disadvantage after just 40 epochs. All the models working with raw audio as input can therefore be thought of as potentially better than the scores obtained in this experiment, given more time to train. LOG2D suffered from additional training, which can indicate a weakness of using log mel-spectrograms as input for complex tasks.

5.2.3 LAR block analysis

The design of the LAR block allows us to analyse the learned audio representation, and the filter responses of the convolutional filters in the layers. The output of each filter can be visualised as a log mel-spectrogram, in order to get an idea of what the different filters in the LAR block layers are responsible for. Once the maxpooling of the LAR block has been done, the output will no longer be meaningful to a human observer, but we can still visualise it and make educated guesses on how the learned audio representation differs when presented with audio representing the different target sound classes. Figure 5.4 shows an example of the audio transformation happening inside the LAR block. 8 out of 128 filters have been selected for simplicity. We can clearly see different frequency regions being emphasised by the different filters. Rather counter intuitively we have to visualise the input and the output of the two convolutional layers as log mel-spectrograms, but it should be duly noted that this example is taken from a model (RAW2D trained for 70 epochs) which works purely on raw audio, so the log mel-spectrograms are created by taking the STFT of the output of each filter only for visualisation purposes. Filter responses for the eight filters from the first convolutional layer can be seen in Figure 5.5. The filter responses show quite complex and erratic filters, however they are all clearly more focused on specific frequency regions and resemble either lowpass, hipass or bandpass filters.

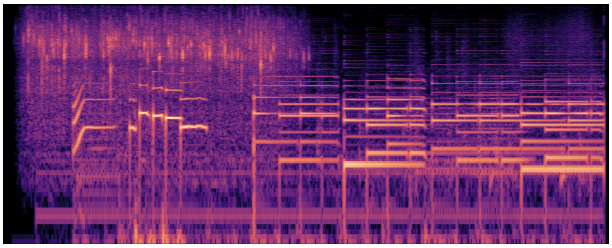
Figures 5.6 and 5.7 show the learned audio representation and the system detections for segments containing respectively *speech* and *running water*. The speech segment also contains two events of the class *electric shaver toothbrush* which are undetected by the system. However the difference between the speech event and the electric shaver event can clearly be seen in the LAR block output.

An interesting observation is how the LAR block output seems to be a good noise canceler, as can be seen between sound events in Figure 5.6. If we study the LAR block representation of the onset and offset of the speech event, there might be an issue that it captures onsets too slow.

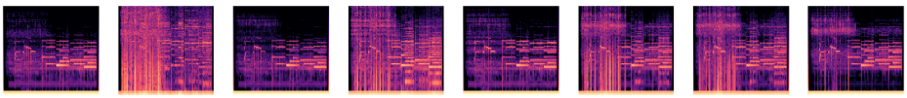
The log mel-spectrogram representation of the running water event in Figure 5.7 is saturated with noise, and to human eyes it is hard to argue that this is a meaningful representation of the audio event. The LAR block output makes no sense to us, but we can still compare the two and make an educated guess that the LAR block output can be just as meaningful as the log mel-spectrogram representation of this sound event.

5.2.4 Operating point tuning

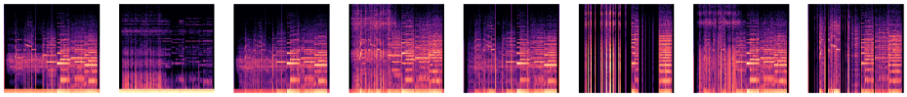
Since the PSDS evaluation calculates performance over a range of 50 operating points (detection thresholds), we can find the operating point for each class that results in



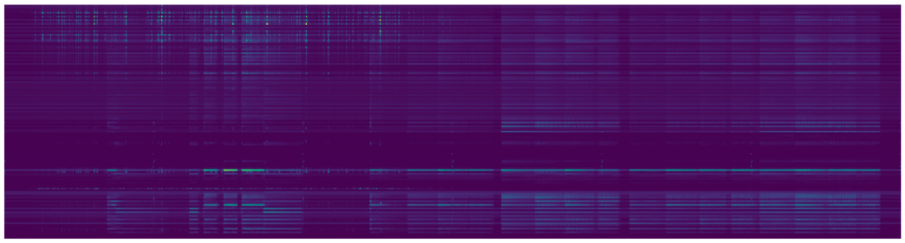
(a) Audio input



(b) Output of first convolutional layer



(c) Output of second convolutional layer



(d) Final output of LAR block

Figure 5.4: Visual representation of output of 8 selected filters and the final output of the LAR block on 10 seconds of evaluation audio. The depthwise filters of the second layer are the corresponding filters to those selected for the first layer. The target class frying is heard in the first 4 seconds of the segment, and guitar playing is audible through the whole segment.

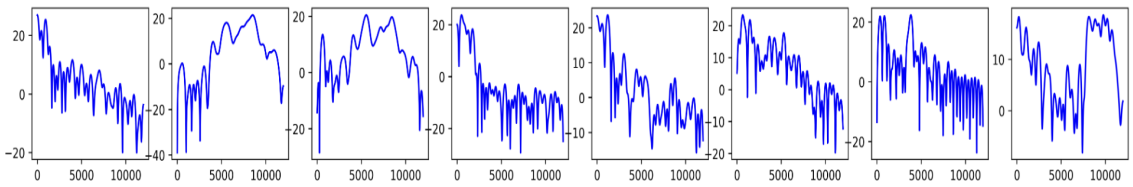


Figure 5.5: Filter response of filters from first convolutional layer seen in Figure 5.4

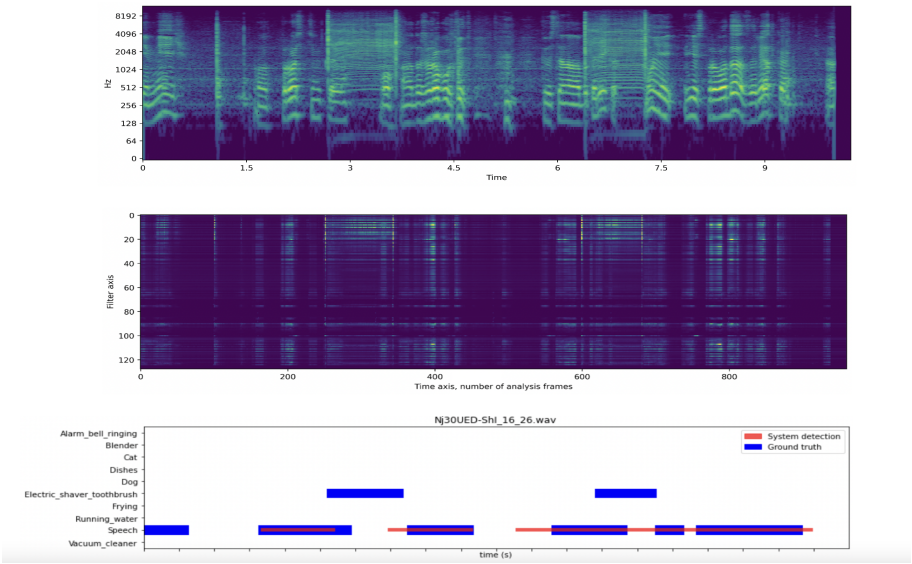


Figure 5.6: Audio containing *speech* and *electric shaver toothbrush*. Top to bottom: Log mel-spectrogram of input evaluation audio, LAR block output, system detection and ground truth. Note the noise canceling of the LAR block output between frames 400 and 600.

the highest class F1 score. By summing the F1 class scores of each class given its optimal operating point, the Macro Average F1 score for an optimally tuned system is given. Table 5.10 displays the Macro Average F1 score after each class has been tuned to its optimal operating point, and all models improve significantly. This indicates the importance of post-processing in a SED system.

F_{size} 64		
Model name	Mac. Avg. F1	Difference
LOG2D	37.7	+10.0
RAW2D	41.1	+ 9.1
LOG1D	32.3	+4.8
RAW1D	35.4	+6.0
F_{size} 256		
LOG2D 70 eps	45.8	+9.0
RAW2D 70 eps	47.1	**+11.7**
LOG2D	**48.1**	+11.0
RAW2D	40.7	+10.0
PURE1D	36.0	+6.1

Table 5.10: Macro Average F1 score for each model with optimal operating points for individual classes, and difference from original untuned model.

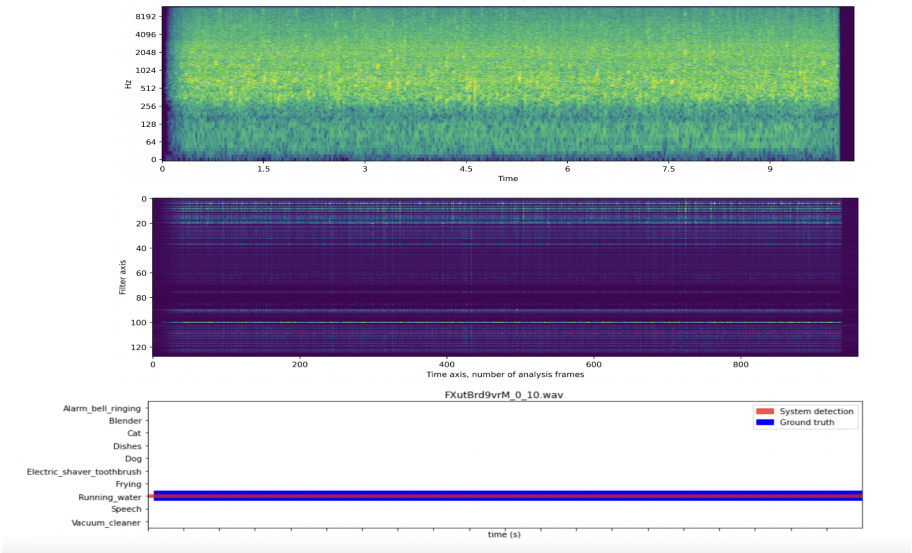


Figure 5.7: Audio containing *running water*. Top to bottom: Log mel-spectrogram of input evaluation audio, LAR block output, system detection and ground truth.

Chapter 6

Conclusions

Log mel-spectrograms have been prevalent as the preferred input to SED systems since the introduction of the CRNN. This thesis shows that possible architectures for working with raw audio and allowing the models to learn the audio representation in an end-to-end fashion can perform equally to and in some specific cases better than graphical time-frequency representations.

With respect to the research questions introduced in section 1.3.2, we conclude the following based on the experimental results detailed in Chapter 5.

RQ 1: How does the performance of a deep learning model that works strictly on raw audio as input compare to the performance of a baseline SED system using fixed graphical time-frequency representation of audio?

The findings prove that raw audio models can perform at the same level or better than models using fixed time-frequency representations. Raw audio end-to-end deep learning architectures for SED are harder to train both in GPU resources and time, as they have to learn to extract a meaningful representation of audio as well as solving the classification and segmentation task.

RQ 2: Does working with raw audio as input to a deep learning model provide any new insights into the challenges of SED?

The results indicate that raw audio end-to-end models can work on smaller analysis frames (2,6 ms) for detection than previously done, without losing too much performance. This can be a strategy in order to reduce the latency of a SED system, which can be crucial in f.ex. health applications and industrial applications. All the models struggle with classifying classes which are poorly represented in the training dataset. This is a known challenge in SED, however the RAW2D model did seem to struggle more than the other models with concern to this particular challenge.

RQ 3: Are there limitations in 2D Convolutional neural networks used in SED which can impact future general purpose machine listening systems? ?

The 2D CNNs used in SED with dimensionality reduction only over the frequency axis are more adept at learning a high-level feature representation that helps the classification process than 1D CNNs . However 1D CNNs did have the least spread among the F1 class score, indicating that the high-level features learned are less class-dependent. In

the context of future general purpose machine listening systems this might imply a better capability of learning general feature representations of audio than 2D CNNs. 1D CNNs would require more training and exploration in order to answer this question fully.

6.1 Contributions

The LAR block proposed in this thesis enables direct comparison between raw audio input and fixed graphical time-frequency representations, which has previously not been done in SED. As deep learning architectures become more and more complex, there is a tendency to design increasingly bigger models to beat state-of-the-art performance, especially in a field driven by competitions such as SED and computer vision. However, choosing to train smaller, directly comparable models in order to pose a fundamental question on the choice of input for SED systems should be a valuable contribution to the field.

Smaller, more compact model architectures are also necessary when implementing SED systems in edge devices like cell phones and smart home monitoring systems. 1D convolutional models using raw audio as input can be a promising strategy for achieving such models due to their computational efficiency.

Besides SED, the ability to directly process and analyse raw audio data could lead to advancements in various fields such as environmental monitoring, surveillance systems, acoustic scene analysis, and more.

6.2 Discussion

It is important to keep in mind the limitations of the SED systems trained for this thesis. They are trained on a very specific task, ie the DCASE dataset for domestic home monitoring. There are only 10 audio classes in the dataset, and even though 27.8 hours of strongly labelled audio can seem like a big dataset, it has been shown to be insufficient for training a modern SED system.

The model architectures might yield a different result when applied to other SED tasks, which makes knowledge of the task and domain knowledge in audio analysis important factors when planning a SED system. Hence, only further experimentation can determine how raw audio input compares to fixed graphical time-frequency representations in other applications.

The findings do however show a clear tendency towards better performance with raw audio input for smaller analysis time frames, and this is expected to translate into other audio analysis tasks. Applied in the time domain, we can expect raw audio input to improve a system's ability to learn amplitude envelopes and transients, as this information requires high time resolution which is subject to a trade off with frequency resolution in a graphical time-frequency representation.

The trainable LAR block as this thesis proposes also enables dynamic frequency resolution. For specific tasks where high frequency resolution in certain registers are required, it would seem probable that raw audio input and a learned audio representation can benefit the system, as the learned audio representation output of the LAR block is not subject to the set frequency resolution of the STFT.

The combination of dynamic frequency resolution and the ability to learn amplitude envelopes and transients with precision is the main rationale for why raw audio input can be beneficial to audio analysis. The trade-off between frequency resolution and temporal resolution in the fixed graphical time-frequency representation is unnecessary

when working strictly in the time domain. However, to explore this hypothesis further we need far bigger datasets with much more variation in the sound events than the dataset used for this thesis' experiments.

In terms of a general purpose machine listening system, the findings of this thesis are too limited in scope to make any predictions. RAW2D was clearly the most task-specific of all the models, while RAW1D was the least task-specific in terms of spread in the class F1 score. There are many other challenges to designing general purpose machine listening systems which have yet to be solved, and the choice of input will probably be decided based on the size of the input data, favouring fixed graphical time-frequency representations as the best input candidate.

LOG2D dropped in performance between 40 epochs and 70 epochs. The regularisation applied in the model with a dropout of 0.5 for every layer was not sufficient to prevent overfitting. If this tendency holds true in further experiments, it must be considered a drawback when using log mel-spectrograms. It could be due to the logarithmic nature of the log mel-spectrograms, where the frequency information in the high registers are too condensed to provide enough information, limiting the model to learn high-level feature representation of a smaller frequency region.

The LOG2D trained for 40 epochs, which had not yet overfitted, greatly benefited when evaluated on the high pass filtered dataset. One can therefore assume that many of the high-level feature representations applied in the low frequency regions caused errors in the original evaluation. The high-level features of a CNN are translation invariant, and when applying local patterns learned in the middle or upper frequency regions on the lower frequency regions the logarithmically scaled nature of the log mel-spectrograms could cause further errors. It should be noted that also RAW2D improved almost as much on the high pass filtered dataset evaluation, therefore one should be careful to blame the log mel-spectrograms. Designing experiments which provide information on what is happening inside deep learning models are complex, and to really test whether the translation invariance of 2D CNNs causes errors in audio analysis tasks, further experiments must be considered.

6.3 Future work

The time frame and computational resources available limited some obvious improvements in the experimental design. The following items are proposed as future work:

- Train all models for up to 100 epochs with decreasing learning rate and early stopping schedule
- Training RAW1D and LOG2D on F_{size} 256
- Training PURE1D on F_{size} 64

The LAR block design proposed in this thesis is only one of many possible deep learning architectures for raw audio input. It can even be argued that it is quite small, and a larger LAR block would have given the model a bigger hypothesis space. For future work we propose to experiment with bigger or more sophisticated architectures to explore the possibilities of the LAR block as discussed in section 5.2.2.

The models using the LAR block in combination with 2D CNNs are clearly struggling with the classes which are less represented in the training data. We might ask if the task of learning the audible representation makes the RAW2D models more susceptible to unbalanced class distribution in training data, and an interesting experiment for future

work would be to train for a set number of epochs on a smaller, less unbalanced dataset before introducing the full dataset. Another possible solution would be to construct an encoder-decoder model with the LAR block as encoder, and pre-train the weights in the LAR block. The rationale being that it is then trained not in order to assist the classification process, but purely to create a meaningful representation of all audio.

Bibliography

Abdoli, S., Cardinal, P., & Koerich, A. L. (2019). End-to-end environmental sound classification using a 1d convolutional neural network. *Expert Systems with Applications*, *136*, 252–263.

Adavanne, S., Pertilä, P., & Virtanen, T. (2017). Sound event detection using spatial features and convolutional recurrent neural network. *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 771–775.

Adavanne, S., Politis, A., Nikunen, J., & Virtanen, T. (2019). Sound Event Localization and Detection of Overlapping Sources Using Convolutional Recurrent Neural Networks. *IEEE Journal of Selected Topics in Signal Processing*, *13*(1), 34–48. https://ieeexplore.ieee.org/document/8567942/

Bilen, Ç., Ferroni, G., Tuveri, F., Azcarreta, J., & Krstulović, S. (2020). A framework for the robust evaluation of sound event detection. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 61–65.

Çakir, E., Parascandolo, G., Heittola, T., Huttunen, H., & Virtanen, T. (2017). Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *25*(6), 1291–1303.

Çakir, E., & Virtanen, T. (2018). End-to-end polyphonic sound event detection using convolutional recurrent neural networks with learned time-frequency representation input. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–7.

Chan, T. K., & Chin, C. S. (2020). A comprehensive review of polyphonic sound event detection. *IEEE Access*, *8*, 103339–103373.

Chen, H., Zhang, P., Bai, H., Yuan, Q., Bao, X., & Yan, Y. (2018). Deep convolutional neural network with scalogram for audio scene modeling. *Interspeech*, 3304–3308.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). Convolutional recurrent neural networks for music classification. *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, 2392–2396.

Chollet, F. (2021). *Deep learning with python*. Simon; Schuster.

Copiaco, A., Ritz, C., Abdulaziz, N., & Fasciani, S. (2021). A study of features and deep neural network architectures and hyper-parameters for domestic audio classification. *Applied Sciences*, *11*(11), 4880.

Dai, W., Dai, C., Qu, S., Li, J., & Das, S. (2017). Very deep convolutional neural networks for raw waveforms. *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 421–425.

Dekkers, G., Lauwereins, S., Thoen, B., Adhana, M. W., Brouckxon, H., Van den Bergh, B., Van Waterschoot, T., Vanrumste, B., Verhelst, M., & Karsmakers, P. (2017). The sins database for detection of daily activities in a home environment using an acoustic sensor network. *Detection and Classification of Acoustic Scenes and Events 2017*, 1–5.

Ebbers, J., & Haeb-Umbach, R. (2022). *Pre-training and self-training for sound event detection in domestic environments* (tech. rep.). Paderborn University, Tech. Rep.

Fant, G. (1968). Analysis and synthesis of speech processes. *Manual of phonetics*, *2*, 173–277.

Fonseca, E., Pons Puig, J., Favory, X., Font Corbera, F., Bogdanov, D., Ferraro, A., Oramas, S., Porter, A., & Serra, X. (2017). Freesound datasets: A platform for the creation of open audio datasets. *Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. Proceedings of the 18th ISMIR Conference; 2017 oct 23-27; Suzhou, China.[Canada]: International Society for Music Information Retrieval; 2017. p. 486-93.*

Frantzolas, T. (2009). *Everything you hear on film is a lie*. TED. https://www.ted.com/talks/tasos_frantzolas_everything_you_hear_on_film_is_a_lie

Gaver, W. W. (1993). How do we hear in the world? explorations in ecological acoustics. *Ecological psychology*, *5*(4), 285–313.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323.

He, Y., Trigoni, N., & Markham, A. (2021). Sounddet: Polyphonic moving sound event detection and localization from raw waveform. *International Conference on Machine Learning*, 4160–4170.

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. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, *29*(6), 82–97.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Kim, N. K., & Kim, H. K. (2021). Self-training with noisy student model and semi-supervised loss function for dcase 2021 challenge task 4. *arXiv preprint arXiv:2107.02569*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, *151*, 107398.

Kiranyaz, S., Ince, T., Hamila, R., & Gabbouj, M. (2015). Convolutional neural networks for patient-specific ecg classification. *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2608–2611.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, *2*.

Lin, L., Wang, X., Liu, H., & Qian, Y. (2019). Guided learning convolution system for dcase 2019 task 4. *arXiv preprint arXiv:1909.06178*.

Mesaros, A., Diment, A., Elizalde, B., Heittola, T., Vincent, E., Raj, B., & Virtanen, T. (2019). Sound event detection in the dcase 2017 challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *27*(6), 992–1006.

Mesaros, A., Heittola, T., & Virtanen, T. (2016a). Metrics for polyphonic sound event detection. *Applied Sciences*, *6*(6), 162.

Mesaros, A., Heittola, T., & Virtanen, T. (2016b). Tut database for acoustic scene classification and sound event detection. *2016 24th European Signal Processing Conference (EUSIPCO)*, 1128–1132.

Mesaros, A., Heittola, T., & Virtanen, T. (2018). Acoustic scene classification: An overview of dcase 2017 challenge entries. *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 411–415.

Mesaros, A., Heittola, T., Virtanen, T., & Plumbley, M. D. (2021). Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, *38*(5), 67–83.

Mitchell, T. M., et al. (2007). *Machine learning* (Vol. 1). McGraw-hill New York.

Miyazaki, K., Komatsu, T., Hayashi, T., Watanabe, S., Toda, T., & Takeda, K. (2020). Weakly-supervised sound event detection with self-attention. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 66–70.

Nam, H., Kim, S.-H., Ko, B.-Y., & Park, Y.-H. (2022). Frequency dynamic convolution: Frequency-adaptive pattern recognition for sound event detection. *arXiv preprint arXiv:2203.15296*.

Natsiou, A., & O'Leary, S. (2022). Audio representations for deep learning in sound synthesis: A review [arXiv:2201.02490 [cs, eess]].

Olah, C. (2019). *Understanding lstm networks* [Accessed: 2022-04-21]. http://https://colah.github.io/posts/2015-08-Understanding-LSTMs

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Parascandolo, G., Huttunen, H., & Virtanen, T. (2016). Recurrent neural networks for polyphonic sound event detection in real life recordings. *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 6440–6444.

Pedersen, P. (1965). The mel scale. *Journal of Music Theory*, *9*(2), 295–308.

Plack, C. J. (2018). *The sense of hearing*. Routledge.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*.

Ramirez, M. A. M., & Reiss, J. D. (2019). Modeling nonlinear audio effects with end-to-end deep neural networks. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 171–175.

Ronchini, F., & Serizel, R. (2022). A benchmark of state-of-the-art sound event detection systems evaluated on synthetic soundscapes. *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1031–1035.

Ronchini, F., Serizel, R., Turpault, N., & Cornell, S. (2021). The impact of non-target events in synthetic soundscapes for sound event detection. *arXiv preprint arXiv:2109.14061*.

Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 4580–4584.

Salamon, J., MacConnell, D., Cartwright, M., Li, P., & Bello, J. P. (2017). Scaper: A library for soundscape synthesis and augmentation. *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 344–348.

Serizel, R., Turpault, N., Eghbal-Zadeh, H., & Shah, A. P. (2018). Large-scale weakly labeled semi-supervised sound event detection in domestic environments. *arXiv preprint arXiv:1807.10501*.

Serizel, R., Turpault, N., Shah, A., & Salamon, J. (2020). Sound event detection in synthetic domestic environments. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 86–90.

Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, *37*(1), 10–21.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. *Proceedings of the AAAI conference on artificial intelligence*, *31*(1).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vecchiotti, P., Ma, N., Squartini, S., & Brown, G. J. (2019). End-to-end binaural sound localisation from the raw waveform. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 451–455.

Virtanen, T., Plumbley, M. D., & Ellis, D. (2018). *Computational analysis of sound scenes and events*. Springer.

Wyse, L. (2017). Audio spectrogram representations for processing with convolutional neural networks. *arXiv preprint arXiv:1706.09559*.