2022

Deep Learning for Audio Segmentation and Intelligent Remixing

Venkatesh, Satvik

UNIVERSITY OF
PLYMOUTH

DEEP LEARNING FOR AUDIO SEGMENTATION AND INTELLIGENT REMIXING

by

SATVIK VENKATESH

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Society and Culture

December 2022

I dedicate this thesis to my grandmother, Devika Rajendran, who sadly passed during my time as a PhD student.

Acknowledgements

This PhD project has been a long and beautiful journey. Walking this road would have been impossible without the constant motivation of people around me. It has taught me the importance of gratitude and humility

I would not have had the opportunity to work on this project without my Director of Studies, Prof. Eduardo Reck Miranda. Working with you has helped me build a strong foundation for my career as a researcher. Your supervision has been invaluable to me during my undergraduate, master's and PhD degrees. I express my sincere gratitude to my second supervisor, Dr. David Moffat, who has taught me a great deal about artificial intelligence, machine learning, and signal processing. Our weekly meetings (in most cases twice a week) on machine learning and the development of the project made my PhD research more enjoyable. It is remarkable how I learned so much more from regular discussions and feedback, instead of being locked with ideas in my own mind. I also thank Dr. David Jenkins who agreed to be my second supervisor in the last year of my PhD, during the crucial phase of writing-up. Also, thank you for initially liaising with SASTRA University for me to pursue a semester exchange programme at the University of Plymouth.

I also had the opportunity to pursue a research internship at Mitsubishi Electric Research Laboratories (MERL) during my degree. I sincerely thank Dr. Gordon Wichern, Dr. Jonathan Le Roux, and Dr. Aswin Subramanian to work with me during my tenure. It was an amazing experience and a learning curve to work on their cutting-edge technology.

I thank Dr. Edward Braund and Dr. Nuria Bonet for their advice during my time as an associate lecturer for the BSc programme in Computing and Music Technology. Dr. Edward Braund's supervision during my master's degree has greatly helped me develop research and writing skills. I thank the amazing friends I have made at ICCMR — Ben, Rachel, Jared, Sam, Clive, and Omar. Clive was kind enough to verify the annotations of audio files used for my research.

I sincerely thank my parents and family back in India, who have showered me continuous love and support during my degree. It was helpful to pursue some part of my degree remotely from India and survive the hardships caused by COVID. They patiently listened to and motivated me while I kept complaining about my experiments not working. I am grateful to my fiancé Aishwarya for being such a great team player in the process of me being a PhD student. Thank you for patiently listening to and helping me deal with my research problems, and for tolerating my project-obsession. I also sincerely thank all my friends and relatives back in India for their continuous love and support.

This project was supported by the EPSRC grant EP/S026991/1. I also thank Sanmat and Adar poonawalla for their scholarship grant in September 2021 to fund my quarantine and COVID-related costs.

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee.

Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

Relevant scientific seminars and conferences were regularly attended at which work was often presented. A list of publications can be found in section 1.6.

Word count for the main body of this thesis: **44,470**

Signed: 

Date: 16-12-2022

Abstract

Name: Satvik Venkatesh

Title: Deep Learning for Audio Segmentation and Intelligent Remixing

Audio segmentation divides an audio signal into homogenous sections such as music and speech. It is useful as a preprocessing step to index, store, and modify audio recordings, radio broadcasts and TV programmes. Machine learning models for audio segmentation are generally trained on copyrighted material, which cannot be shared across research groups. Furthermore, annotating these datasets is a time-consuming and expensive task. In this thesis, we present a novel approach that artificially synthesises data that resembles radio signals. We replicate the workflow of a radio DJ in mixing audio and investigate parameters like fade curves and audio ducking. Using this approach, we obtained state-of-the-art performance for music-speech detection on in-house and public datasets.

After demonstrating the efficacy of training set synthesis, we investigate how audio ducking of background music impacts the precision and recall of the machine learning algorithm. Interestingly, we observed that the minimum level of audio ducking preferred by the machine learning algorithm was similar to that of human listeners. Furthermore, we observe that our proposed synthesis technique outperforms real-world data in some cases and serves as a promising alternative.

This project also proposes a novel deep learning system called You Only Hear Once (YOHO), which is inspired by the YOLO algorithm popularly adopted in Computer Vision. We convert the detection of acoustic boundaries into a regres-

sion problem instead of frame-based classification. The relative improvement for F-measure of YOHO, compared to the state-of-the-art Convolutional Recurrent Neural Network, ranged from 1% to 6% across multiple datasets. As YOHO predicts acoustic boundaries directly, the speed of inference and post-processing steps are 6 times faster than frame-based classification. Furthermore, we investigate domain generalisation methods such as transfer learning and adversarial training. We demonstrated that these methods helped our algorithm perform better in unseen domains.

In addition to audio segmentation, another objective of this project is to explore real-time radio remixing. This is a step towards building a customised radio and consequently, integrating it with the schedule of the listener. The system would remix music from the user's personal playlist and play snippets of diary reminders at appropriate transition points. The intelligent remixing is governed by the underlying audio segmentation and other deep learning methods. We also explore how individuals can communicate with intelligent mixing systems through non-technical language. We demonstrated that word embeddings help in understanding representations of semantic descriptors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Audio Segmentation is a field of study, in where the aim is to find changing points in the content of an audio stream (Theodorou et al., 2014). It divides the audio into segments such that each segment belongs to a specific acoustic class. It is useful for indexing audio archives and target-based distribution of media. It also serves as a pre-processing step for tasks such as speech recognition, where regions of speech activity are detected before converting the audio to text. This project investigates audio segmentation for the novel task of intelligently remixing radio signals.

This thesis is a part of a bigger project called RadioMe[1]. RadioMe is an EPSRC-funded project, which aims to performs real-time radio remixing for people with dementia to reduce agitation and provide them with personalised diary reminders. In order to remix radio signals in real-time, the content of radio needs to be understood. In other words, audio segmentation needs to be performed to identify optimal points for remixing. Therefore, this thesis investigates algorithms to detect regions of Speech and Music in an audio stream.

[1]RadioMe is an EPSRC-funded project (grant EP/S026991/1) involving **academic partners**: University of Plymouth, University of Glasgow, University of Sussex, **industrial partners**: Bauer Media, BBC, and CereProc Limited and **charities**: Alzheimer's Society, MHA Care Group, Sussex Partnership, and NHS Foundation Trust.

There is a growing interest to use machine learning and deep learning for audio content-retrieval tasks such as audio segmentation. Over the years, the accuracy of segmentation algorithms have increased due to the improvement of machine learning architectures and the availability of more training data. This opens up opportunities to improve state-of-the-art algorithms by developing novel approaches. Often, an improvement in accuracy comes with an increase in computational cost. As this thesis remixes live radio streams, it is also interesting to develop algorithms that are faster and require less computational resources.

1.2 Project Aims

Machine Learning models for audio segmentation are generally trained using proprietary audio such as television and radio broadcast. This imposes a serious hindrance in the reproducibility of research because this audio cannot be shared across different research groups. Moreover, annotating these datasets is a time-consuming and expensive task. In this project, we explore the novel idea of artificially synthesising audio that resembles a radio broadcast. By doing so, we can synthesise large training sets for deep neural networks.

This project investigates how state-of-the-art algorithms for audio segmentation can be advanced. First, we compare the existing neural network architectures in the literature to find out which ones perform best. Second, we propose novel architectures to improve the accuracy and speed of machine learning models. Third, we investigate how the model can perform on multiple and unseen domains.

Moreover, in this project, we explore how audio segmentation can improve existing real-time remixing approaches. We try to integrate functions such as playing diary reminders and music from the listener's playlist seamlessly into the live radio stream. Lastly, we investigate how individuals can communicate with intelligent mixing systems with non-technical language.

The project aims are summarised as follows:

1. Improve the reproducibility of audio segmentation research through artificial data synthesis and data augmentation techniques.

2. Advance the state-of-the-art algorithms for audio segmentation and classification by designing and optimising neural network architectures.

3. Develop a system for real-time remixing of radio by detecting appropriate transition points.

1.3 Research Questions

1. What would be an effective way to train machine learning models for audio segmentation with limited access to domain data?

2. How can we advance state-of-the-art algorithms by investigating novel pattern recognition problems?

3. To what extent can deep learning be used to improve intelligent mixing approaches?

1.4 Methods

Below is an overview of the methods adopted to render research in audio segmentation and intelligent remixing. More details on specific methods are detailed where appropriate in the text.

1.4.1 Python

Python 3 was used for all experiments in this thesis. Over the years, it has gained tremendous popularity within the community for Music Information Retrieval (MIR), speech recognition, and sound event detection. Sometimes researchers make their source code openly available along with their article, which assists reproducibility and speeds up implementation. Audio transformations such as

shortening silences in audio were performed with the help of Sound eXchange (SoX)[2]. The Librosa package (McFee et al., 2015) was used to extract audio features such as short-time Fourier transform (STFT), power spectrum, mel spectrograms, and so on. Librosa was also adopted for operations like adjusting the sampling rate of audio. The SoundFile[3] package was used to load audio files in the program. To visualise experimental results, graphs were plotted with the help of Matplotlib[4].

As this project also addresses intelligent remixing of live radio, a multi-threaded audio application was developed with the help of Pyaudio[5]. Internet radio stream was accessed by the Python application through FFmpeg[6].

1.4.2 Deep Learning

Deep learning architectures in this thesis were implemented by using TensorFlow 2 / Keras. Keras is a higher-level application programming interface (API) to easily build large neural networks. It provides a framework that enables researchers to iterate their ideas quickly. It provides support to create fully connected, convolutional, softmax, and long short-term memory (LSTM) layers, to name a few. It also has integrated support to perform backpropagation for training neural networks.

Deep learning research requires extensive resources due to large datasets and networks. Therefore, training of neural networks is performed with the help of Graphical Processing Units (GPUs) to parallelise and speed up the process. Research groups often use High-Performance Computing (HPC) clusters to facilitate deep learning research. GPUs are installed within these clusters which are used for computation. As we did not have access to such computing resources at the time of my PhD, I used Google Colab[7], which is a free service provided by Google. It allows individuals to use GPUs like K80, T4, and P100 on Virtual Machines. The

[2]SoX: http://sox.sourceforge.net/
[3]SoundFile: https://pypi.org/project/SoundFile/
[4]Matplotlib: https://matplotlib.org/
[5]Pyaudio: https://pypi.org/project/PyAudio/
[6]FFmpeg: https://ffmpeg.org/
[7]Google Colab: https://colab.research.google.com/

datasets for training networks and experimental data were transferred between the Colab Virtual Machine and Google Drive. I wrote a blog post that explains the various optimisations I developed to easily transfer data between Colab and Google Drive with minimal latency (Venkatesh, 2021).

1.4.3 Literature Sources

In this section, I outline the important competitions, conferences, and journals that are relevant to my PhD project. However, please note that this list is not exhaustive. The Music Information Retrieval Evaluation eXchange (MIREX) in 2018, hosted a competition for music-speech detection in broadcast audio. The task had two test sets — (1) 27 hours of audio from 8 different TV program types from different countries and (2) 10 hours of audio corresponding to French TV and radio programs. The annual MIREX competition is coupled with the International Society for Music Information Retrieval (ISMIR) conference, whose proceedings comprise key papers cited by this thesis (Schlüter & Grill, 2015; Stoller et al., 2018; Lemaire & Holzapfel, 2019).

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) is an annual conference that focuses on many audio and acoustic signal processing applications. Within the conference, there is also a sub-category titled 'Detection and Classification of Acoustic Scenes and Events', which is relevant to this thesis. There is also an annual challenge hosted by IEEE AASP called the Detection and Classification of Acoustic Scenes and Events (popularly known as DCASE). Some important tasks within the challenge include sound event detection in domestic environments, acoustic scene classification, and unsupervised anomaly detection.

Journals relevant to audio segmentation include IEEE/ACM Transactions on Audio, Speech, and Language Processing, EURASIP Journal on Audio, Speech, and Music Processing, and Applied Sciences. As this thesis also focuses on intelligent remixing of radio, useful sources were found in the Journal of the Audio Engineering Society and the Digital Audio Effects (DAFx) conference. As a part of training,

I completed two online courses offered by Coursera, titled *Machine Learning*[8] and *Deep Learning*[9].

1.5 Contributions

Below is a brief summary of contributions from this thesis:

- In chapter 3, we answer the first research question — *"What would be an effective way to train machine learning models for audio segmentation with limited access to domain data?"*. We proposed a novel data synthesis procedure to generate large-scale training sets for audio segmentation. We replicate the workflow of an audio mixing engineer to automatically create radio-like material. Using this technique, we obtained state-of-the-art performance on music-speech detection on in-house and public datasets. We also investigate the effects of training set synthesis and compare real-world and synthetic training sets.

- The findings in chapter 4 addressed the second research question — *"How can we advance state-of-the-art algorithms by investigating novel pattern recognition problems?"*. Here, we proposed a novel algorithm called You Only Hear Once (YOHO) that converts sound event detection from a classification to a regression problem. It is inspired by the famous You Only Look Once (YOLO) algorithm in Computer Vision. The relative improvement for F-measure of YOHO, compared to the state-of-the-art Convolutional Recurrent Neural Network, ranged from 1% to 6% across multiple datasets for audio segmentation and sound event detection. As the output of YOHO is more end-to-end and has fewer neurons to predict, the speed of inference is at least 6 times faster than segmentation-by-classification. In addition, as this approach predicts acoustic boundaries directly, the post-processing and smoothing are about 7 times faster.

[8]Machine Learning: https://www.coursera.org/learn/machine-learning
[9]Deep Learning: https://www.coursera.org/specializations/deep-learning

- In chapter 5, we investigated how machine learning models would generalise to unseen domains. We demonstrated how techniques like transfer learning and domain-adversarial training can be applied to achieve domain generalisation in audio segmentation systems. We obtained state-of-the-art performance with an improvement from 90.2% to 92.05% on the MIREX competition dataset. The conclusions in this chapter also contributed towards answering the second research question — *"How can we advance state-of-the-art algorithms by investigating novel pattern recognition problems?"*.

- In chapter 6, we explored the third research question — *"To what extent can deep learning be used to improve intelligent mixing approaches?"*. Here, we presented a real-time radio remixing system that incorporates diary reminders and songs from the listener's playlist. The remixing is governed by an audio segmentation algorithm running in real-time. Furthermore, we also demonstrated that word embeddings can be used to represent semantic descriptors for automatic EQ mixing. Using this technique, the machine learning model can generate EQ settings for words that it has not seen before. This way, people can communicate with intelligent mixing systems with non-technical language.

1.6 List of Publications

Below is a list of publications during my time as a PhD student. In cases where I am not the primary author, my contributions are detailed.

Chapter 3

- Venkatesh, S., Moffat, D., Kirke, A., Shakeri, G., Brewster, S., Fachner, J., Odell-Miller, H., Street, A., Farina, N., Banerjee, S., et al. (2021a). Artificially synthesising data for audio classification and segmentation to improve speech and music detection in radio broadcast. In *IEEE International Con-*

ference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Ontario, Canada, (pp. 636–640). doi: 10.1109/ICASSP39728.2021.9413597.

- Venkatesh, S., Moffat, D., & Miranda, E. R. (2021b). Investigating the effects of training set synthesis for audio segmentation of radio broadcast. *Electronics*, *10*(7), 827. doi: 10.3390/electronics10070827.

Chapter 4

- Venkatesh, S., Moffat, D., & Miranda, E. R. (2022b). You only hear once: a YOLO-like algorithm for audio segmentation and sound event detection. *Applied Sciences*, *12*(7), 3293. doi: 10.3390/app12073293.

Chapter 5

- Venkatesh, S., Wichern, G., Subramanian, A., & Le Roux, J. (2022c). Disentangled surrogate task learning for improved domain generalization in unsupervised anomalous sound detection. Tech. rep., Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge

Chapter 6

- Venkatesh, S., Moffat, D., & Miranda, E. R. (2022a). Word embeddings for automatic equalization in audio mixing. *Journal of the Audio Engineering Society*, *70*(9), 753–763. doi: 10.17743/jaes.2022.0047

- Di Campli San Vito, P., Brewster, S., Venkatesh, S., Miranda, E., Kirke, A., Moffat, D., Banerjee, S., Street, A., Fachner, J., & Odell-Miller, H. (2022). Radiome: Supporting individuals with dementia in their own home... and beyond? In *CHI Conference on Human Factors in Computing Systems (CHI '22) Workshop 32, New Orleans, Louisiana, USA, 30 Apr 2022*. doi: 10.36399/gla.pubs.267520
  – I was involved in developing machine learning for audio segmentation and intelligent remixing in the RadioMe system.

- Shakeri, G., Brewster, S., Venkatesh, S., Moffat, D., Kirke, A., Miranda, E., Banerjee, S., Street, A., Fachner, J., & Odell-Miller, H. (2021). Radiome: challenges during the development of a real time tool to support people with dementia. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 08–13, 2021, Yokohama, Japan*. doi: 10026.1/17584
  – I was involved in developing machine learning for audio segmentation and intelligent remixing in the RadioMe system.

- Venkatesh, S., Moffat, D., & Miranda, E. (2019). Radiome: Artificially intelligent radio for people with dementia. In *Proceedings of DMRN+14: Digital Music Research Network One-Day Workshop, London, UK*

1.7 Structure

Chapter 2: I present a survey of the audio segmentation literature. The chapter comprises relevant background information, explains the audio segmentation pipeline, types of segmentation algorithms, factors specific to machine learning, and metrics.

Chapter 3: We develop a method to synthesise large-scale training sets for audio segmentation. After demonstrating the robustness of the synthesis technique, we investigate how audio ducking of background music impacts the precision and recall of the machine learning algorithm. Lastly, we evaluate the effectiveness of synthesised, real-world, and combined approaches for training models, to understand if the synthetic data presents any additional value.

Chapter 4: First, we compare state-of-the-art neural network architectures to pick the best model for the task. Second, we investigate if training machine learning models directly on raw audio provides any benefit. Third, we propose a novel method called YOHO for sound event detection.

Chapter 5: We investigate principles such as domain adaptation and generalisation

for audio segmentation to explore how algorithms can generalise better to unseen domains. We demonstrated the effectiveness of two methods — transfer learning and adversarial training.

Chapter 6: In this chapter, we explore real-time radio remixing which is governed by underlying audio segmentation. We also investigate how individuals can communicate with intelligent mixing systems through non-technical language.

Chapter 7: This is the concluding chapter which comprises a reflection on all the research questions and potential avenues for future work.

Chapter 2

Background: Deep Learning for Audio Segmentation

In this chapter, we present an overview of the practices adopted in the audio segmentation literature, varying from the pipeline to the different machine learning algorithms adopted for the task. In recent years, many studies have demonstrated the effectiveness of deep learning for audio information retrieval tasks and therefore, we give special attention to these state-of-the-art practices. Many changes in approaches have occurred during the paradigm shift from traditional machine learning to deep learning. These changes include the feature extraction process, amount of training data, data augmentation, and other methods to minimise overfitting.

Audio segmentation is the process of dividing audio into homogeneous sections, such as music and speech. It identifies the presence of audio classes and their respective acoustic boundaries. Hence, it enables us to automatically extract metadata regarding an audio signal. Applications of segmentation include content-based audio retrieval, indexing audio archives, target-based distribution of media, and as a pre-processing step for speech recognition (Dhanalakshmi et al.,

2009; Gimeno et al., 2020).

The acoustic classes present in audio depend on the nature of the content. For example, in a radio programme, the important classes include Music, Speech, Silence, and Sound Effects, to name but a few (Theodorou et al., 2014; Chourdakis et al., 2019). This survey primarily focuses on segmentation of broadcast signals like radio and television. The structure of this chapter is as follows.

2.1. Audio Segmentation Pipeline
2.2. Audio Features
2.3. Types of Segmentation Algorithms
2.4. Machine Learning Algorithms
2.5. Post-processing / Smoothing
2.6. Comparison
2.7. Factors Specific to Machine Learning
2.8. Metrics
2.9. Discussion

2.1 Audio Segmentation Pipeline

Figure 2.1 illustrates an overview of the audio segmentation pipeline. The first step is to extract features from the audio signal. Some commonly adopted features are the mel spectrogram (Lemaire & Holzapfel, 2019) and Mel Frequency Cepstral Coefficients (MFCC) (Meléndez-Catalán et al., 2018). After extracting audio features, it goes through an algorithm that extracts segments of audio. This stage also detects the underlying audio classes within the segments. Subsequently, the post-processing stage carries out two important functions — (1) smoothe the output of the algorithm by eliminating spurious audio events, and (2) convert

the output into human-readable labels. The output labels would contain each audio class, with its respective temporal start and end time in the audio signal. An example of output labels is shown in Figure 2.1.

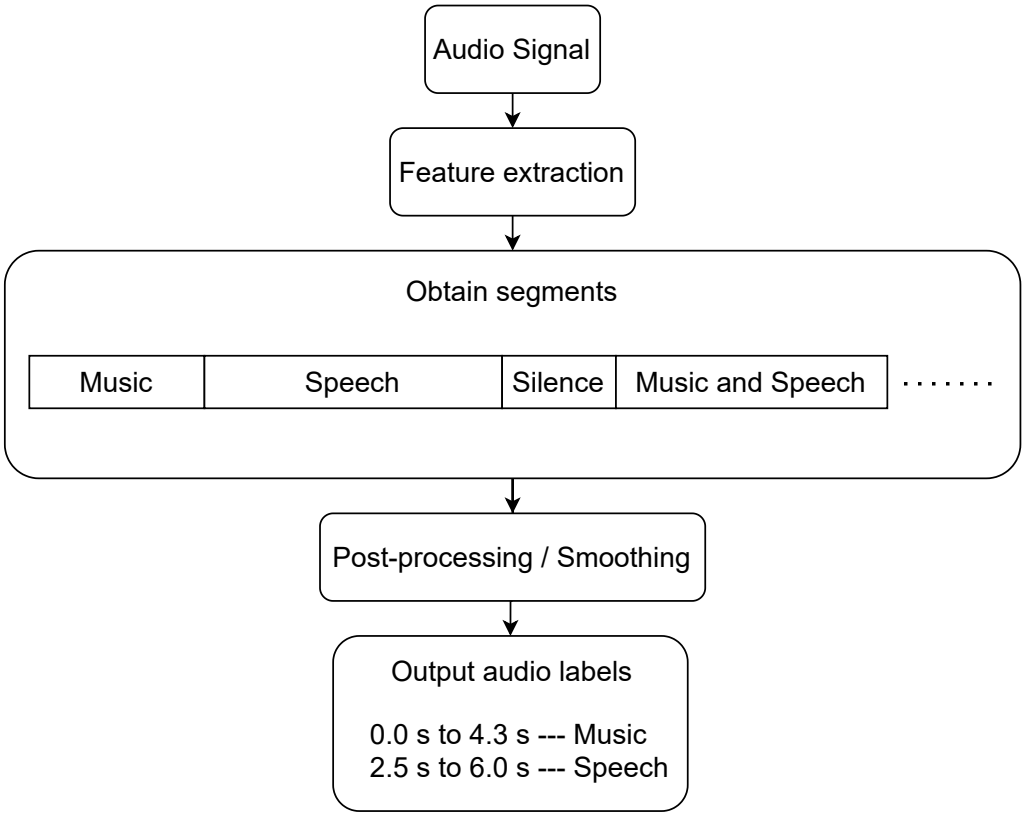


Figure 2.1: An illustration of the audio segmentation pipeline.

2.2 Audio Features

Historically, features for audio segmentation have generally been inspired by research in Speech Recognition (Butko & Nadeu, 2011). In most cases, audio is converted into a frequency-based representation through a Fourier transform. In addition to mel spectrogram and MFCC, some studies have also used perceptual linear prediction (PLP), chroma features (Gimeno et al., 2020), and frequency-filtered filter-bank energies (Butko & Nadeu, 2011). In recent years, there is also a growing interest in end-to-end deep learning, which directly processes raw audio (Lee et al., 2017). Amongst the choice of features, mel spectrogram and MFCC have been the most popularly adopted ones by studies (Butko & Nadeu, 2011;

Meléndez-Catalán et al., 2018; Lemaire & Holzapfel, 2019). A mel spectrogram is a 2D representation of audio that is optimised for human hearing (Choi et al., 2017a). In recent years, most deep learning studies for audio segmentation use mel spectrograms (Lemaire & Holzapfel, 2019; Gimeno et al., 2020). They have also been adopted for related tasks such as singing voice detection (Schlüter & Grill, 2015), speech recognition (Sainath et al., 2015a), and environmental sound detection (Salamon et al., 2017).

MFCC converts the mel spectrogram to a more compressed format by reducing it to a matrix of coefficients. Prior to the onset of deep learning, MFCCs were more commonly used with traditional machine learning algorithms. The reduced dimensionality of audio features helped the algorithms generalise better. However, deep learning models are trained on larger training sets and hence, benefit from using mel spectrograms directly.

Before extracting the mel spectrogram from an audio signal, we need to compute the Short-time Fourier transform (STFT). The hop size of the STFT defines the size of each time step in the spectrogram. A smaller hop size provides greater time-resolution for the STFT and vice-versa. In the audio segmentation literature, the hop size has ranged from 10 ms (Gimeno et al., 2020), 15 ms (Hussain et al., 2018) to 23 ms (Lemaire & Holzapfel, 2019). Many audio segmentation studies evaluate their algorithms on 10 ms segments. In other words, for every 10 ms of audio, it is checked whether the algorithm output the correct labels. For instance, to evaluate the music-speech detection algorithms at the Music Information Retrieval Evaluation eXchange (MIREX) 2015 and 2018, the organisers used a segment size of 10 ms. Therefore, in this thesis we adopted a hop size of 10 ms to ensure that the time-resolution of the audio segmentation algorithm is 10 ms.

The lowest detectable frequency in an audio signal depends on the window size of the STFT. A larger window size provides a greater context of audio at each time step. Therefore, a larger window size leads to better frequency resolution, but poorer time resolution (Cheuk et al., 2020; Wright, 1999). The window size in

audio segmentation studies has varied from 25 ms (Gimeno et al., 2020; Hussain et al., 2018) to 46 ms (Lemaire & Holzapfel, 2019). Studies on Music Information Retrieval (MIR) tasks such as chord recognition and genre classification have also used larger window sizes such as 93 ms and 185 ms (Jiang et al., 2011; Ravelli et al., 2010). For this thesis, to select an optimal window size, we performed informal tests with sizes of 23, 46, and 93 ms. We selected 46 ms as the window size as it performed slightly better than the other sizes.

Another specification of the mel spectrogram is the number of mel bands. The number of mel bands has generally been choices between 64, 80, and 96 (Lemaire & Holzapfel, 2019; Gimeno et al., 2020). Gimeno et al. (2020) found that using 80 mel bands was better than 64 and 96. In addition, they also showed that using chroma features along with the first and second-order derivatives improved the accuracy of the system, especially when Music belonged to the set of audio classes. Chroma features are optimised for music where the spectrum is projected into 12 bins, representing the semitones of the chromatic scale (Gimeno et al., 2020). For example, they are used for chord recognition (Jiang et al., 2011; Papadopoulos & Peeters, 2007) and identifying the chorus in songs (Bartsch & Wakefield, 2001).

Audio features such as mel spectrograms and MFCC discard the *phase* of audio signals and only consider the magnitude spectrum. As segmentation aims to detect acoustic boundaries, phase can contain important information. However, there has been less attention given to using phase for audio segmentation. This is due to the circular nature of phase, which is often challenging to use as an audio feature. Some recent studies have explored phase unwrapping through deep learning (Masuyama et al., 2020; Thieling et al., 2021), which maybe promising avenues to adopt phase.

Studies that explore music-speech detection in broadcast signals have generally downsampled the audio to 8 kHz (Meléndez-Catalán et al., 2018), 16 kHz (Gimeno et al., 2020), or 22.05 kHz (Lemaire & Holzapfel, 2019). This is probably due to the advantages of dimensionality reduction and saving disk space. Further-

more, studies have downmixed broadcast programmes to mono instead of using stereo. This makes the system more flexible for radio and television streams that transmit only one channel. However, studies that explore environmental sound event detection have adopted higher sampling rates such as 44.1 kHz due to a greater number of acoustic classes (Mesaros et al., 2017; Salamon et al., 2017).

2.3 Types of Segmentation Algorithms

Primarily, there are two types of algorithms for audio segmentation — (1) distance-based segmentation and (2) segmentation-by-classification.

2.3.1 Distance-based Segmentation

In this style of segmentation, a distance metric such as Euclidean distance (Xue et al., 2010), Bayesian information criterion (BIC) (Huang & Hansen, 2006), or generalized likelihood ratio (GLR) (Wang et al., 2008) is calculated. For a given audio, a distance curve is plotted against a graph as shown in Figure 2.2. The peaks in the graph are associated with regions of high acoustic change. Thus, the audio is split into segments at these peaks. Subsequently, the audio classes within these segments is detected through a separate classification algorithm. The advantage of this method is that it is generally unsupervised and prior knowledge of the audio classes is not necessary. However, the disadvantage is that it is more sensitive to dissimilarities within the same acoustic class.

In the literature, audio features used for distance-based segmentation include MFCC and zero crossing rate (Huang & Hansen, 2006). Furthermore, when musical information is present in the audio, additional features such as linear prediction coefficients and linear spectral pairs have been adopted (Huang et al., 2009; Theodorou et al., 2014). BIC has been widely adopted in speaker diarisation

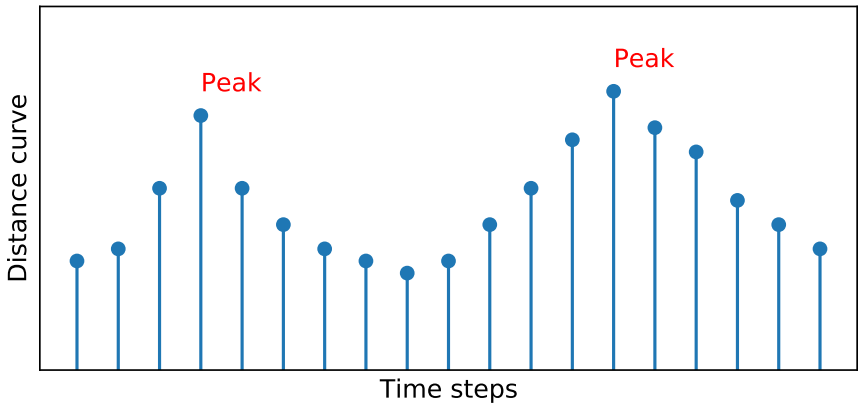


Figure 2.2: A plot of the distance against time steps. The peaks are associated with regions of high acoustic change. Figure adapted from Theodorou et al. (2014).

to generate break-points for changes in speaker (Chen et al., 1998; Kotti et al., 2008), language (Wu et al., 2005), and environment (Chen et al., 1998).

2.3.2 Segmentation-by-classification

In segmentation-by-classification, as the name suggests, each audio frame is individually classified. In the pre-processing stage, when extracting mel spectrograms, the audio is divided into small sections called frames. After classifying each frame, we effectively obtain the start and end points of each acoustic class. An illustration of this process can be found in Figure 2.3. In recent years, especially after the advances in deep learning, this technique has gained popularity and is more widely used than distance-based segmentation (Lemaire & Holzapfel, 2019; Gimeno et al., 2020). In addition, the pipeline of performing frame-based classification and obtaining acoustic boundaries is relatively simpler. It is also commonly adopted for other sound event detection tasks such as environmental sounds (Cakır et al., 2017; Salamon et al., 2017) and singing voice detection (Schlüter & Grill, 2015; Kum & Nam, 2019). More examples of studies using segmentation-by-classification can be found in sections 2.4 and 2.6.

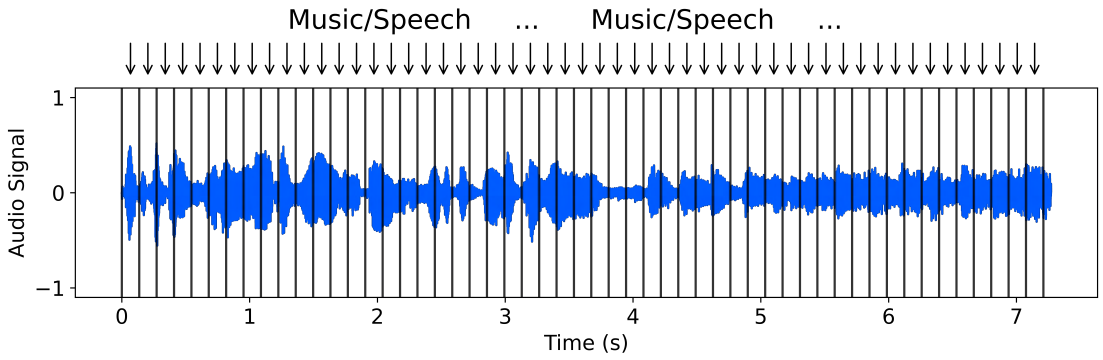


Figure 2.3: An illustration of segmentation-by-classification. Each frame of audio is classified to detect the presence of Music and Speech.

2.4 Machine Learning Algorithms

Traditionally, machine learning algorithms such as the Gaussian mixture model (GMM) (Kos et al., 2009), Hidden Markov Model (HMM) (Butko & Nadeu, 2011), Support Vector Machine (SVM) (Lo et al., 2010), and factor analysis (Castán et al., 2014) have been adopted for audio segmentation. In the Albayzin-2010 competition for Audio segmentation of broadcast news, the best performing model was the HMM. Subsequently, Castán et al. (2014) showed that factor analysis was more effective for segmentation. However, in recent years, deep neural networks surpass the performance of traditional machine learning algorithms. Recurrent neural network architectures such as bidirectional long short-term memory (B-LSTM) and bidirectional gated recurrent unit (B-GRU) are known to be effective for temporal data such as audio (Hochreiter & Schmidhuber, 1997; Cho et al., 2014). Additionally, Convolutional Neural Networks (CNNs) have been effective in interpreting patterns in mel spectrograms and are used for audio classification (Hershey et al., 2017) and segmentation (Meléndez-Catalán et al., 2018). Gimeno et al. (2020) used a B-LSTM to obtain state-of-the-art results on the Albayzin-2010 competition dataset. Lemaire & Holzapfel (2019) demonstrated that Temporal Convolutional Network (TCN) performed better than the B-LSTM for Music-Speech detection in radio broadcast.

Another state-of-the-art architecture popularly used in the sound event detection literature is the convolutional recurrent neural network (CRNN) (Sainath et al., 2015a; Choi et al., 2017b; Cakır et al., 2017). This network has the advantage of combining convolutions and recurrent units into one structure. The CRNN model has obtained state-of-the-art results on multiple datasets for environmental sound detection. For example, the top-performing model in the third task of the DCASE challenge 2017 for sound event detection was a CRNN (Adavanne & Virtanen, 2017). Furthermore, CRNNs have obtained state-of-the-art performance on the Urban-SED dataset (Salamon et al., 2017; Martín-Morató et al., 2019; Dinkel et al., 2021).

2.4.1 Convolutional Neural Network

A Convolutional Neural Network can be defined as a network that uses a *convolution* operation instead of general matrix multiplication in at least one layer (Goodfellow et al., 2016). CNNs grew extremely popular in the Computer Vision literature because of their ability to generalise better. They capture location-invariant spatial features, rather than treating each pixel independently. CNNs carry an advantage of *parameter sharing*, which is, when the machine learning model develops an ability to detect a particular feature, the same ability can be reused in multiple locations. Another advantage of CNNs is the sparsity of connections, that is, the output value of a neuron depends only on a few input values in the previous layer, thus reducing the number of parameters to train. Convolutional layers are often followed by a pooling layer such as max pooling (Zhou & Chellappa, 1988) to reduce the size of the representation. This improves generalisation and speeds up computation.

Zeiler & Fergus (2014) presented a method to visualise feature maps of convolutional layers for Computer Vision tasks, where the initial layers identify small local patterns, such as edges or circles. The subsequent layers progressively com-

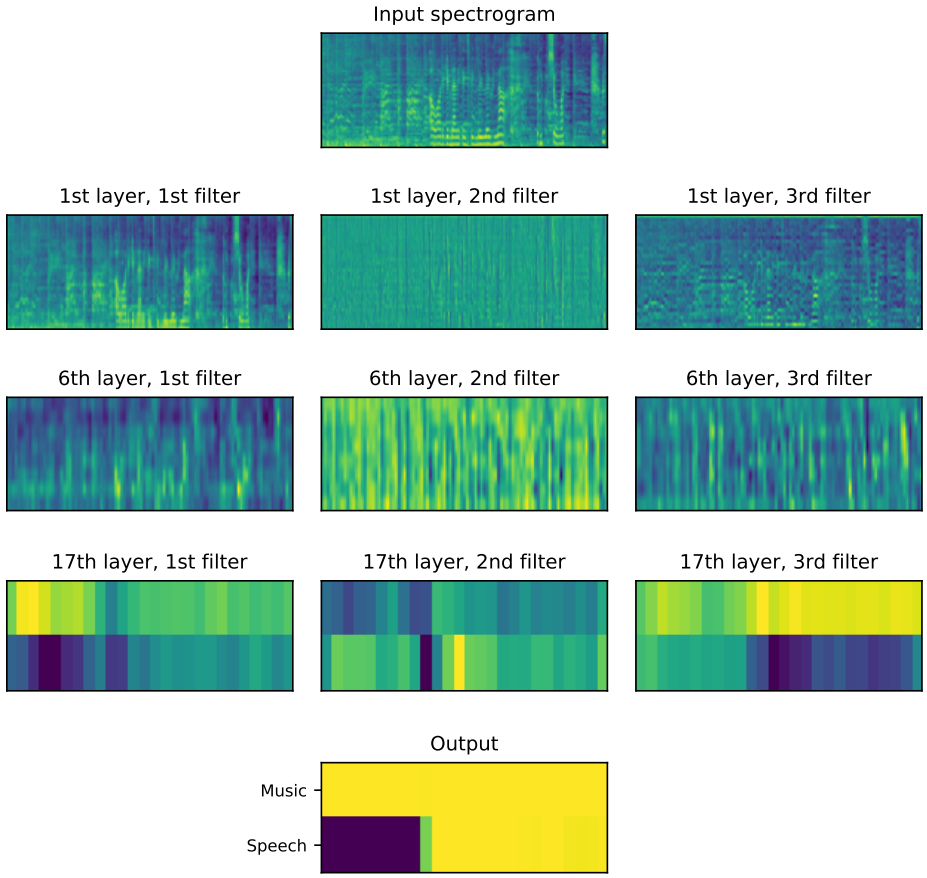


Figure 2.4: This figure plots the output of convolutional layers at different depths of the network. It illustrates how convolutions help in capturing high-level features in the spectrogram. The input spectrogram is an 8 s audio file that contains Music from 0 s to 8 s and Speech from 3 s to 8 s. These values were calculated from the CNN presented in section 4.3.2.

bine them into more meaningful structures, such as textures and mesh patterns; to later on identifying class-specific aspects like dog's face and bird's leg (Zeiler & Fergus, 2014; Yamashita et al., 2018). In addition to Computer Vision, CNNs were shown to be highly effective for audio classification (Hershey et al., 2017) because mel spectrograms can be treated as images. Papakostas & Giannakopoulos (2018) investigated the benefits of using deep visual features for music-speech detection. Convolutions on sound signals can be performed using 1D or 2D kernels. 1D kernels perform convolutions only along the time domain. 2D kernels perform convolutions along the time as well as frequency domain. Performing convolutions on both dimensions is known to have advantages over just one dimension

(Choi et al., 2017b). Figure 2.4 shows how an audio spectrogram is processed by a CNN at different depths of the network.

2.4.2 Recurrent Neural Network

Recurrent Neural Networks were designed to process sequential data such as text and audio. As shown in figure 2.5, RNNs are called recurrent because they perform the same task for every element of a sequence, with the output depending on the previous computations (Graves et al., 2013; Coşkun et al., 2017). In addition, the memory component of RNNs could store information about the past (Liu et al., 2017). The advantage of RNNs over fully connected layers is that it is capable of sharing features across different positions in a sequence. This enables it to capture dependencies between words within a sentence or samples within an audio signal.

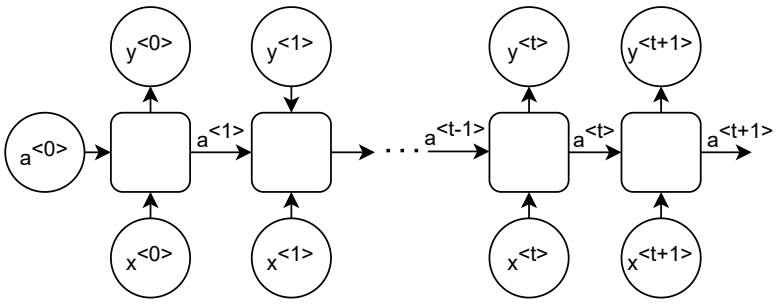


Figure 2.5: An illustration of the Recurrent Neural Network (RNN). x, y, a, and t stand for input, output, activation, and time respectively.

The LSTM network (Hochreiter & Schmidhuber, 1997) is a commonly adopted RNN for speech recognition and audio classification. An LSTM cell has three gates — update, forget, and output. These three gates help the LSTM network memorise data over time steps. Another type of RNN is the Gated Recurrent Unit (GRU) (Cho et al., 2014). It contains two gates instead of three — update and reset gate. It has fewer parameters than the LSTM cell and is less prone to over-fitting. GRUs have been effectively used for sound event detection (Lu & Duan, 2017) and speech recognition (Ravanelli et al., 2018). Bidirectional variants of LSTMs and GRUs are known to improve the performance for audio information retrieval. In such a network, the audio signal is passed forward and backward to learn both

dependencies.

2.4.3 Temporal Convolutional Network

The Temporal Convolutional Network is a family of networks that perform causal convolutions and are designed to handle sequential data (Bai et al., 2018). TCNs also adopt other bespoke techniques such as dilated convolutions to inflate the size of the kernel and effectively, increase the receptive field (Yu & Koltun, 2015). Additionally, skip-connections of layers through residual blocks (He et al., 2016), which are popular in CNNs have also been used by TCNs.

Bai et al. (2018) presented a theoretical and empirical study regarding the advantages of TCNs over RNN architectures. Firstly, TCNs can be parallelised during inference because prediction at a time step does not depend on the prediction at an earlier time step. Therefore, the input sequence can be processed as a whole instead of sequentially. Secondly, a common problem with RNN architectures is exploding and vanishing gradients because they attempt in learning long-term dependencies (Bengio et al., 1994). As *backpropagation* in TCNs does not follow the time dimension, this problem is avoided.

Lemaire & Holzapfel (2019) proposed a non-causal Temporal Convolutional Network (ncTCN), which learns forward and backward dependencies in audio signals. It performed better than the traditional TCN for Music-Speech detection. Furthermore, Meléndez-Catalán et al. (2020) adopted this architecture for a task called *Music Loudness Estimation*, which divides an audio signal into three classes — Foreground Music, Background Music, and No Music.

2.4.4 Convolutional Recurrent Neural Network

As mentioned earlier, the Convolutional Recurrent Neural Network combines convolutional and recurrent layers into one structure. The initial layers of such a

network are convolutional layers followed by recurrent layers. The convolutions can be performed with either 1D or 2D kernels, with the latter performing better for audio classification (Choi et al., 2017b). Some studies have used LSTMs to build the recurrent blocks (Sainath et al., 2015a; Lemaire & Holzapfel, 2019) and others have used GRUs instead (Cakır et al., 2017; Choi et al., 2017b). After the recurrent layers, some studies have also included fully connected layers and termed it Convolutional Long Short-Term Memory Fully Connected Deep Neural Network (CLDNN) (Sainath et al., 2015a; Lemaire & Holzapfel, 2019). Sainath et al. (2015a) developed the CLDNN architecture for speech recognition, which does not perform segmentation-by-classification. Moreover, Lemaire & Holzapfel (2019) explored CLDNN for segmentation-by-classification. However, CRNN without fully connected layers has been more popular for sound event detection (Cakır et al., 2017; Adavanne & Virtanen, 2017).

2.5 Post-processing / Smoothing

The predictions made by the neural network may contain spurious audio events. Suppose the duration of an audio event is very short, it might be a false positive. As you can see in Figure 2.6, there are short pauses in the detection of speech. These predictions can be smoothed to obtain a more reliable prediction from the network. Median filtering is one technique used in the literature (Schlüter & Sonnleitner, 2012). Another approach uses threshold-dependent values (Lemaire & Holzapfel, 2019). For example, if the duration of speech is less than 1.3 s, the prediction can be discarded. In addition, if the silence between consecutive speech events is less than 0.8 s, the gap can be smoothed. The literature has also adopted machine learning models such as the HMM for smoothing. For instance, Gimeno et al. (2020) passed the predictions of a B-LSTM network into an HMM module for smoothing.

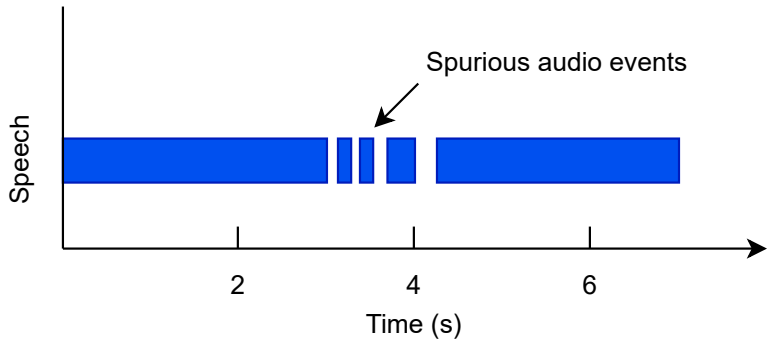


Figure 2.6: Spurious audio events in predictions of the neural network.

2.6 Comparison

In this section, we compare the various algorithms in the literature for Music-Speech detection. Traditional machine learning methods such as HMM, SVM, and BIC were widely adopted at the Albayzin-2010 competition, which segmented audio into five classes — Speech, Music, Speech+Noise, Speech+Music, and Other (Butko & Nadeu, 2011). The top rank (Gallardo Antolín & San Segundo Hernández, 2010) at the competition obtained an error rate of 0.30. They used the segmentation-by-classification approach with MFCCs, chroma and spectral entropy as features (Gimeno et al., 2020). They adopted an hierarchical HMM, where audio is first segmented into Music / non-Music. Subsequently, the non-Music is segmented into Speech+Music / non-Speech+Music. Finally, the non-Speech+Music is segmented into Speech / Speech+Noise (Gallardo Antolín & San Segundo Hernández, 2010; Butko & Nadeu, 2011). The second place at the competition adopted distance-based segmentation and obtained an error rate of 0.33. BIC was used to detect the segment boundaries and SVM was used to classify the segments (Docio-Fernandez et al., 2010; Butko & Nadeu, 2011).

Castán et al. (2014) improved the segmentation-by-classification approach through factor analysis. Instead of adopting a hierarchical structure, this technique compensates the within-class variability by using class-dependent factor loading matrices to reduce the mismatch between the training and test set. This study obtained an error rate of 0.23 on the Albayzin-2010 competition dataset, surpassing

the top rank. Gimeno et al. (2020) adopted a deep learning architecture, which is the B-LSTM to obtain an error rate of 0.19 on the Albayzin-2010 competition dataset. Again, they used segmentation-by-classification and explored temporal pooling techniques within the neural network to improve the performance of audio segmentation.

At the MIREX Music-Speech Detection competition 2015, the top rank was secured by Marolt (2015) with an overall F-measure of 89.41%. They trained a logistic regression classifier on 3 s chunks of audio. During testing, a weighted average of individual predictions is calculated by running the classifier on the entire audio file. The second rank at the competition was obtained by Lidy (2015) with an overall F-measure of 88.49%. They used a CNN with one convolutional layer, one fully connected hidden layer, and one softmax layer for the output.

At the MIREX music-speech detection competition 2018, all the submissions were evaluated on two different test sets:
Dataset 1: 27 hours of audio from 8 different TV program types from France, Germany, Spain and the United Kingdom.
Dataset 2: 10 hours of audio corresponding to French TV and radio programs, provided by INA (French National Institute of Audiovisual).
The top rank was again secured by Marolt (2018). They had submitted two types of models to the competition — (1) the same logistic regression classifier submitted by them in 2015 and (2) deep residual neural network (ResNet). Looking at an averaged performance over both test sets, the ResNet performed better than the logistic regression classifier for music detection. However, the logistic regression classifier surpassed the performance of the ResNet for speech detection. The reader is referred to the MIREX website for more details on the results (Schlüter et al., 2018).

Lemaire & Holzapfel (2019) explored a deep learning architecture called the ncTCN for Music-Speech detection. The study aggregated a large dataset (approx. 172 h) by combining in-house datasets (which cannot be shared as it is

proprietary audio) and openly available datasets. The openly available datasets included MUSAN (Snyder et al., 2015), GTZAN music-speech (Tzanetakis & Cook, 2000), Scheirer & Slaney (1997), and ESC-50: Dataset for Environmental Sound Classification (Piczak, 2015). On the in-house test set, Lemaire & Holzapfel (2019) obtained an overall F-measure of 96.8%. They also tested their model on the MIREX competition dataset 2. For Speech F-measure, the ncTCN obtained state-of-the-art performance of 94.6%, surpassing the previous highest of 93.3% by the Logistic Regression model of Marolt (2018). However, for Music F-measure, the ncTCN obtained 87.9%, which is lower than 92.3% obtained by the logistic regression model of Marolt (2018).

It can be observed in the above paragraphs that the number of studies using distance-based segmentation has reduced over the years. The top tank at the Albayzin-2010 competition and the following studies that improved state-of-the-art performance adopted segmentation-by-classification (Butko & Nadeu, 2011; Castán et al., 2014; Gimeno et al., 2020). Even at the MIREX competitions 2015 and 2018, the submissions have predominantly adopted segmentation-by-classification (Marolt, 2015, 2018; Lidy, 2015; Choi et al., 2018). Another important trend is the use of deep learning models in recent studies. For instance, Gimeno et al. (2020) obtained state-of-the-art performance on the Albayzin-2010 dataset using a B-LSTM. Lemaire & Holzapfel (2019) obtained state-of-the-art performance for speech detection on the MIREX 2018 competition dataset. However, a simple logistic regression system proposed by Marolt (2018) still surpassed the performance of the ncTCN for music-detection. This conveys that although deep learning models have great potential to improve detection accuracy, they are also prone to overfitting if the training set is not large and diverse enough. More details on overfitting is given in section 2.7.4. Moreover, deep learning models such as the CRNN and CNN have obtained state-of-the-art performance on multiple datasets for sound event detection (Adavanne & Virtanen, 2017; Martín-Morató et al., 2019; Salamon et al., 2017). Table 2.1 summarises various algorithms in the literature for audio segmentation and sound event detection.

Table 2.1: A summary of various studies for audio segmentation and sound event detection.

Study	**Audio Classes**	**Algorithm**	**Evaluation**
Lemaire & Holzapfel (2019)	Music, Speech	ncTCN	$F_{overall} = 96.8\%$ on in-house test set; $F_{music} = 87.9\%$, $F_{speech} = 94.6\%$ on MIREX (2018) dataset 2
Marolt (2018)	Music, Speech	Logistic Regression	$F_{music} = 39.0\%$, $F_{speech} = 91.2\%$ on MIREX (2018) dataset 1; $F_{music} = 92.3\%$, $F_{speech} = 93.3\%$ on MIREX (2018) dataset 2
Marolt (2018)	Music, Speech	ResNet	$F_{music} = 54.8\%$, $F_{speech} = 90.9\%$ on MIREX (2018) dataset 1; $F_{music} = 91.6\%$, $F_{speech} = 91.4\%$ on MIREX (2018) dataset 2
Choi et al. (2018)	Music, Speech	MLP	$F_{music} = 49.4\%$, $F_{speech} = 77.2\%$ on MIREX (2018) dataset 1; $F_{music} = 79.7\%$, $F_{speech} = 84.6\%$ on MIREX (2018) dataset 2
Marolt (2015)	Music, Speech	Logistic Regression	$F_{overall} = 89.41\%$ on MIREX (2015) dataset
Lidy (2015)	Music, Speech	CNN	$F_{overall} = 88.49\%$ on MIREX (2015) dataset
Meléndez-Catalán et al. (2018)	Music, No Music	CNN	$F_{music} = 90.0\%$, $F_{no\_music} = 90.1\%$ on MIREX (2018) dataset 1; $F_{music} = 95.7\%$, $F_{no\_music} = 93.6\%$ on MIREX (2018) dataset 2
Gimeno et al. (2020)	Speech, Music, Speech+Noise, Speech+Music, Other	B-LSTM with HMM resegmentation and temporal pooling	$\mathtt{Error\ rate} = 0.19$ on the Albayzin-2010 evaluation
Castán et al. (2014)	Speech, Music, Speech+Noise, Speech+Music, Other	Factor Analysis	$\mathtt{Error\ rate} = 0.23$ on the Albayzin-2010 evaluation
Gallardo Antolín & San Segundo Hernández (2010)	Speech, Music, Speech+Noise, Speech+Music, Other	HMM	$\mathtt{Error\ rate} = 0.30$ on the Albayzin-2010 evaluation

Study	Audio Classes	Algorithm	Evaluation of best algo.
Docio-Fernandez et al. (2010)	Speech, Music, Speech+Noise, Speech+Music, Other	BIC for segmentation and SVM for classification;	$\texttt{Error rate} = 0.33$ on the Albayzin-2010 evaluation
Adavanne & Virtanen (2017)	Environmental sound detection for six acoustic classes.	CRNN	$\texttt{Error rate} = 0.79$ in DCASE challenge 2017, task 3
Jeong et al. (2017)	Environmental sound detection for six acoustic classes.	CNN;	$\texttt{Error rate} = 0.81$ in DCASE challenge 2017, task 3
Lu & Duan (2017)	Environmental sound detection for six acoustic classes.	B-GRU;	$\texttt{Error rate} = 0.83$ in DCASE challenge 2017, task 3
Martín-Morató et al. (2019)	Environmental sound detection for ten acoustic classes.	CRNN with envelope estimation;	$F_{\text{overall}} = 64.7\%$ on the Urban-SED dataset
Salamon et al. (2017)	Environmental sound detection for ten acoustic classes.	CNN;	$F_{\text{overall}} = 56.9\%$ on the Urban-SED dataset
Salamon et al. (2017)	Environmental sound detection for ten acoustic classes.	CRNN;	$F_{\text{overall}} = 56.0\%$ on the Urban-SED dataset

2.7 Factors Specific to Machine Learning

2.7.1 Datasets

In order to train machine learning models for audio segmentation, we require audio signals to be labelled. The labels need to contain the composite audio classes along with their respective acoustic boundaries. Note that the literature comprises many datasets labelled at the file level. That is, there are separate audio files of

music and speech. For example, genre-recognition datasets contain many music examples (Tzanetakis & Cook, 2002) and speech recognition datasets contain many speech examples (Panayotov et al., 2015). However, the problem is that these files are not mixed like a radio or TV programme. In broadcast audio, the content is well-mixed, with speech and music seamlessly transitioning between each other, and often overlapping.

Generally, studies have trained machine learning algorithms on proprietary audio (Schlüter & Sonnleitner, 2012; Lemaire & Holzapfel, 2019; Gimeno et al., 2020). Audio is obtained directly from the broadcasters and labelled either by the authors or by external annotators. The process of annotating data is a time-consuming and expensive process. It takes four to five hours to annotate an hour of audio (Meléndez-Catalán et al., 2019). Furthermore, the labelled data cannot be shared because broadcast audio is proprietary material. This hinders the reproducibility of audio segmentation research.

Amongst the openly available datasets, the MuSpeak Team (2015) presented an example dataset for the Music Information Retrieval Evaluation eXchange (MIREX) 2018 music and speech detection competition. Moreover, Open Broadcast Media Audio from TV (OpenBMAT) is another openly available dataset concentrating on the estimation of relative loudness of music, but not Music-Speech detection. As mentioned earlier, there are many openly available datasets containing separate files of music and speech. Some include the MUSAN corpus (Snyder et al., 2015), GTZAN music and speech detection dataset (Tzanetakis & Cook, 2000), the dataset by Scheirer & Slaney (1997), and the LibriSpeech corpus (Panayotov et al., 2015).

2.7.2 Dataset Splits

The dataset available to train machine learning models is generally divided into three parts — train, validation, and test sets. A large and diverse training set

helps the algorithm learn various patterns and thus, making it robust. However, it is important to prevent overfitting. Overfitting occurs when the algorithm performs well on the training data, but does poorly on unseen data. Hence, a cross-validation set (or simply called validation set) is used when training the model. In recent practices, the evaluation on the validation set is often done at the end of each epoch. An epoch is defined as a run through the entire training set. After achieving a satisfactory performance on the validation set, the model is finally evaluated on the test set. This ensures that we have not overfit the validation set.

The amount of data for training, validation, and testing depends on the context. Traditionally, many have recommended a 70-20-10% split, but this has changed because of more data being available in recent years. For validation and testing, sufficient data should be kept aside to obtain a reliable evaluation of the algorithm. Adding too much data into these splits is basically wasting training data, and adding less into these splits leads to erroneous evaluations. Some examples of splitting the dataset in the literature include — Salamon et al. (2017) set aside 16.6 h for training and 5.5 h each for validation and testing. Lemaire & Holzapfel (2019) set aside 33 h, 9.5 h, and 4.7 h for training, validation, and testing respectively.

When splitting the dataset into parts, it is important to ensure that the audio files are different from each other. For example, it is a bad idea to include the first 10 s of a song for training and the next 10 s of the same song for validation. Instead, it is common practice to use separate audio files for training, validation, and testing (Lemaire & Holzapfel, 2019; Gimeno et al., 2020).

Studies have also adopted other methods such as k-fold cross-validation to split the datasets. In such a method, the dataset is split into k-folds. For example, the TUT Sound Events 2017 dataset split the data into four folds. Initially, one of the folds can be set aside for validation and the others for training. This process can be performed across all folds and thus, using up the entire data for training. Some studies have suggested methods to prevent overfitting the validation set in

such a setup (Ng et al., 1997; Forman & Scholz, 2010).

2.7.3 Data Augmentation

Data augmentation performs transformations on data to artificially enhance the size of training sets and effectively improve the model's performance. In Computer Vision, common image transformations include rotation, horizontal and vertical flipping, and colour modification, to name but a few (Shorten & Khoshgoftaar, 2019). Studies have shown data augmentation to be effective for sound event detection tasks (Schlüter & Grill, 2015; Salamon & Bello, 2017). Some common audio augmentations include pitch shifting, time stretching, dynamic range compression, and random frequency filtering. An important factor to consider when augmenting audio is to preserve the meaning of labels for the original audio. For instance, Salamon & Bello (2017) observed that the confusion between air conditioner and engine idling classes increased due to augmentation.

Schlüter & Grill (2015) investigated various data augmentation techniques for singing voice detection. They found that pitch shifting and random frequency filtering improved the model's performance. The impact of time-stretching and randomly varying loudness was negligible. In addition, corrupting the audio with Gaussian noise and random dropout worsened the performance of the model. Lemaire & Holzapfel (2019) incorporated time stretching, pitch shifting, Gaussian filtering, loudness manipulation, and block mixing for Music-Speech detection in radio broadcast.

Lemaire & Holzapfel (2019) randomly mixed audio files that were labelled at the file level. This way, the neural network is trained to simultaneously identify the presence of speech and music in the audio. Gimeno et al. (2020) adopted another technique called *mixup* for audio segmentation. Mixup is an augmentation technique that creates convex combinations of pairs of examples and their labels (Zhang et al., 2018). Two examples are combined into one and the labels are no

longer one-hot encoded. Instead, the labels are multiplied by a factor of how much each acoustic class is present in the *mixed* example.

A problem associated with audio augmentations is the computational cost. Unlike augmenting images, processes such as pitch shifting and frequency filtering are computationally expensive. Therefore, applying augmentations in real-time when training deep neural networks slows down the training process. Additionally, the augmented audio examples need to be converted to mel spectrograms and thus, increasing the bottleneck. Therefore, a common practice is to perform the STFT on audio and store them locally (Schlüter & Grill, 2015; Lemaire & Holzapfel, 2019). The audio augmentations are applied directly to spectrograms. Subsequently, the spectrograms are converted to the mel scale during training. However, despite these optimizations, audio augmentations considerably hinder the speed of training deep neural networks.

Park et al. (2019) proposed a technique called SpecAugment for speech recognition that performs augmentations directly on mel spectrograms. In this case, the mel spectrogram is treated like an image. SpecAugment has three components — (1) time warping, (2) randomly masking a series of frequency bins, and (3) randomly masking a series of time steps. Although, SpecAugment was effective and straightforward for speech recognition, it is more complicated when adopting it for sound event detection. For example, time warping and randomly masking a series of time steps would require the reconstruction of labels for acoustic boundaries. However, this is not the case with speech recognition.

2.7.4 Regularisation

Regularisation is a way of reducing overfitting in neural networks. There are multiple regularisation methods adopted by researchers in the literature. A commonly used approach is dropout (Srivastava et al., 2014), where random neurons from the network are dropped before each epoch. This minimises the possibility for a

single neuron to have a very high influence on the output and effectively, reducing overfitting. Some other techniques include L1 and L2 normalisation, which adds a penalty to the cost function and thus, creating a regularising effect (Kukačka et al., 2017). Researchers have also investigated how to optimise regularisation techniques for RNNs and sequential data. For example, Gal & Ghahramani (2016) presented a technique called recurrent dropout, which drops the same network units at each time step.

Another commonly used technique for regularisation is early stopping (Yao et al., 2007). It works on the following principle. When training a neural network, the error on the validation set begins to reduce. After a point, when the network begins to overfit, the error starts increasing. Early stopping stops the training before the error starts increasing and therefore, producing a regularising effect. Another advantage of early stopping is that it saves computational power by not training the network for unnecessary epochs.

2.7.5 Transfer Learning

Knowledge learned for one task can be applied to a different task. This process is known as transfer learning. For example, VGGish is a neural network that distinguishes 632 audio classes in the AudioSet database (Gemmeke et al., 2017; Hershey et al., 2017). Chourdakis et al. (2019) adapted the VGGish network to distinguish between only three audio classes — Music, Speech and Sound Effects. Transfer learning can be used as a method to manage data scarcity. Instead of training a model from the start, it is often simpler to adapt a pre-trained network for the required task. Figure 2.7 shows an example of transfer learning.

As spectrograms are popularly adopted as features for audio classification, these can also be considered to be images. Therefore, Papakostas & Giannakopoulos (2018) used CNNs trained for Computer Vision and applied transfer learning to perform music-speech detection. Generally, in transfer learning, the initial layers

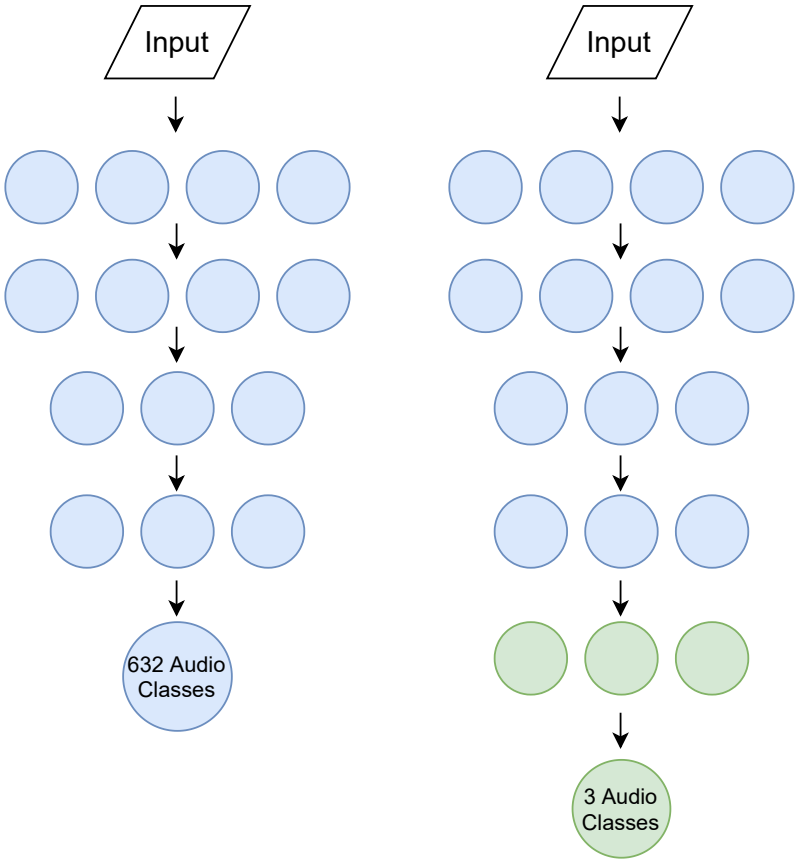


Figure 2.7: An illustration of transfer learning. The neurons in blue denote that the weights of the original network are left unchanged. The weights of neurons in green are calculated when training the new network.

of the pre-trained network are kept frozen. A small number of layers are added to the end of the network and only these layers are trained for the new task. These are coloured green in Figure 2.7. After the final layers are trained for the new task, many studies perform an additional fine-tuning step over the entire network (Diment & Virtanen, 2017). This is done with a very small learning rate to only make small changes to the weights of the network.

2.7.6 Problem Formulation

In broadcast signals, audio classes occur independently of each other. In other words, speech and music can occur either separately or simultaneously. Studies have addressed this problem as a multi-class (Gimeno et al., 2020) or multi-label task (Lemaire & Holzapfel, 2019). Gimeno et al. (2020) detected five different

classes — Speech, Music, Speech+Noise, Speech+Music, and Other. In such a multi-class setting as shown in Figure 2.8, the output layer of the network is a softmax unit, which chooses between the different audio classes. Note that the combination of Speech and Music is considered to be a separate class. However, Lemaire & Holzapfel (2019) presented Music-Speech detection as a multi-label problem by performing binary classification on each acoustic class as shown in Figure 2.8.

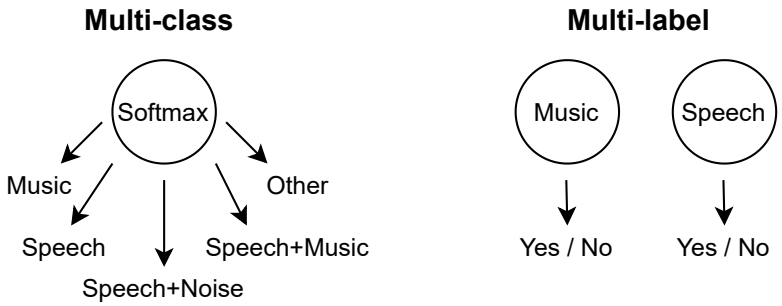


Figure 2.8: A comparison of multi-class and multi-output systems.

For sound event detection problems such as environmental audio, the multi-label system is more popular where each audio class is represented as a binary classification (Cakır et al., 2017). Cakir et al. (2015) compared multi-label and multi-class systems for environmental audio and showed that multi-label generalises better. An advantage of the multi-label approach is that the dependencies across acoustic classes are captured. For instance, Speech+Music is considered to be a combination of two acoustic classes instead of a completely different class. Multi-class detection does not acknowledge this dependency.

2.7.7 Loss Functions and Optimisers

As mentioned earlier, sound event detection has mostly been designed as a classification task in the literature and has therefore adopted loss functions specifically designed for classification. On one hand, multi-label systems generally use binary cross-entropy, which is a commonly used loss function for binary classification. On the other hand, multi-class systems use categorical cross-entropy. When training

a machine learning model, the loss function is minimised by using an optimiser. Common ones in the literature include Adam (Choi et al., 2017b; Cakır et al., 2017), Stochastic Gradient Descent (SGD) (Schlüter & Grill, 2015; Lemaire & Holzapfel, 2019), and RMSProp (Parascandolo et al., 2016). Adam (Kingma & Ba, 2015) has been the most popular choice among studies and generally converges faster than the other approaches. However, there is no consensus to pick the best optimizer for sound event detection.

2.7.8 Training Strategies

It is a common practice in the literature to normalise mel spectrograms before training (Schlüter & Grill, 2015; Meléndez-Catalán et al., 2018; Lemaire & Holzapfel, 2019). Normalisation ensures that the values in all features are within similar ranges and thus, speeding up training. Moreover, for Deep Learning, the training data is fed to the neural network in mini-batches. This way, the neural network does not need to parse the entire training set to perform one round of backpropagation. Instead, backpropagation is performed after each batch. Common batch sizes in the literature include 32 (Schlüter & Grill, 2015; Lemaire & Holzapfel, 2019), 64 (Chen et al., 2020), and 128 (Adavanne & Virtanen, 2017; Miyazaki et al., 2020a).

Batch Normalisation (BN) (Ioffe & Szegedy, 2015) is a popular method to speed up training. BN minimises *internal covariance shift* by normalising layer inputs. Using BN, a model's training curve converges in fewer epochs. An added advantage of using BN is that it produces a regularisation effect and thus, improving the generalisation of a model. Convolutional blocks are often fitted with BN in networks such as CNN, TCN, and CRNN (Cakır et al., 2017; Lemaire & Holzapfel, 2019). Ba et al. (2016) proposed a technique called Layer Normalisation and showed that it is more effective than BN for recurrent architectures like LSTM and GRU.

Training deep learning models requires high computational power, especially Graphical Processing Units (GPUs). GPUs are capable of parallelising training pipelines and therefore, obtaining high speed-ups. The NVIDIA CUDA Deep Neural Network library (cuDNN) supports primitives for GPU-accelerated Deep Learning. Libraries such as TensorFlow, Torch, Theano, Caffe, and Deeplearning4j rely on such GPU-acceleration (Parvat et al., 2017). As GPU resources may not be accessible to many, cloud-based platforms such as Google Colab and Kaggle have gained popularity for Deep Learning in recent years.

2.7.9 Semi-supervised and Weakly-supervised Learning

To address data scarcity, studies have adopted weakly-supervised learning for sound event detection (Kumar & Raj, 2016; Kong et al., 2019). *Weak labels* contain information about the classes present in an audio sequence, but not their acoustic boundaries. This is different from *strongly labelled data*, which contains information about the acoustic boundaries as well. Fewer resources are required to annotate weakly labelled datasets (Adavanne et al., 2019). As labels are available only at the sequence level, the loss is calculated between the sequence-level prediction and the weak label when training the neural network (Miyazaki et al., 2020b). Datasets aggregated for Audio Classification can be employed for Audio Segmentation through weakly-supervised learning. For example, Adavanne et al. (2019) adopted AudioSet (Gemmeke et al., 2017) for Sound Event Detection.

Semi-supervised learning is an approach to combine unlabelled data with labelled data for training neural networks (Van Engelen & Hoos, 2020). After training the machine learning model on labelled data, Zhang & Schuller (2012) showed that the performance of Sound Event Classification can be further improved by using unlabelled data. Traditionally, self-training and co-supervised learning were well-known approaches to semi-supervised learning (de Sa, 1994; Blum & Mitchell, 1998). Many studies have explored semi-supervised learning for

deep neural networks using the *cluster assumption* paradigm (Sajjadi et al., 2016). Doukhan & Carrive (2017) adopted semi-supervised learning for Music-Speech detection. Tarvainen & Valpola (2017) proposed a mean-teacher approach that has gained popularity within the Sound Event Detection community (Yan et al., 2020; Lin et al., 2020). In such a setup, there exists a teacher model and a student model that train on large-scale unlabelled data. The difference in predictions of both models on unlabelled data is exploited to improve regularisation.

2.8 Metrics

Accuracy can be used to evaluate the performance of a machine learning model, but the problem arises when the amount of data for each audio class is different. For example, if 95% of the test set contains silence and only 5% has brakes-squeaking sounds, then the algorithm can afford to predict silence all the time with an accuracy of 95%. This way we would never be able to detect the presence of brakes-squeaking. Hence, accuracy is not a good metric in this scenario. Mesaros et al. (2016) presented a comprehensive survey discussing the various metrics proposed for sound event detection. They also created an open-source Python toolbox called *sed_eval* to conveniently calculate metrics for sound event detection. Precision or *positive prediction value* can be defined as the following. Among all the values that were detected as positive by the classifier, how many of them are actually correct. Conversely, recall or *sensitivity* can be defined as the following. Among all the examples that were true in the ground truth, how many were successfully detected by the classifier. Equation 2.1 and 2.2 formulates precision and recall respectively.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.2}$$

There needs to be a trade-off between precision and recall because an increase in one is often accompanied by a decrease in the other. Hence, F-measure or F_1 score combines precision and recall into a single metric by calculating their harmonic mean as shown in equation 2.3.

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.3}$$

In some cases, precision and recall may have different degrees of importance. For a probability p output by the machine learning algorithm and a decision threshold θ, the audio class is present if $p \geqslant \theta$ and the audio class is absent if $p < \theta$. This decision threshold is generally set to 0.5. In other words, the audio class is present if the algorithm is at least 50% confident. However, in some cases we might want this to be different. For example, in an audio surveillance system that detects suspicious activity, we want to ensure that no events of suspicious activity are missed, but it is acceptable to have some false alarms. In other words, we want the algorithm to have a high *sensitivity* to minimise the chance of missing a suspicious action. Therefore, we can set the decision threshold to 0.3, which would improve the recall of the algorithm, but compromise its precision.

F-measure is a widely adopted metric for studies and competitions in audio segmentation (Schlüter et al., 2018; Lemaire & Holzapfel, 2019) and sound event detection (Salamon et al., 2017; Mesaros et al., 2017). In these studies, precision and recall are given equal importance and a decision threshold of 0.5 is normally used. Studies have also adopted error rate to evaluate models for segmentation (Butko & Nadeu, 2011; Gimeno et al., 2020; Mesaros et al., 2017). It is inspired by the *Word Error Rate* in speech recognition. Error rate aggregates errors in terms of *insertions (I)*, *deletions (D)* and *substitutions (S)*. It can be formulated by equation 2.4

(Mesaros et al., 2016).

$$\text{Error Rate} = \frac{\sum_{k=1}^{K} S(k) + \sum_{k=1}^{K} D(k) + \sum_{k=1}^{K} I(k)}{\sum_{k=1}^{K} N(k)} \tag{2.4}$$

with $N(k)$ being the number of active sound events.

2.8.1 Segment-based and Event-based Metrics

The predictions made by the machine learning algorithm need to be compared with the ground truth. This can be done by breaking the reference labels into short segments, such as 10 ms. Hence, for every 10 ms in the ground truth, we check if the machine learning algorithm has made the correct prediction. This is known as segment-based metrics. Depending on the task, larger segments maybe used for evaluation. For example, the TUT sound event detection dataset (Mesaros et al., 2017) and Urban-SED (Salamon et al., 2017), which focus on environmental sound detection, used 1 s-segments for evaluation.

Event-based metrics, as the name suggests compare each acoustic event as a whole. The audio class predicted by the machine learning needs to be the same as the one in the ground truth. Subsequently, the onset and offset times of each event need to fall within a threshold, such as 500 ms. A problem with event-based metrics is that small differences in the practices of annotators can lead to significantly different results (Lemaire & Holzapfel, 2019). For example, if there is a pause of 0.9 s in Speech, one annotator may consider this to be two separate Speech events and the other may consider it to be a single event. Table 2.2 contains two examples of calculating segment-based and event-based metrics. In the second example, the machine learning detects a pause of 0.9 s, but gets highly penalised by the event-based metrics.

Ground truth	Predictions	Segment-based F-measure (%)	Event-based F-measure (%)
0.0, 4.4, Music 2.6, 5.9, Speech	0.0, 4.3, Music 2.5, 6.0, Speech	98.06	100
0.0, 4.4, Music 2.6, 5.9, Speech	0.0, 4.3, Music 2.5, 4.0, Speech 4.9, 5.9, Speech	92.41	40

Table 2.2: Two examples of predictions made by the machine learning model. The segment-based and event-based metrics were calculated using the *sed_eval* toolbox. You can observe that the event-based F-measure drops to 40% in the second example because it compares acoustic events as a whole.

2.8.2 Micro and Macro F-measure

When there are more than two acoustic classes, the overall F-measure of the system can be calculated in two ways — micro-averaging and macro-averaging. Micro-averaging gives equal weight to every instance in the test set. The true positives, false positives, and false negatives are aggregated over the entire set to calculate the metrics (Mesaros et al., 2016). Contrarily, in macro-averaging, each acoustic class is given equal importance. The metrics are separately calculated for each class and averaged in the end. Generally, micro-averaging has been more commonly adopted by studies to present the overall F-measure (Mesaros et al., 2017; Salamon et al., 2017; Lemaire & Holzapfel, 2019).

2.9 Discussion and Contributions

This chapter surveyed the literature on adopting Machine Learning for Audio Segmentation. Deep Neural Networks are effective for content-based retrieval in broadcast audio. Both, Sound Event Detection and Audio Segmentation have similar goals — to detect acoustic classes and their respective boundaries within an audio stream. Although Audio Segmentation aims to divide audio into homogenous sections, it is often modelled as a Sound Event Detection task. Therefore,

there is considerable overlap between practices in both domains.

Research into Artificial Intelligence (AI) can be broadly divided into data-centric and model-centric approaches (DeepLearning.AI, 2021). Data centric AI focuses on the data itself, which essentially acts as the fuel for the machine learning algorithm. This includes collecting larger training sets, improving the quality of training examples, optimising the pre-processing and feature extraction process, and data augmentation. Model-centric AI focuses on developing novel neural network architectures and improving current ones. In a talk by Andrew Ng, he interestingly points out that 80% of the time spent by an AI engineer is data cleaning and aggregation. However, 80% of the research published in AI is model-centric and only 20% is data-centric. Therefore, he stresses the importance of data-centric approaches to address real-world problems (DeepLearning.AI, 2021).

A crucial problem in the Audio Segmentation literature is the lack of openly available datasets because of copyright issues. This is a major hurdle for a researcher to freshly explore audio segmentation. Therefore, this thesis investigates training set synthesis in chapter 3. We replicate the workflow of an audio mixing engineer to generate large-scale training sets.

The audio community for information retrieval has often been inspired by developments in Computer Vision. For instance, harnessing the generalising capabilities of CNNs and treating spectrograms as images. Sometimes, important information such as phase is discarded in the process. Thus, there is growing attention towards end-to-end Deep Learning that trains models directly on raw audio. Interestingly, less attention has been devoted to the output of Deep Neural Networks. Most models perform frame-based classification, where acoustic classes are detected within each frame. However, human annotations contain time-stamps of acoustic boundaries, which are different from frame-based classification. Therefore, in chapter 4, we investigate ways to directly output human-readable labels and minimise the computational overload for post-processing.

In section 2.6, we observed that performance of different machine learning models varies between datasets. For instance, Lemaire & Holzapfel (2019) obtained a Music F-measure of 97.1% on the in-house test set. However, the same algorithm obtained a Music F-measure of 87.9% on the MIREX music-speech detection dataset 2. This shows that shifts in data distribution can lead to a drop in performance of the audio segmentation algorithm. In chapter 5, we investigate how the machine learning model can be made robust to shifts in data distributions.

This thesis also aims to remix broadcast audio in real-time. Hence, the computational load for Music-Speech detection should be minimal, especially if we plan to deploy the remixing engine in frameworks that have less computation power. For flexibility, it would be advantageous to bypass the GPU during inference because it is not present on all machines. In chapter 6, we demonstrate how audio segmentation can be performed on live radio broadcast.

Chapter 3

Artificially Synthesising Training Sets

In this chapter, we answer the first research question — *"What would be an effective way to train machine learning models for audio segmentation with limited access to domain data?"*. The previous chapter recognised the scarcity of data to train machine learning models for audio segmentation. This is primarily due to copyright concerns associated with broadcast material. Moreover, the literature comprises many openly available databases with separate examples of music and speech. These range from genre-recognition datasets for music to speech-recognition datasets for speech examples. However, these datasets contain separate files of music and speech and are not mixed like radio programmes. In broadcast audio, speech and music seamlessly transition between each other and can also occur simultaneously. In this chapter, we present a novel approach to artificially synthesise large-scale training sets for deep learning models. We adopt datasets with separate files of music and speech and automatically mix them in the style of radio DJ by incorporating various fade curves and audio ducking principles. We show that the data synthesis procedure is a highly effective technique to synthesise training sets for audio segmentation. These results were published as a conference paper in IEEE ICASSP (Venkatesh et al., 2021a).

In the second half of this chapter, in sections 3.4, 3.5, and 3.6, we investigate

the effects of training set synthesis. These experiments were published as a journal paper in *Electronics* (Venkatesh et al., 2021b). Firstly, we explore how audio ducking of background music impacts the precision and recall of the machine learning algorithm. Secondly, we examine how the quantity of synthetic training data impacts the results. Finally, we evaluate the effectiveness of synthesised, real-world, and combined approaches for training models, to understand if the synthetic data presents any additional value. Results also show that the minimum level of audio ducking preferred by the machine learning algorithm was similar to that of human listeners. After testing our model on in-house and public datasets, we observe that our proposed synthesis technique outperforms real-world data in some cases and serves as a promising alternative.

3.1 Data Synthesis Procedure

Each audio example synthesised by this technique was 8 s long. We felt that 8 s was long enough to clearly identify an audio class and capture transitions that might occur between one audio class to another. This is similar to the length of audio examples used in the literature. For instance, Gimeno et al. (2020) used 3 s examples and Lemaire & Holzapfel (2019) used 6.3 s examples for Music-Speech detection. Salamon et al. (2017) adopted 10 s examples for environmental sound detection. Please note that in this project, artificial training set synthesis is referred to mixing speech and music files that already exist. We are not generating the sound signal itself. We randomise various parameters within appropriate ranges to replicate the process of a mixing engineer and create a variety of training examples for the neural network.

We considered four combinations of audio classes — speech, music, speech over music, or other. Speech includes the radio DJ speaking about a particular topic, interviews with other individuals, and news, to name a few. Note that music genres such as *a cappella* do not fall under the speech category. It is common for

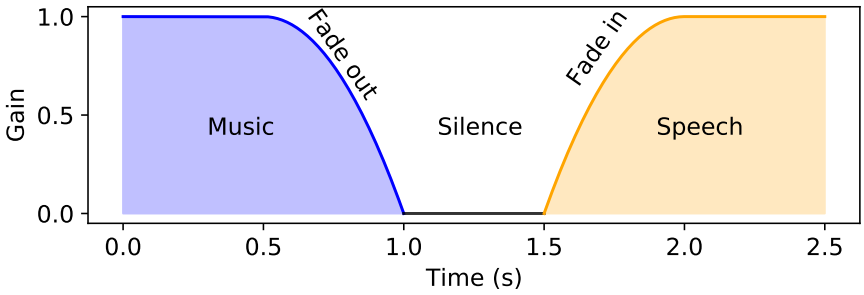
the radio DJ to speak over music for commercials, to introduce a song, or even announce travel news over background music. There are also cases where songs are played back to back without any speech between them. The "other" class in the literature refers to miscellaneous sounds that do not fall under the speech or music categories. This is more common in television broadcast as opposed to radio. It can include special sound effects like clapping, *foley*, and environmental sounds. In radio signals, the "other" class can occur in the form of background noise such as wind when interviewing someone located in a field. Note that a period of silence in the radio also falls within the "other" category.

The examples were categorised into two types — (1) Multi-class examples and (2) Multi-label examples. The former focuses on audio where either music, speech, or noise can occur. There is no simultaneous occurrence of two acoustic classes. Whereas, in multi-label examples, we specifically focus on audio with speech over background music.

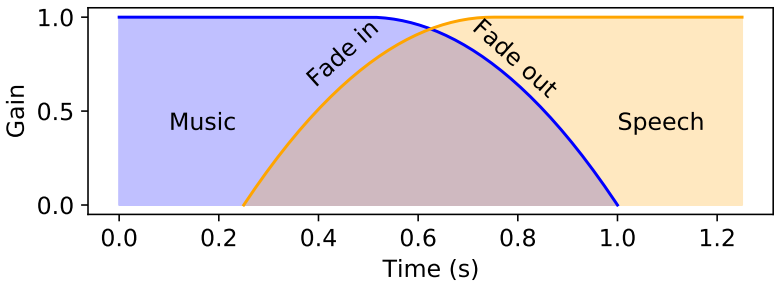
3.1.1 Audio Transitions

Two types of transitions were observed in radio programmes, termed as — (1) normal fade transition and (2) cross-fade transition. In the normal fade, an audio class fades out, followed by a period of silence, and then a new audio class fades in. For example, a Radio DJ introduces a song, followed by a short gap, and then the song starts playing. Moreover, during interviews, the DJ could ask a question, followed by a short period of silence, and then the interviewee answers. In a cross-fade transition, there is a slight overlap between the two classes. As one audio class is fading out, a new class fades in. For instance, one song smoothly cross-fading into another song. Figure 3.1 illustrates the two types of audio transitions.

For multi-class examples, the number of transitions in each synthetic example can either be zero or one. Zero means that there are no transitions at all. This would be a random 8 s segment of purely music, speech, or noise. If there is one



(a) Normal fade



(b) Cross-fade

Figure 3.1: Two types of audio transitions in multi-class examples.

transition, nine permutations can possibly arise as shown in Figure 3.2.

For initial tests, we investigated the possibility of increasing the number of transitions to greater than one, but the program flow became unnecessarily complicated. Furthermore, there is a very minimal chance to have more than one transition within an 8 s window in radio programmes. There is a 50-50% chance for the number of transitions to be zero or one. Noise is less likely to occur in radio programmes. Therefore, we set the probability of speech, music, and noise to be 0.4, 0.4, and 0.2 respectively.

For multi-label examples, as shown in Figure 3.3, we performed audio ducking of background music. Audio ducking is the process of reducing one signal with respect to another, in this case, reducing background music with respect to speech. Again, we can either have no transitions at all or one transition. For simplicity, we did not consider noise for multi-label examples. Therefore, if there are no transitions, there is only one possible combination of audio classes — music+speech. If there is one transition, there are four possible permutations. Please

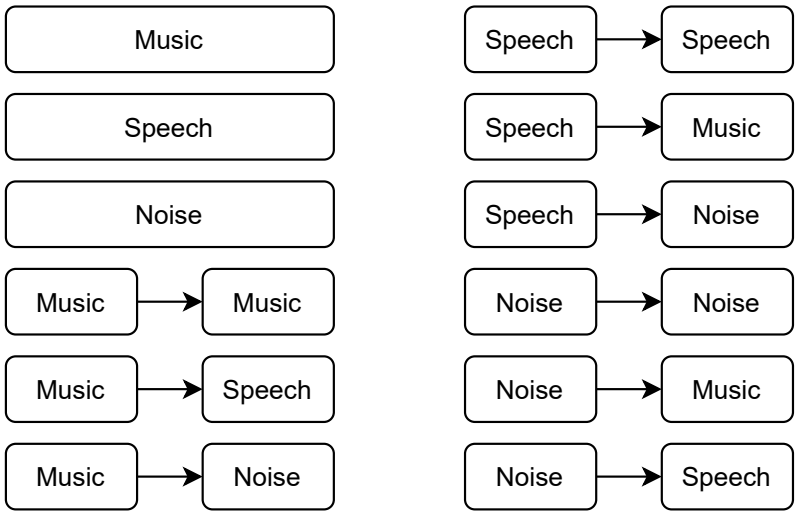


Figure 3.2: The different permutations of audio classes when there are either no transitions or one transition.

refer to Figure 3.4 for an illustration of the different permutations in multi-label examples. There are five different cases:

1. Music+Speech: The entire example contains speech and music occurring simultaneously without any transitions.

2. Music+Speech to Music: Initially, audio ducking is performed on the music and the volume is increased after the speech stops.

3. Music+Speech to Speech: The background music fades out at the transition point. The volume of speech is constant throughout the audio.

4. Music to Music+Speech: Initially, music is being played and ducking is performed when the speech starts.

5. Speech to Music+Speech: Music fades in at the transition point. The volume of speech is constant throughout the audio.

3.1.2 Time-related Variables

There are many time-related parameters involved in the audio mixing pipeline. For example, the duration of a fade curve or the exact time at which a transition

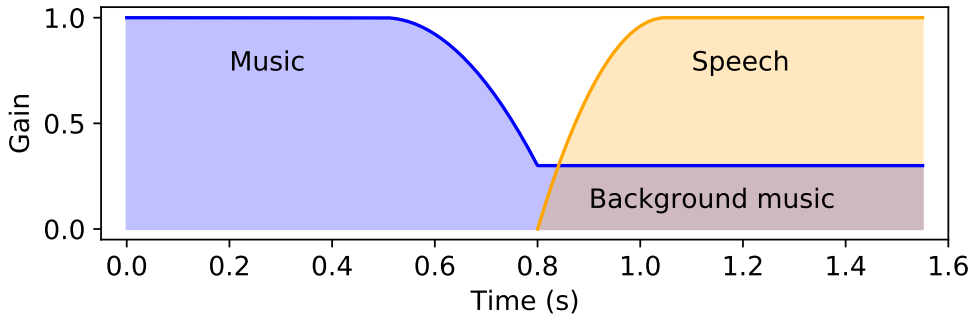


Figure 3.3: An illustration of audio ducking while synthesising multi-label examples.

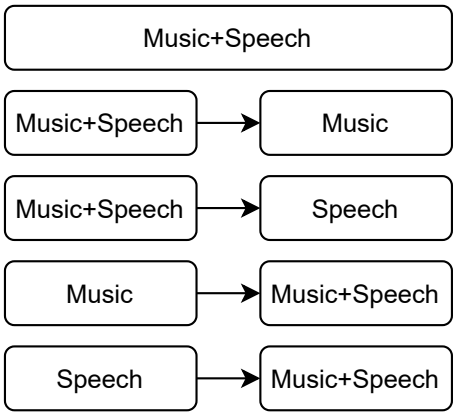


Figure 3.4: The different permutations of audio classes for multi-label examples.

occurs. As our objective is to artificially synthesise a diverse training set, we randomised these parameters. All the random sampling was done using uniform distributions within specified ranges.

Within the duration of an 8 s example, the time-stamp of an audio transition is randomised within the range of 1.5 s to 6.5 s. Subsequently, a fade duration is randomised within a range that is feasible, as explained in the following example. If the time-stamp of transition is at 5 s, the minimum fade-out duration could be 0 s (which is no fade-out) and a maximum of 3 s. This technique helps us render very quick as well as gradual audio transitions.

3.1.3 Fade Curves

Fades are a crucial component in audio mixing. A fade-in gradually increases the volume to the desired level and a fade-out gradually decreases the volume to 0. A fade curve is a mathematical function that defines how the fade is carried out. We considered four popularly used fade curves in the literature (Tarr, 2018) — linear, exponential convex, exponential concave, and s-curve. Figure 3.5 shows the different fade curves.

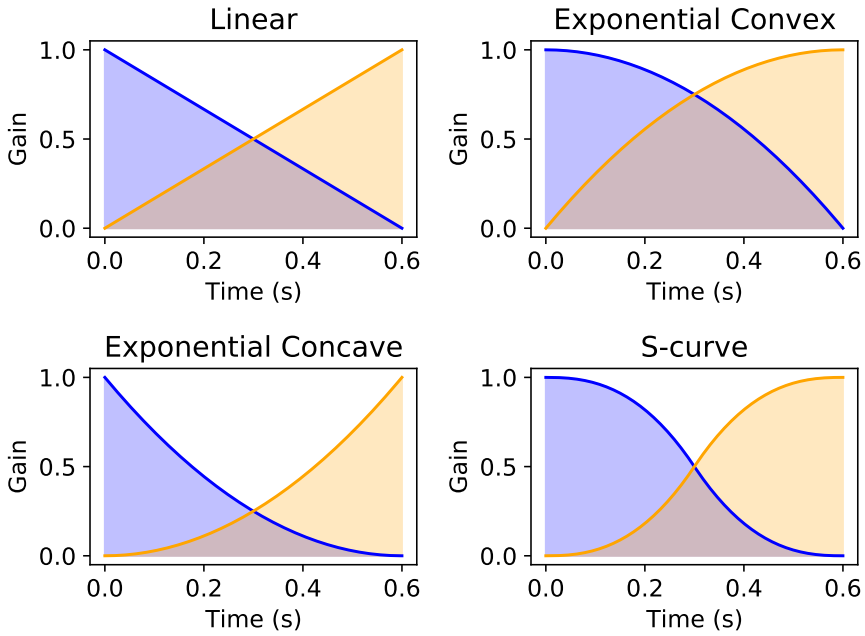


Figure 3.5: The four different fade curves present in audio transitions.

If there is one transition in the synthetic example, there is a fade-out curve as well as a fade-in curve associated with it. To choose a fade-out curve, we randomly select one of the four fade curves, with all of them being equally likely. The fade-in curve is selected the same way. However, the selection of fade-in and fade-out curves is independent of each other. In other words, if the fade-out curve is linear, the fade-in curve could be any of the curves with equal probability. For the exponential convex, exponential concave, and s-curve, there is an additional exponent value that needs to be defined. This value was randomly chosen between 1.5 to 3.0.

3.1.4 Sampling Audio Files

Audio files were arranged in directories such that each audio class has a separate directory. There were three directories — music, speech, and noise. After selecting a permutation of audio classes, we pick a random file from the directory. Subsequently, a random segment from the audio file is extracted. The duration of the segment depends on the position at which the transition occurs. For example, if the transition occurs at 4.6 s and the permutation of audio classes is music to speech, then the duration of the music segment is 4.6 s and the duration of the speech segment is 3.4 s (the duration of each synthetic example is 8 s).

3.1.5 Audio Ducking

Audio ducking is the process of reducing one signal with respect to another. In this context, we reduce the volume of background music with respect to speech. Audio ducking is a common practice in broadcast programmes because it makes speech intelligible. Different radio stations have varying guidelines for mixing engineers to perform audio ducking. Torcoli et al. (2019) conducted a comprehensive analysis of the Loudness Difference (LD) between speech and background music, where LD was measured in Loudness Units (LU). An increase in level by 1 LU leads to an increase of 1 decibel (dB) (Torcoli et al., 2019). Listeners from different backgrounds had varying preferences. On average, LDs preferred by experts were 4 LU less than those preferred by non-experts. In addition, individuals belonging to older age groups preferred greater LDs to clearly understand speech. Figure 3.3 illustrates audio ducking with a defined LD between speech and background music.

The literature does not provide us with an ideal value for LD. It depends on the mixing engineer, target audience, and nature of audio content. Many broadcasters recommend a minimum of 7 to 10 LU for speech over music (Torcoli et al.,

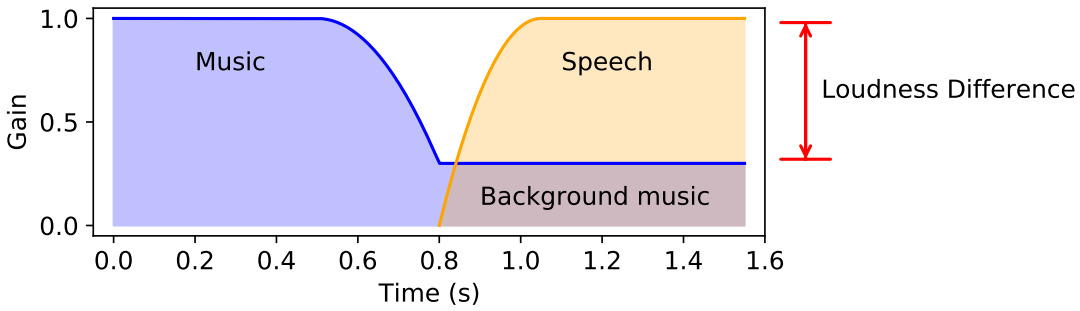


Figure 3.6: A diagram depicting the Loudness Difference between speech and background music.

2019). Others, for instance, the UK Digital Production Partnership, recommends a minimum LD of 4 LU (UK Digital Production Partnership (DPP), 2017).

Higher LDs cause the background music to become quieter. This leads to clearer speech, but the music becomes less impactful. Again, this depends on the nature of audio content. Depending on the programme, the LD could be as high as 23 LU (Torcoli et al., 2019). Moreover, OpenBMAT, a music-detection, contains audio files as low as -51 Loudness Units relative to Full Scale (LUFS).

There are two ways to perform audio ducking—volume automation and side-chain compression. We have adopted the former technique because it is relatively easier to calculate LD values. The loudness of audio was calculated using the integrated loudness metric by ITU-R (2017) BS.1770-4. We adopted the pyloudnorm[1] Python package presented by Steinmetz & Reiss (2021). Note that we had to normalise only one channel because audio files in the data repository were mixed down to mono. Figure 3.7 presents an overview of how the loudness is calculated. The audio is first passed through a K-filter that comprises a high-shelf filter of 1681 Hz and a highpass filter of 38 Hz. Subsequently, the audio is divided in blocks of 400 ms with 75% overlap. The mean square is calculated to obtain the energy of each block. The loudness l of each block j is calculated by equation 3.1.

$$l_j = -0.691 + 10\log_{10} z_j \tag{3.1}$$

[1]pyloudnorm: https://github.com/csteinmetz1/pyloudnorm

The audio blocks are then gated to discard those with very low loudness. The integrated loudness of the entire audio is calculated by averaging the energies of the audio blocks above the threshold. Please refer to ITU-R (2017); Steinmetz & Reiss (2021) for more details of the algorithm.

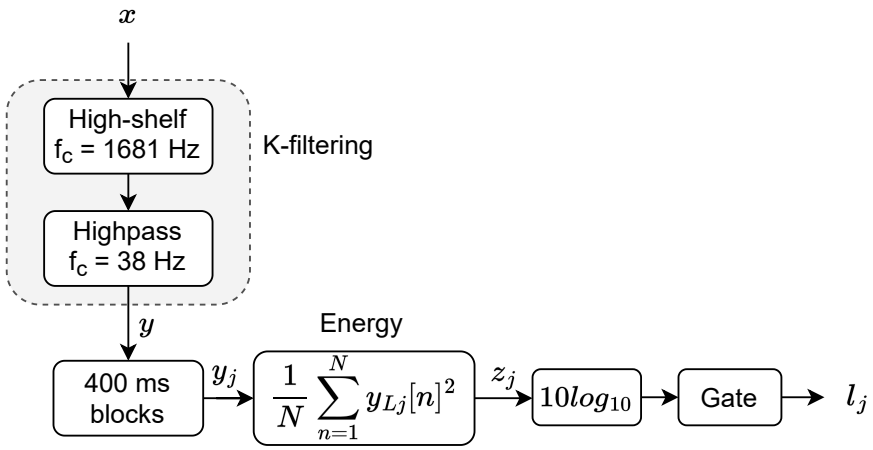


Figure 3.7: An overview of how loudness is measured in the ITU-R BS.1770 system.

During data synthesis, we calculate the loudness of the speech segment. Subsequently, we adjust the gain of background music to the required LD. In this chapter, we evaluate how the machine learning model trains over different ranges of LDs. Section 3.4 presents the methodology for these experiments.

We also surveyed the literature to consider other methods to measure loudness. Wichern et al. (2015) compared different loudness measurement methods for automatic mixing. The study found that a simple energy-based loudness model, which is the ITU-R BS.1770, performed significantly better than sophisticated psychoacoustic models that also included masking effects. The ITU-R BS.1770 has also been adopted by multiple automatic mixing studies (Pestana & Barbosa, 2012; Fenton, 2018) and therefore, we selected this approach for loudness normalisation.

3.1.6 Overview

Figure 3.8 depicts an overview of the data synthesis procedure. Note that in cases where audio ducking is performed, the network needs to predict the presence of

both music and speech. In addition, when an audio class is fading in or out, the entire fade curve is labelled as 1. We do not consider the power of the audio with respect to the mixture gain.

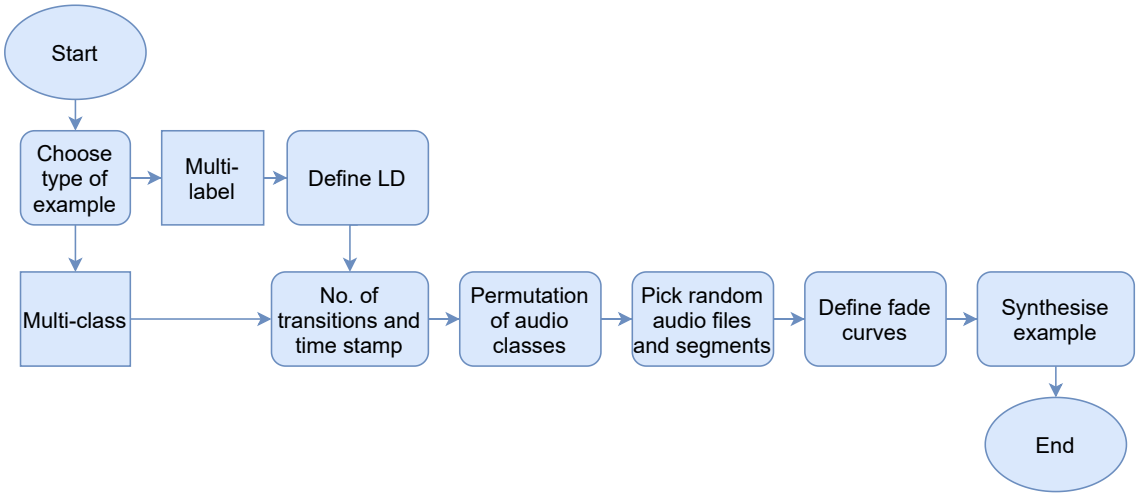


Figure 3.8: A flow diagram depicting an overview of the data synthesis procedure.

Unless mentioned otherwise, the probabilities for all events were equally weighted. For example, the chance of occurrences for multi-label and multi-class examples is 50% each. Four fade curves were considered in the chapter and thus, each fade curve has a probability of 25%. Similarly, the probabilities for other events were calculated based on the total possible number of occurrences.

3.2 Datasets

3.2.1 Data Repository for Data Synthesis

To generate audio that resembles a radio broadcast, we aggregated a data repository that contains audio files labelled as either music or speech. Please note that these files are not mixed and contain only one acoustic class. Among the datasets popularly used for music-speech detection, we included the MUSAN corpus (Snyder et al., 2015), GTZAN music and speech detection dataset (Tzanetakis & Cook, 2000), and the Scheirer and Slaney dataset (Scheirer & Slaney, 1997). After

analysing errors in early informal experiments, without having access to the test set, we observed that speech was confused with wind instruments like the flute. Hence, we included the Instrument Recognition in Musical Audio Signals (IRMAS) dataset (Bosch et al., 2012) which includes many examples of wind instruments and GTZAN genre recognition (Tzanetakis & Cook, 2002) for additional music examples.

It was also challenging for the network to differentiate between singing voice and spoken voice. Therefore, we included a part of the LibriSpeech corpus (Panayotov et al., 2015), which serves as a good collection of speech examples. In addition, vocal sections without musical accompaniment are harder to identify. Thus, we included the Singing Voice Audio Dataset (Black et al., 2014) which contains unaccompanied vocals.

To enable the network to identify task-irrelevant data, we included audio files from the "noise" category in the MUSAN corpus. These noise files were environmental sounds, unintelligible speech, and sound effects, to name but a few. The total number of unique audio files for music, speech, and noise was 6876, 6885, and 665 respectively.

3.2.2 Real-world Radio Data

In this chapter, we aim to use synthetic data for training the neural network. However, this cannot be used for validation and testing because it is not real-world data. We collected 18 h of radio broadcast from BBC Radio Devon, who is our collaborator on the project. For convenience, I split the audio into 1 h files. I manually annotated the audio files for the presence of music and speech. As Meléndez-Catalán et al. (2019) pointed out, annotating an hour of audio takes approximately four to five hours. I took a similar duration to annotate the BBC recordings. I annotated a maximum of one hour of radio recordings per day to minimise fatigue. The annotations were saved as text files. The literature com-

prises other approaches for annotation — (1) JSON Annotated Music Specification (JAMS) proposed by Humphrey et al. (2014) and (2) BMAT Annotation Tool (BAT) developed by Meléndez-Catalán et al. (2017). However, as we are focusing only on two audio classes, I directly used text files for convenience and simplicity.

Three hours of our annotations on the BBC data were verified by an external audio mixing engineer. He was paid for his time and was not involved in the research. Additionally, a random section of 15 min was blind-annotated by him independently. We found an agreement of 99.49% with our annotations by using 10ms segment verifications. This was done to ensure that audio events were similarly perceived by different people.

As we are working with radio data, there are occasional repetitions of audio segments such as jingles. Hence, whenever we split the data for training, validation, and testing, extra care was taken to minimise repetition across the different dataset splits. The exact ratio of splits is explained in the methods section of the respective experiments.

Another dataset that contains real-world data is the MuSpeak dataset (MuSpeak Team, 2015). It contains seven audio files and has a total length of approximately 5 h. Whenever we split the data for training, validation, and testing, we did not split the audio files to prevent the risk of having similar content across different dataset splits.

3.3 Evaluate the Robustness of Data Synthesis

In this section, we investigate if the data synthesis procedure is an effective way to synthesise training sets for audio segmentation. We adopt only synthetic data for training the neural network and real-world data for evaluation. The results presented in this section were published in the proceedings of IEEE ICASSP 2021 (Venkatesh et al., 2021a).

3.3.1 Methods

Pre-processing and Feature Extraction

All the sound files in the data repository and real-world radio recordings were resampled to 22.05 kHz. Stereo signals were converted to mono by averaging the channels. In the data repository, there were many occurrences of unlabelled silence. For example, in speech recordings, long pauses between speech were not labelled. Therefore, we removed/shortened silences in the audio files by using Sound eXchange (SoX)[2].

For our data synthesis to work smoothly, it is convenient to have audio files with a minimum duration of 8 s. Music files in the IRMAS dataset are only 3 s long. In addition, some speech examples in LibriSpeech and noise examples in MUSAN are shorter than 8 s. Therefore, we looped these audio files multiple times to reach the required duration.

We extracted 80 log-scale-mel bands in the range of 64 Hz to 8 kHz from the audio files. The hop size and fast Fourier transform (FFT) size were set to 220 (10 ms) to 1024 (46 ms) respectively. Features were extracted using Librosa (McFee et al., 2015). Audio segments were peak-normalised before passing them through the pipeline for data synthesis. Finally, the whole audio file was peak-normalised after synthesis.

Validation and Test Sets

As mentioned earlier, we cannot use synthetic data for validation and testing. At this juncture of my PhD project, I had finished annotating 9 h of radio data from BBC Radio Devon. The methods adopted for annotation are explained in section 3.2.2. The files were split into files of 1 h duration. Additionally, we included

[2]SoX: http://sox.sourceforge.net/

the MuSpeak dataset (MuSpeak Team, 2015), which approximately contains 5 h of audio. The audio files in both datasets were randomly shuffled and split as approximately 50-50% for validation and test set. Note that we incorporated each audio file as a whole for shuffling and did not split the file.

The above paragraph explained the in-house validation and test set adopted for this experiment. In order to investigate our model's performance on external test sets, we also evaluated it on dataset number 1 of the Music Information Retrieval Evaluation eXchange (MIREX) 2018 music and speech detection competition[3]. This dataset contains 27 hours of audio from various TV programmes. Although our data synthesis was designed for radio programmes, this dataset would provide us with a good evaluation of our model and compare it with state-of-the-art architectures.

Network Architecture

For this study, we adopted a CRNN, which is a state-of-the-art architecture for audio classification and segmentation tasks (Choi et al., 2017b; Cakır et al., 2017). Figure 3.9 shows the network architecture. The input shape of the network was $802 \times 80 \times 1$, equivalent to 802 time-steps and 80 mel bins. The output of the network comprised 802×2 neurons with sigmoid activations, where two neurons perform a binary classification for music and speech at every time step. The network performs multi-output detection, independently detecting the regions of music and speech. This is important for models working with radio data because music and speech can occur simultaneously. Binary cross-entropy was used as the loss function.

We used the Adam optimizer (Kingma & Ba, 2015) with a constant learning rate of 0.001 and batch size of 128. The first two layers of the network were 2D convolutional layers with a kernel size of 7 and a stride of 1. The input was

[3]https://music-ir.org/mirex/wiki/2018:Music_and/or_Speech_Detection

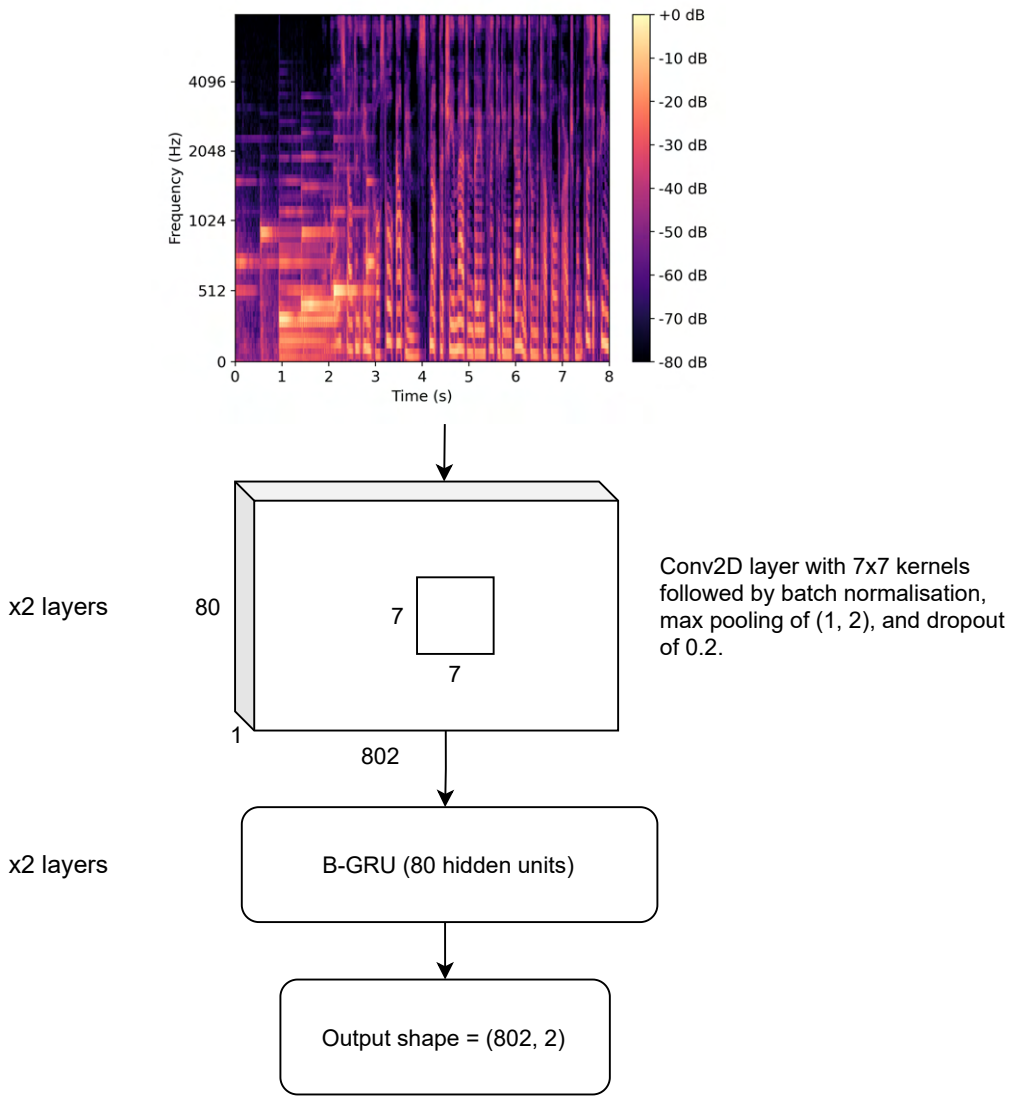


Figure 3.9: The Convolutional Recurrent Neural Network used for music-speech detection. Each audio example is 8 s long. The audio is converted to a mel spectrogram with shape 802×80, with 802 corresponding to the number of time stamps and 80 corresponding to the number of mel bins. The output is 802×2, with one neuron making predictions for speech and the other for music.

padded with zeros such that 'same' convolutions were performed to ensure that the time resolution remains the same. The next two layers were bidirectional gated recurrent units (B-GRU) with 80 units each.

In this study, we evaluated the model using different training sets, which would be explained in section 3.3.1. Hence, a model architecture was finalised by optimising the performance across different training datasets. For regularisation, we implemented early stopping (Yao et al., 2007) and used batch normalisation after all the layers. Max pooling along the dimension of Mel bins was performed

after the convolutional layers. A dropout of 0.2 was added only after the convolutional layers because we observed that it was not effective for the B-GRU layers.

Training Datasets

In order to evaluate the effectiveness of our data synthesis algorithm, we constructed four training datasets. All datasets contain 40960 examples of 8 s audio (which is approximately 91 h of audio). Initial tests conveyed this was an adequate number of examples to train the network.

1. Dataset-only files (d-OF): This dataset contains audio segments of only speech, music, or noise. There was no mixing of audio events within each example. 40960 examples were randomly sampled from our data repository. We did not include the whole corpus because of computational limitations and to manage redundancy.

2. Dataset-only files and background music (d-OFB): In addition to d-OF, this dataset contains examples of speech over background music. The volume of background music was normalised according to the method explained in section 3.1.5. However, this dataset did not contain any audio transitions.

3. Dataset-no normalisation (d-NN): In this dataset, the data synthesis was performed as explained in section 3.1, except for the loudness normalisation of background music according to the loudness of foreground speech. However, all examples of speech, music, and noise were peak-normalised before synthesis.

4. Dataset-data synthesis (d-DS): In this dataset, the data synthesis was performed exactly as explained in section 3.1.

Post-processing and Evaluation

The neural network was designed to make predictions over 8 s audio. During testing, for files that were longer than 8 s, we traversed the audio with a window size of 8 s and a hop size of 6 s. Within each 8 s-window, we discarded the predictions for the first and last second of audio, because the border-predictions might be unreliable. This method was adopted from the paper by Gimeno et al. (2020). Suppose the audio was less than 8 s long; the signal was right padded with zeros to obtain the required duration.

To eliminate spurious transitions, we set thresholds for minimum durations of audio events (Lemaire & Holzapfel, 2019). The minimum durations for speech and music were 1.3 s and 3.4 s respectively. The maximum silence between speech events was 0.4 s. The maximum silence between music events was 0.6 s. These values were adapted from the study by Lemaire & Holzapfel (2019) and were manually adjusted to suit our in-house dataset.

We evaluated our machine learning models using the *sed_eval* toolbox (Mesaros et al., 2016), which has been popularly adopted by the literature (Lemaire & Holzapfel, 2019; Cakır et al., 2017; Mesaros et al., 2017). Segment-level evaluation was performed with a segment size of 10 ms. All experimental results in this paper were presented after smoothing the predictions made by the machine learning models.

3.3.2 Results

Table 3.1 presents the model's performance on different datasets. The highest overall F-measure was obtained by d-DS, which implemented the entire data synthesis procedure. The F-measures of d-OF and d-OFB were at least 3% lower than d-DS because their datasets did not contain audio transitions. This demonstrates that modelling radio DJ-like transitions is an effective technique. Additionally, there

is a marginal difference between d-OF and d-OFB, which explains that adding background music to speech in the training examples is not sufficient, but there need to be audio transitions.

The dataset d-NN contained background music that was peak-normalised, but not normalised with respect to the loudness of foreground speech. Therefore, the music F-measure of d-DS surpasses the value of d-NN by more than 2%. This illustrates that randomising the loudness of background music with respect to foreground speech within an LD of 7 to 18 was an effective method. Speech F-measure for d-NN was slightly greater than d-DS. However, this might be because the background music in d-NN was at a relatively constant volume, which improves speech detection but compromises music detection.

Table 3.1: The F-measure of our CRNN model trained on different datasets. The bold values indicate the largest number in each column.

Dataset	$\mathbf{F_{overall}}$	$\mathbf{F_s}$	$\mathbf{F_m}$
d-OF	93.54	94.58	92.99
d-OFB	93.68	94.95	92.99
d-NN	95.33	**96.44**	94.73
d-DS	**96.69**	96.17	**96.97**

Table 3.2 shows the segment-level evaluation of our d-DS model on the MIREX speech and music detection dataset. The evaluations of other submissions were obtained from the MIREX website. Our model significantly outperforms the other models for F-measures of music. This is attributed to the presence of audio transitions and loudness normalisation of background music in the synthesised dataset. Our model also obtains the highest F-measure for speech detection.

All the other submissions in the competition used real-world data (Choi et al., 2018; Marolt, 2018). Therefore, these results demonstrate that our data synthesis is a highly effective approach for audio segmentation. Moreover, there was another task in MIREX 2018 that was solely for music detection. Our model places second in this task, preceded by the submission by Meléndez-Catalán et al. (2018). Their model was trained on 30 hours of TV programmes, which comes from the same

data distribution. It is important to note that the MIREX evaluation dataset can contain background music over foreground speech, audience noises, sound effects, everyday-life sounds, sounds of the city, and so on. As our data synthesis procedure only considered foreground speech, it explains the poor precision for music in table 3.2. Our model predicted many of the sound effects as music. The performance of our model over TV programmes can be improved by considering these factors in the data synthesis.

Table 3.2: F-measure, precision, and recall of our CRNN model trained on 'd-DS' and other algorithms evaluated on dataset number 1 of MIREX 2018 speech and music detection competition.

Algo.	$\mathbf{F_m}$	$\mathbf{P_m}$	$\mathbf{R_m}$	$\mathbf{F_s}$	$\mathbf{P_s}$	$\mathbf{R_s}$
Choi et al. (2018)	49.36	62.4	40.82	77.18	**96.83**	64.15
Marolt (2018)	38.99	80.72	25.7	91.15	87.95	94.6
Marolt (2018)	54.78	85.7	40.26	90.9	89.45	92.41
Marolt (2018)	31.24	**98.73**	18.56	90.86	83.83	**99.17**
d-DS	**85.76**	79.37	**93.27**	**92.21**	89.71	94.85

3.3.3 Discussion

The results in section 3.3.2 demonstrated the effectiveness of the data synthesis procedure. Only artificially synthesised data was used to train a model for audio segmentation and classification. We adopted a training dataset belonging to a different distribution from the validation and test sets. Despite this, we obtained a high F-measure on our local test set. Furthermore, we obtained state-of-the-art performance for speech and music detection on the MIREX 2018 competition dataset.

As labelling large amounts of data is an expensive and time-consuming task, our data synthesis procedure serves as a potential solution to generate large amounts of training data. In the next section, we investigate the effect of audio ducking for training set synthesis. We randomised the LD between 7 and 18 by conducting surveying the literature and informal tests. Next, we evaluate how different ranges of audio ducking impact the performance of the machine learning

model.

3.4 Loudness Difference Selection

The results presented in this section were published in *Electronics*, as a journal paper in a special issue on Machine Learning Applied to Music/Audio Signal Processing (Venkatesh et al., 2021b). For further experiments in this chapter, the training, validation and test sets were re-organised because an additional 9 h of radio recordings were labelled and used for the study. This was done to compare real-world training sets and artificial training sets, which is explained in section 3.6. The data was reorganised to minimise potential repetitions in radio programmes across dataset splits. An overview of the contents of the validation and test splits in the dataset can be found in table 3.3. The training sets differ for each experiment and are detailed in the relevant sections.

Table 3.3: Contents of validation, and test datasets for music-speech detection. Real-world radio data was collected from BBC Radio Devon. MuSpeak (MuSpeak Team, 2015) is an openly available dataset containing annotations for music and speech.

Dataset Division	Contents
Validation	5 h from BBC Radio Devon + 2 h from MuSpeak
Test	4 h from BBC Radio Devon + 1 h 42 min from MuSpeak

3.4.1 Neural Network Architecture

The neural network architecture used for this experiment was different from the one used in the previous section. This is because we had conducted a different set of experiments before arriving at the architecture. We compared state-of-the-art architectures and performed hyperparameter-tuning using an automatic tuning technique called Hyperband (Li et al., 2017). The details of these experiments are

presented in chapter 4 in section 4.1. We would not go into the details here because this chapter focuses on training set synthesis. The architecture is briefly explained in table 3.4.

Table 3.4: The neural network architecture used for this experiment. More details are given in chapter 4.

Parameter	Value
No. of Conv. layers	3
Kernel size	{3, 11, 11}
No. of filters	{128, 128, 6}
No. of B-GRU layers	2
No. of B-GRU Units	{80, 40}

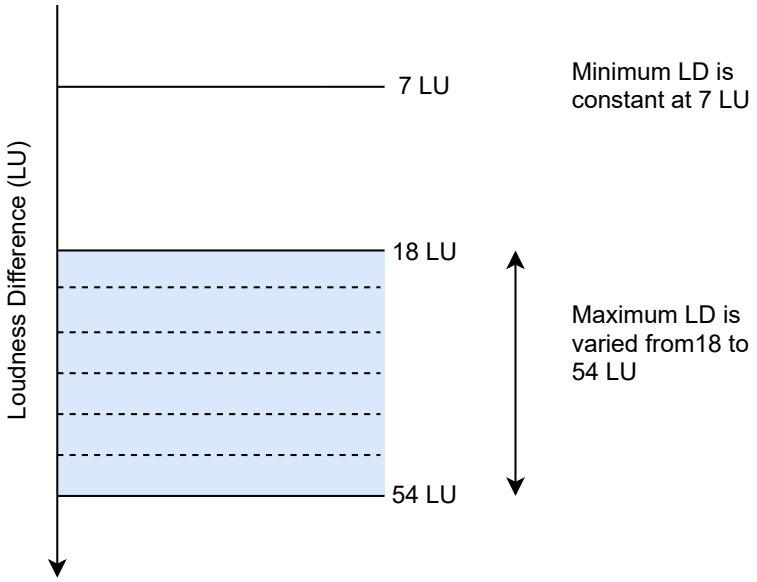
The network adopted the Adam optimiser (Kingma & Ba, 2015) with an initial learning rate of 0.001. Early stopping (Yao et al., 2007) with a patience of 20 epochs was implemented during training. In addition, we decayed the learning rate by a factor of 0.84 after 10,000 weight updates. The batch size was set to 128. Furthermore, in this network, we adopted Layer Normalisation (Ba et al., 2016) instead of Batch Normalisation (Ioffe & Szegedy, 2015). The motivation behind doing so is explained in chapter 4.

3.4.2 Experimental Set-Up

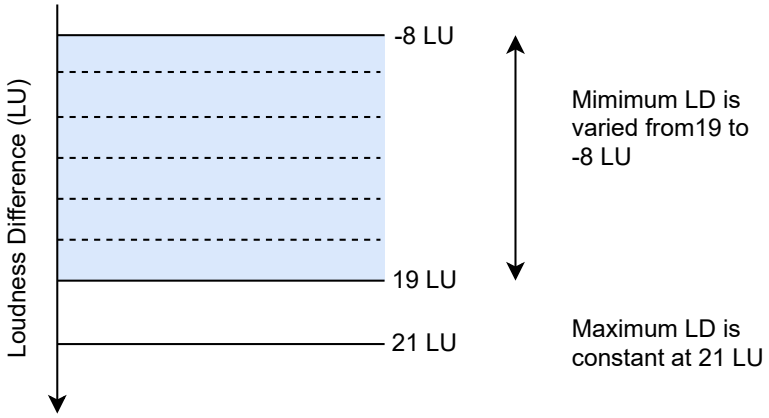
In this experiment, we used only synthetic data to train the neural network. For each audio example with speech over background music, we need to select an LD between the audio classes. This LD cannot be constant because the neural network will become biased towards learning a specific LD. Therefore, we chose random LD values from a uniform distribution. However, the literature does not provide us with a clear-cut range of LDs. For our data synthesis procedure, we need to select an optimal maximum and minimum value of LD.

First, we set the minimum value at 7 LU. The maximum value was varied from 18 to 54 LU with steps of 3 LU, as shown in Figure 3.10a. We synthesised 5120 examples and trained the network over these examples for each configuration. The

choice of maximum LD is expected to only influence the performance on music. The greater the LD, the lower the volume of background music and vice versa. If the background music is sufficiently loud, we can precisely detect the presence of music. However, if the background music becomes too low, it becomes harder for the listener to precisely detect the presence of music. Hence, we analysed the precision, recall, and F-measure of music. We repeated the experiment five times using different random seeds. Regression analysis was performed on the observations using SPSS (IBM Corporation, 2017). Analysis of variance (ANOVA) was conducted to evaluate the level of significance.



(a) The minimum LD is fixed at 7 LU. The maximum LD is varied from 18 to 54 LU.



(b) The maximum LD is fixed at 21 LU. The minimum LD is varied from 19 to -8 LU. Note that negative values of LD indicate that the background music is louder than speech.

Figure 3.10: Experiments conducted to find an optimal maximum and minimum value of LD.

Similarly, we set the maximum value at 21 LU and the minimum value was varied from 19 to -8 LU with a step of -3 LU, as shown in Figure 3.10b. The smaller the LD, the louder the background music is. Negative LDs stand for cases when the background music is louder than the speech. Therefore, if the LD is too low or negative, we expect the precision of speech to be hindered. This often occurs in advertisements and radio jingles, which have smaller LDs. Again, for each configuration, we synthesised 5120 examples and trained the network over these examples. The minimum LD is expected to only influence the intelligibility of speech. Hence, we analysed the precision, recall, and F-measure of speech. We repeated the experiment five times. Regression analysis and ANOVA were performed on the observations.

3.4.3 Results and Discussion

F-measure

F-measure is a metric that combines precision and recall by calculating their harmonic mean (Sokolova & Lapalme, 2009). Figure 3.11 presents a quadratic regression of how the F-measure of music changes with maximum LD. The maxima for validation, test, and combined curves lie at 23, 27, and 24 LU respectively. The validation and test curves were significant ($p < 0.001$). However, the combined curve that jointly plots validation and test observations was not significant ($p > 0.05$). Therefore, our results suggest that the optimal value of maximum LD lies somewhere between 23 and 27 LU.

Torcoli et al. (2019) suggested that depending on the nature of broadcast content, the LD can be as high as 23 LU, which has a reasonable overlap with our findings. However, we did not find a single optimal value for maximum LD based on F-measure but a range of values from 23 to 27 LU.

Figure 3.12 shows a regression analysis of how the F-measure of speech varies

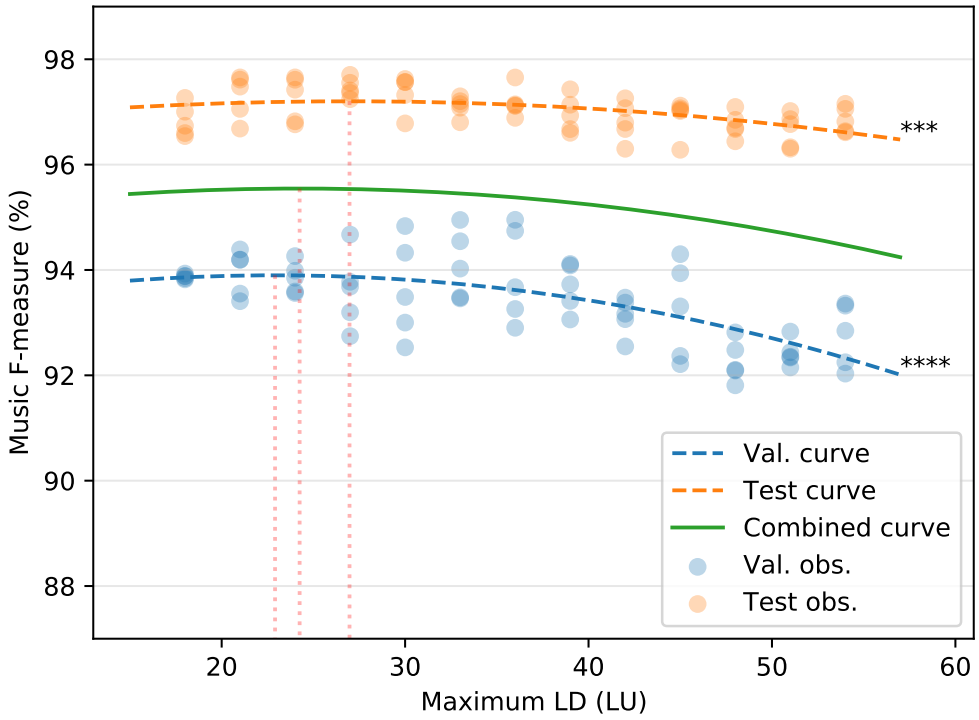


Figure 3.11: F-measure for different values of maximum loudness difference (LD). The minimum LD was fixed at 7 LU. Validation observations (Val. obs.) and test obs. are the actual values obtained in experiments. The val. and test curves were generated through quadratic curve fitting. The combined curve aggregates the observations on val. and test set for quadratic curve fitting. The vertical dotted lines in red indicate the maxima of the curves. ANOVA was performed on the observations and the asterisks indicate the level of significance (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$, ****: $p < 0.0001$).

with minimum LD. The maxima for validation, test, and combined curves lie at 8, 2, and 5 LU respectively. The validation and test curves were significant ($p < 0.0001$). However, the combined curve that jointly plots validation and test observations was not significant ($p > 0.05$). Therefore, our results suggest that the optimal value of minimum LD lies somewhere between 2 and 8 LU.

As explained in Section 3.1.5, many broadcasters recommend a minimum of 7 to 10 LU for speech over music and some recommend a minimum LD of 4 LU (Torcoli et al., 2019; UK Digital Production Partnership (DPP), 2017). Note that these are only recommendations to make speech intelligible. However, there are cases where mixing engineers choose LDs close to zero or even negative LDs (Torcoli et al., 2019). Therefore, in order to maximise F-measure, our results suggest that the minimum LD should lie between 2 and 8 LU.

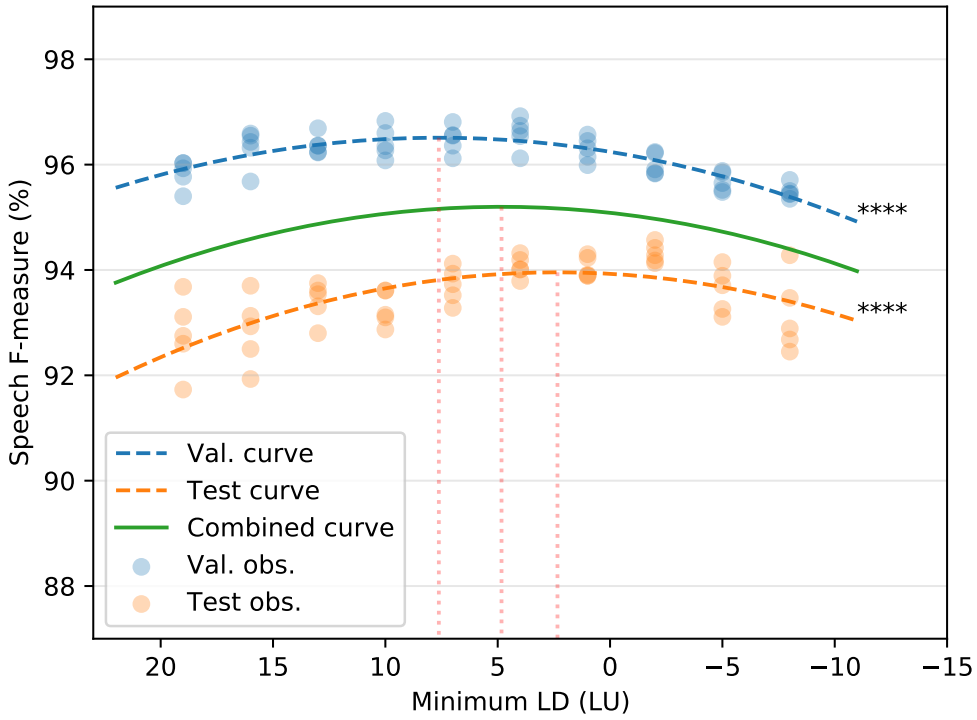
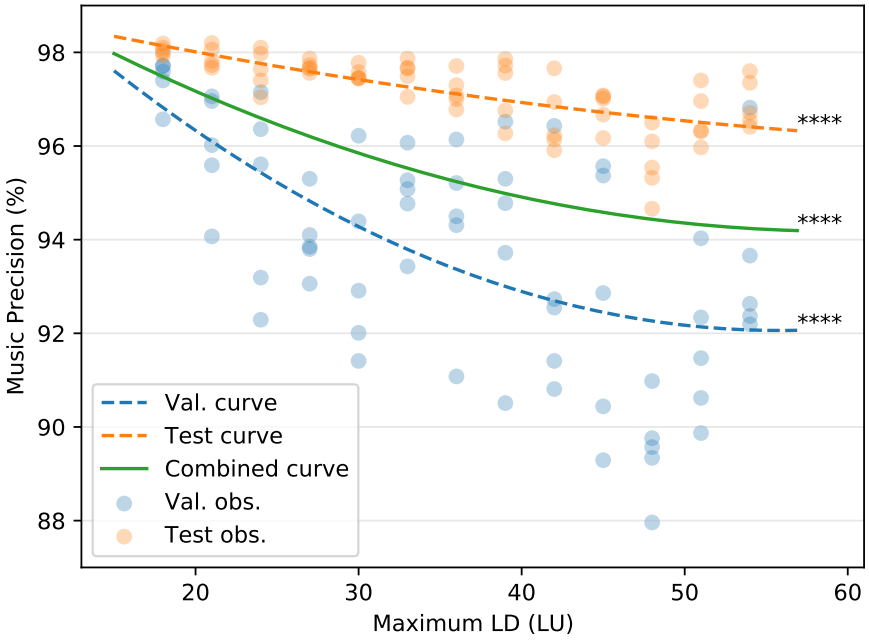


Figure 3.12: F-measure for different values of minimum LD. The maximum LD was fixed at 21 LU. Validation observations (Val. obs.) and test obs. are the actual values obtained in experiments. The val. and test curves were generated through quadratic curve fitting. The combined curve aggregates the observations on val. and test set for quadratic curve fitting. The vertical dotted lines in red indicate the maxima of the curves. ANOVA was performed on the observations and the asterisks indicate the level of significance (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$, ****: $p < 0.0001$).

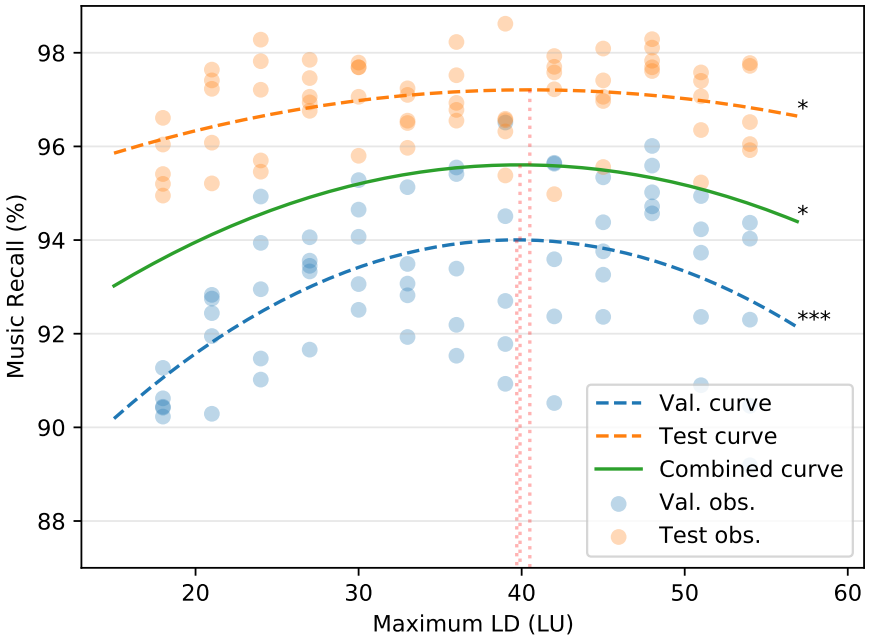
As the F-measures of combined curves were statistically insignificant, it may benefit from further analysis. Are we overfitting the validation set? Or is there an optimal LD for different datasets? To address the questions, we analysed precision and recall in the following subsection.

Precision and Recall

Figure 3.13a presents a regression analysis of how the precision of music varies with maximum LD. A quadratic curve was fitted to the observations. All curves evidently demonstrate that precision decreases as the maximum LD increases ($p < 0.0001$). Therefore, if the neural network was trained on examples that have very low volumes of background music, the precision is hindered.



(a) Precision of music.



(b) Recall of music. The vertical dotted lines in red indicate the maxima of the curves.

Figure 3.13: Evaluation of different values for maximum LD. The minimum LD was fixed at 7 LU. Validation observations (Val. obs.) and test obs. are the actual values obtained in experiments. The val. and test curves were generated through quadratic curve fitting. The combined curve aggregates the observations on val. and test set for quadratic curve fitting. ANOVA was performed on the observations and the asterisks indicate the level of significance (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$, ****: $p < 0.0001$).

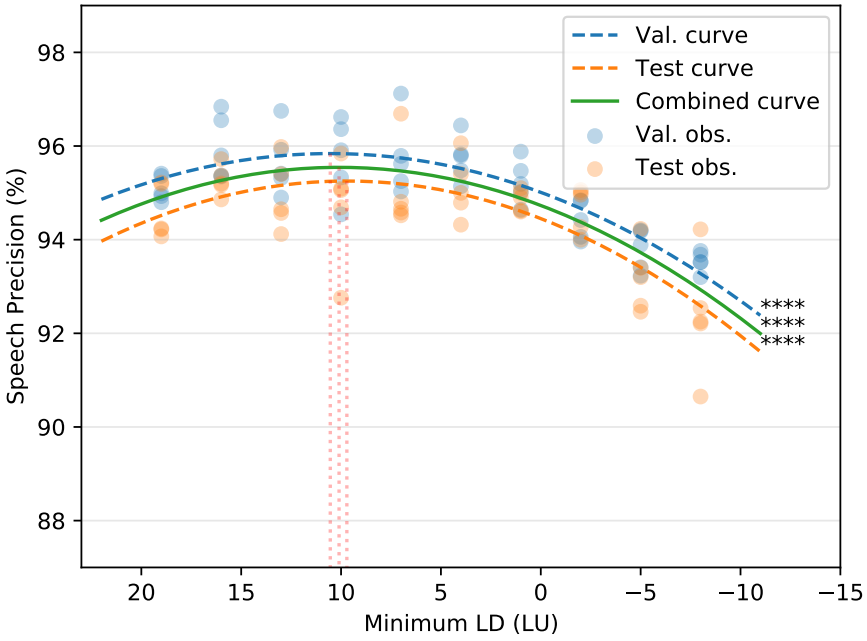
On the other hand, Figure 3.13b shows the relationship between recall of music and maximum LD. Initially, recall increases as the LD increases. However,

the maxima for the validation, test, and combined curves lie approximately at 40 LU. This shows that the recall does not increase beyond an LD of 40 LU. In other words, the background music becomes too low to be perceived. Hence, we recommend that the maximum LD should never be greater than 40 LU, even if the researcher desires a high recall.

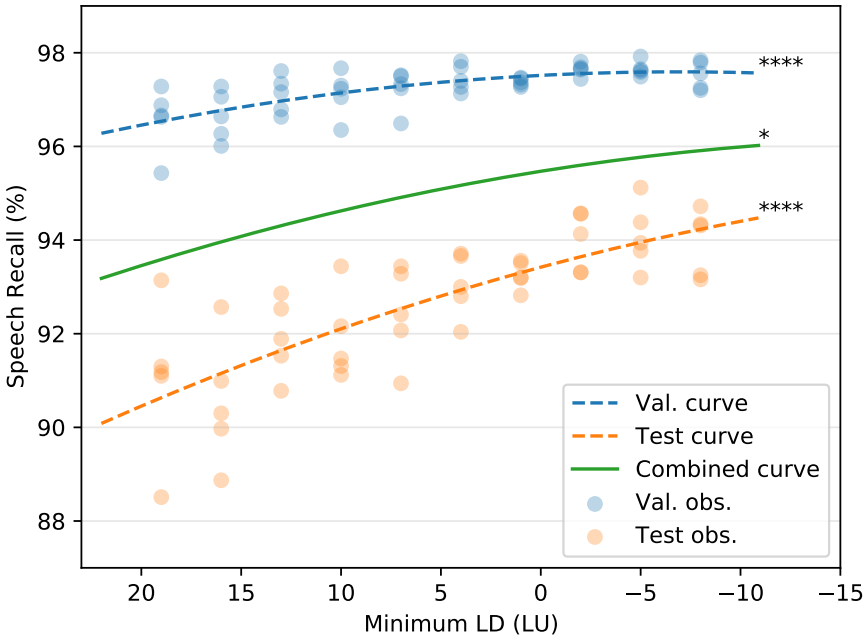
Figure 3.14a shows a regression analysis of speech precision with regards to minimum LD. Initially, as the minimum LD decreases, there is an increase in precision. The maxima for all the curves lie at approximately 10 LU. Below 10 LU, there is a decrease in precision. This is an interesting observation because the same LD is preferred by human listeners (Torcoli et al., 2019). Torcoli et al. (2019) conducted a study on the LDs preferred by human listeners. Based on the test results, they recommended a minimum of 10 LU between speech and background music. Contrarily, many broadcasters use LDs less than 10 LU, which makes speech intelligible (Torcoli et al., 2019). It is noteworthy that machine learning models in this study also preferred a minimum LD of 10 LU to maximise the precision of speech.

Figure 3.14b shows that recall continuously improves with a decrease in minimum LD. This is because the machine learning model learns more examples where speech and background music have a small LD. This also shows that the real-world radio examples collected from BBC Radio Devon contain cases where speech and background music have small LDs. However, as shown in Figure 3.14a, the precision of speech is affected in these cases.

The results in this subsection and Section 3.4.3 clearly demonstrate that there is a trade-off between precision and recall when selecting a maximum and minimum LD. A researcher may adjust these settings according to their objectives. Moreover, F-measure assigns an equal weightage to precision and recall, which might not be desirable. For further experiments in our study, we did not make any decisions based on the regression analysis because it includes the test set. Instead, we chose values that obtained the highest mean for F-measure on our validation set. The



(a) Precision of Speech. The vertical dotted lines in red indicate the maxima of the curves.



(b) Recall of Speech.

Figure 3.14: Evaluation of different values for minimum LD. The maximum LD was fixed at 21 LU. Validation observations (Val. obs.) and test obs. are the actual values obtained in experiments. The val. and test curves were generated through quadratic curve fitting. The combined curve aggregates the observations on val. and test set for quadratic curve fitting. ANOVA was performed on the observations and the asterisks indicate the level of significance (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$, ****: $p < 0.0001$).

minimum and maximum LD were set to 4 and 33 LU respectively.

3.5 Impact of Dataset Size

3.5.1 Experimental Set-Up

As we are artificially synthesising training data, we have the flexibility to create large training sets. To develop an understanding of how the neural network's performance improves with the size of the training set, we evaluated the performance over 5120, 10,240, 20,480, and 40,960 examples. To evaluate the significance level, a two-way ANOVA was performed with dataset size and dataset type (validation and test) as factors. A posthoc Tukey test was performed to make pairwise comparisons.

The number of unique speech and music files in the data repository is only 6885 and 6876 respectively. In order to manage redundant examples, we adopted a new training strategy. For every training cycle, we randomly select 40 mini-batches and calculate the error on the validation set. This way, we prevent the risk of overfitting within epochs. This training strategy was adapted from the study by Schlüter & Grill (2015). However, we used the Adam optimiser with the same learning rate schedule explained in Section 3.4.1.

3.5.2 Results

Figure 3.15 plots the overall F-measure against different training set sizes. We did not observe any patterns in precision and recall because in this section, the minimum and maximum LD values were constant at 4 and 33 LU. As we increased the size of the training set, we observed a slight increase in F-measure on the validation set. However, on the test set, an increase was found from 5120 to 10,240 examples but not in subsequent intervals. ANOVA showed that the dataset size had a significant effect on the F-measure ($p < 0.05$). Posthoc tests showed that there

were significant differences for two pairs—(5120 & 10,240) and (5120 & 40,960). All the other pairs were statistically insignificant. Therefore, we can conclude that the performance does not significantly improve beyond 10,240 training examples. This might be due to our data repository comprising approximately 6800 unique examples for each audio class.

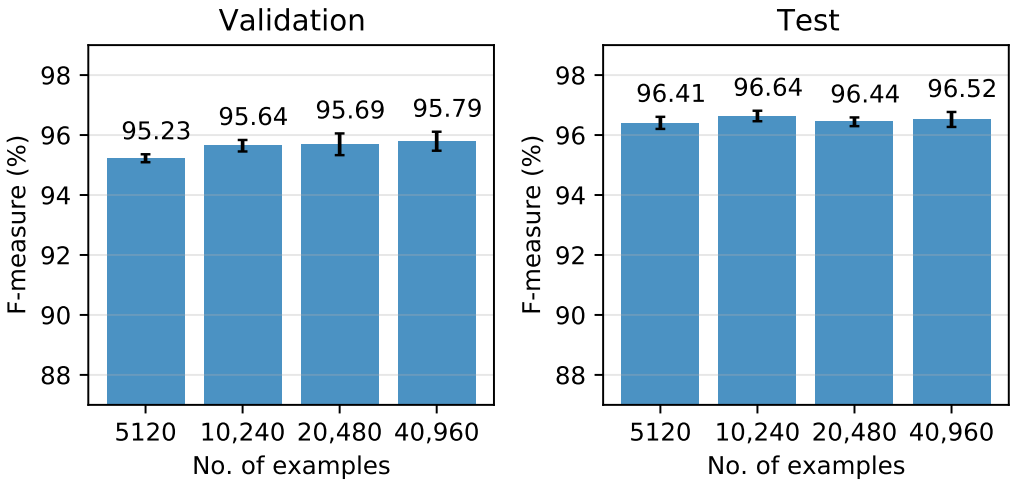


Figure 3.15: The overall F-measure for different training set sizes. These results were calculated on the validation set and test set. The bar chart shows the mean and standard deviation for five iterations of data synthesis and training using different random seeds.

For the next experiment in our paper, we used 40,960 training examples instead of 10,240 examples. This is because 40,960 examples obtained the highest F-measure on our validation set and we did not make any decisions based on the test set.

3.6 Comparison of Real-World and Artificial Data

3.6.1 Experimental Set-Up

To evaluate the effectiveness of our data synthesis algorithm, it is important for us to compare the performance with real-world data. Therefore, we trained the model on four different training sets as shown below. Each of them comprised 40,960 8 s-long audio examples.

1. SSE: Sound segment examples (SSE) consisted of audio directly sampled from the data repository. This does not contain mixed audio.

2. SSE+RRE: This is the combination of SSE and Real-world radio examples (RRE). This is the type of training set used by most audio segmentation studies.

3. ARE+RRE: In this set, we used a combination of Artificial Radio Examples (ARE), which are synthesised using our data synthesis procedure and RRE.

4. ARE: Here, we used only ARE for training the model.

During training, to manage redundant examples, we adopted the same training strategy as explained in Section 3.5.1. All four models were tested on our in-house dataset and the MIREX competition dataset. Similar to earlier experiments, we trained models for five iterations of data synthesis using different random seeds. However, we created an ensemble of the five models. A simple average (Breiman, 1996; Goodfellow et al., 2016) over the sigmoid outputs of the five networks was calculated to obtain the final prediction. By creating an ensemble, we would obtain a single number evaluation on our test set instead of means and standard deviations. Therefore, we can compare our models with state-of-the-art algorithms submitted to the MIREX competition, which adopt the same testing paradigm. Moreover, a group of networks reduces generalisation error by combining predictions (Breiman, 1996; Goodfellow et al., 2016).

3.6.2 Results

The models were trained using four different training sets — SSE, SSE+RRE, ARE, and ARE+RRE. Table 3.5 shows the evaluation on our in-house dataset. SSE and ARE comprise examples from the same data repository. However, ARE increases the overall F-measure by 4.5%. This demonstrates the effectiveness of the data synthesis algorithm.

SSE+RRE is the type of training set generally used by audio segmentation studies (Lemaire & Holzapfel, 2019). Interestingly, the overall F-measure of ARE also surpasses SSE+RRE. Considering the fact that RRE belongs to the same data distribution as the test set, this is a significant development. ARE+RRE evidently outperforms the other models. It harnesses the advantage of both synthetic radio examples and real-world data.

Table 3.5: This evaluation of precision, recall, and F-measure for speech and music was conducted on our in-house test set. It compares our model trained on different training sets. The bold values indicate the largest number in each column.

Training Set	$\mathbf{F_{overall}}$	$\mathbf{F_s}$	$\mathbf{P_s}$	$\mathbf{R_s}$	$\mathbf{F_m}$	$\mathbf{P_m}$	$\mathbf{R_m}$
SSE	92.23	88.21	**97.34**	80.64	93.84	**98.06**	89.97
SSE+RRE	96.64	93.78	95.11	92.49	97.81	97.3	98.33
ARE	96.89	94.73	96.64	92.9	97.77	97.72	97.82
ARE+RRE	**97.35**	**95.05**	96.21	**93.92**	**98.3**	97.73	**98.87**

The upper half of Table 3.6 presents the results of our models on the MIREX competition dataset. We observe similar trends as the results in Table 3.5—ARE outperforms SSE and SSE+RRE. Interestingly, ARE has a higher F-measure for music than ARE+RRE but obtains a lower speech F-measure. It is important to note that in this case, the training and test datasets were annotated by different research groups. Therefore, there would be disagreements in annotations for low volumes of background music. This trend was also observed in the OpenBMAT dataset (Meléndez-Catalán et al., 2019), which had low agreement percentages across annotators for background music in very low volumes. The advantage of artificially synthesising data is that it is independent of this human error in labelling. Contrarily, in speech annotations, there is less scope for disagreements. Therefore, ARE+RRE has a higher Speech F-measure than ARE.

The lower half in Table 3.6 shows the previously published results obtained from the MIREX website. The music F-measure of all our models surpassed the performance of earlier submissions. This can be attributed to the variety of datasets in our data repository. The submissions by Marolt et al. Marolt (2018) surpassed three of our models in the speech F-measure. However, ARE+RRE obtained a

Table 3.6: This evaluation was conducted on the MIREX competition dataset. The upper half compares our model trained on different training sets. The bottom half shows the previously submitted algorithms (Algo.) to the competition. The results were obtained from the MIREX website. Venkatesh et al. (2021a) was the earlier study that adopts our data synthesis procedure, as explained in section 3.3.2. '-' indicates that $F_{overall}$ was not calculated for the submission. The bold values indicate the largest number in each column.

Training Set/Algo.	$F_{overall}$	F_s	P_s	R_s	F_m	P_m	R_m
SSE	79.61	81.79	89.16	75.54	76.27	**86.62**	68.12
SSE+RRE	86.57	88.15	92.48	84.2	84.3	85.33	83.3
ARE	88.52	90.73	91.96	89.53	**85.51**	79.55	92.43
ARE+RRE	**89.09**	**92.16**	**92.64**	**91.69**	85.01	77.22	**94.55**
Choi et al. (2018)	-	77.18	**96.83**	64.15	49.36	62.4	40.82
Marolt (2018)	-	91.15	87.95	94.6	38.99	80.72	25.7
Marolt (2018)	-	90.9	89.45	92.41	54.78	85.7	40.26
Marolt (2018)	-	90.86	83.83	**99.17**	31.24	**98.73**	18.56
Venkatesh et al. (2021a)	**89.53**	**92.21**	89.71	94.85	**85.76**	79.37	**93.27**

higher speech F-measure.

The ARE model in Table 3.6 obtained slightly poorer results than the one presented in our previous study (Venkatesh et al., 2021a), as explained in section 3.3.2. This is because we used a different neural network. In the current section, hyperparameter tuning was performed on ARE+RRE by using Hyperband (more details are available in chapter 4). However, in the previous study (Venkatesh et al., 2021a), we manually optimised the network for only ARE. Additionally, we had included dropout for regularisation and used Batch Normalisation instead of Layer Normalisation. This suggests that it might be good to include some form of regularisation such as dropout so that the neural network performs better on data distributions it has not seen before.

3.7 Discussion

In this chapter, we demonstrated the efficacy of our data synthesis procedure. In section 3.3, only artificially synthesised data was used to train the model. We adopted a training dataset belonging to a different distribution from the validation

and test sets. Despite this, we obtained a high F-measure on our local test set. Furthermore, we obtained state-of-the-art performance for speech and music detection on the MIREX 2018 competition dataset.

In sections 3.4, 3.5, and 3.6, we investigated the effects of using synthetic data. Section 3.4 investigated the impact of LD between speech and background music while training neural networks. There was a trade-off between precision and recall when selecting maximum and minimum LDs. Moreover, if the LD was less than 10 LU, the precision of the network gets affected. This choice of minimum LD for the machine learning model was similar to that of human listeners in the literature (Torcoli et al., 2019). This paper also recommended that the LD between speech and background music should not be greater than 40 LU because the music becomes too quiet to be detected. This emphasises that the task of background music-detection needs to be defined in greater detail by using threshold-related variables.

In section 3.6, we compared the effectiveness of real-world and synthetic training sets. Interestingly, artificial data surpasses the performance of real-world data in some scenarios. It generalises better to other data distributions because it is less prone to human error and subjective disagreements between annotators.

There are noticeable differences between the BBC Radio Devon recordings and the data repository we have used for data synthesis. The BBC recordings have greater dynamic range compression, cleaner speech, and generally use side-chain compression for audio ducking. Therefore, including a small number of radio recordings in the training dataset might improve the model's performance. Additionally, incorporating audio effects like dynamic range compression in the data synthesis pipeline might improve the model's performance.

As this chapter has significantly reduced the time and resources required to label datasets, it opens up many possibilities for future work. Audio features like Mel spectrograms and MFCCs discard the phase of the audio and only consider its

magnitude. This might hinder the algorithm's ability to capture audio transitions. End-to-end deep learning, as suggested by some researchers Lemaire & Holzapfel (2019); Lee et al. (2017), seems like a promising approach to audio segmentation. The results in Section 3.5.2 showed that the performance did not significantly improve beyond 10,240 examples. This suggests that other methods such as Generative Adversarial Networks (GANs) Goodfellow et al. (2014); Yang et al. (2018) might improve the quality of synthetic training sets.

3.8 Publications, Code, and Contributions

- Venkatesh, S., Moffat, D., Kirke, A., Shakeri, G., Brewster, S., Fachner, J., Odell-Miller, H., Street, A., Farina, N., Banerjee, S., et al. (2021a). Artificially synthesising data for audio classification and segmentation to improve speech and music detection in radio broadcast. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Ontario, Canada*, (pp. 636–640). doi: 10.1109/ICASSP39728.2021.9413597.

  – In this conference paper, we presented the novel data synthesis procedure for audio segmentation of radio broadcast. We show that the use of fade curves and audio ducking improves the performance of the algorithm. We obtained state-of-the-art performance for music-speech detection on in-house and public datasets. All the code associated with this study is available at this GitHub repository (https://github.com/satvik-venkatesh/audio-seg-data-synth/). During peer review, one of the reviewers suggested the idea of investigating the parameters of data synthesis itself. This motivated us to extend the study to a journal paper (Venkatesh et al., 2021b), which would comprehensively evaluate the parameters of data synthesis.

- Venkatesh, S., Moffat, D., & Miranda, E. R. (2021b). Investigating the effects of training set synthesis for audio segmentation of radio broadcast. *Electronics*, *10*(7), 827. doi: 10.3390/electronics10070827.

– This journal paper was published in a special issue for Machine Learning Applied to Music/Audio Signal Processing. In this paper, we investigated the various effects of data synthesis, such as audio ducking and dataset size. One of the most interesting findings was that minimum level of audio ducking preferred by the machine learning algorithm was similar to that of human listeners. In addition, we compare real-word training sets and artificial training sets and understand the advantages of the latter. All the code associated with this study is available at this GitHub repository (https://github.com/satvik-venkatesh/train-synth-audio-seg/). There were many helpful comments received during the peer review process, such as evaluating the statistical significance of experiments in sections 3.4 and 3.5. Only in the second revision of the paper, we explored the idea of performing quadratic regression analysis for the experiments on the loudness difference between speech and music. To summarise, the novel contributions of this study were: (1) compare state-of-the-art neural network architectures for audio segmentation, (2) investigate how the loudness difference between speech and background music influences the performance of segmentation, (3) examine how the size of the training set improves performance (4) compare real-world training sets and synthetic training sets.

Chapter 4

Neural Network Architectures for Audio Segmentation

In this chapter, we explore model-centric approaches to improve music-speech detection, which addresses the second research question — *"How can we advance state-of-the-art algorithms by investigating novel pattern recognition problems?"*. First, we systematically compare deep learning architectures commonly adopted for this task in the literature — CNN, GRU, LSTM, TCN, and CRNN. Results showed that CRNN is the best performing model. Subsequently, we explored the benefits of using raw audio for audio segmentation. As training on raw is different from mel spectrograms, we evaluated different architectures such as CRNN and Wave-U-Net (Stoller et al., 2018). However, the performance of the machine learning model did not match the performance on mel spectrograms and conveyed that the benefits are limited for audio segmentation.

Lastly, we propose a novel algorithm called the You Only Hear Once (YOHO), which is inspired by the You Only Look Once algorithm in Computer Vision. We convert audio segmentation from a classification problem to a regression problem, making the pipeline more end-to-end. We evaluate YOHO on multiple datasets

for sound event detection and found that it generalises better than the CRNN. As it predicts acoustic boundaries directly through regression, it is about seven times faster faster than segmentation-by-classification. The work on YOHO was published as a journal paper in *Applied Sciences* (Venkatesh et al., 2022b).

4.1 Comparison of Deep Learning Architectures

Recently, many studies have adopted deep learning for audio segmentation and classification. State-of-the-art architectures include the Convolutional Neural Network (CNN) (Meléndez-Catalán et al., 2018; Kwon et al., 2020; Lee et al., 2021), Bidirectional Long Short-Term Memory (B-LSTM) (Gimeno et al., 2020), Bidirectional Gated Recurrent Unit (B-GRU) (Phan et al., 2017), Convolutional Recurrent Neural Network (CRNN) (Choi et al., 2017b; Cakır et al., 2017; Zhang et al., 2020), and Temporal Convolutional Network (TCN) (Lemaire & Holzapfel, 2019). CRNN and TCN generally outperform the other architectures because they take advantage of both convolutional and recurrent layers (Lemaire & Holzapfel, 2019). In this section, we compare state-of-the-art architectures to choose the best one for our task. The results presented in this section were published in *Electronics*, as a journal paper in a special issue on Machine Learning Applied to Music/Audio Signal Processing (Venkatesh et al., 2021b).

4.1.1 Experimental Set-up

The training set used for hyperparameter tuning was a combination of real-world data and artificially synthesised data, which was approximately 10 hours and 23 hours respectively. The input layer for each neural network was $802 \times 80 \times 1$, corresponding to 802 times steps and 80 Mel bins. The output of each network had 802×2 neurons with sigmoid activations. Two neurons performed a binary classification for speech and music separately at each time step. Please note that

the model performed multi-output detection, where the occurrences of music and speech are mutually exclusive. All neural networks were trained using the Adam optimiser (Kingma & Ba, 2015) with a learning rate of 0.001. The loss function was binary cross-entropy.

Techniques like Batch Normalisation (BN) (Ioffe & Szegedy, 2015) and Layer Normalisation (LN) (Ba et al., 2016) are adopted to speed up the training process in neural networks. These methods also provide the benefit of producing a regularization effect. Ba et al. (2016) showed that LN was more effective than BN for recurrent architectures like LSTM and GRU. However, BN is more effective for CNNs.

Convolutional Neural Network

For the CNN, we passed the input through a set of 2D convolution layers. After each convolution layer, we performed batch normalization, ReLU activation, and Max pooling. Max pooling was performed only on the frequency axis in order to maintain the time resolution of the network. After the convolution blocks, we had a set of fully connected (FC) layers. Each FC layer was followed by dropout.

Bidirectional Long Short-Term Memory and Bidirectional Gated Recurrent Unit

We passed the input through a series of B-LSTM or B-GRU layers. All layers used tanh activation functions. Initial tests showed that LN was considerably more effective than BN for B-GRU and B-LSTM networks. Also, we observed that a combination of dropout and LN adversely impacted the performance of the network. Hence, each recurrent layer was followed by only LN and no dropout.

Non-causal Temporal Convolutional Network

TCN is a family of architectures that perform 1D convolutions on sequential data. They are known to perform better than B-LSTM and B-GRU architectures on a vast range of tasks (Bai et al., 2018). Lemaire & Holzapfel (2019) proposed the non-causal Temporal Convolutional Network (ncTCN) for audio segmentation, which passes the sequence forward and backward, similar to B-LSTM. The study also showed that ncTCN was more effective than TCN for segmentation.

For our study, ncTCN was implemented by using this GitHub repository[1]. The input was passed through a series of ncTCN layers. We did not observe a difference in performance between LN and BN. Hence, LN was applied after each layer to maintain uniformity with B-LSTM and B-GRU.

Convolutional Recurrent Neural Network

CRNN, as the name suggests, comprises a series of convolution layers followed by recurrent layers. Similar to TCN, we did not observe a noticeable difference between LN and BN and thus, used LN for the network. The input was passed through a set of 2D convolutions. Each convolution was followed by ReLU activation, Max pooling and LN. After the convolution blocks, we had a series of B-GRU layers, each of them followed by LN.

Hyperparameter Tuning

For each network, hyperparameter tuning needs to be performed separately. To present an unbiased comparison, we adopted an automatic hyperparameter method called Hyperband (Li et al., 2017). The list of hyperparameters explored for each architecture can be found in table 4.1. For tuning, we set the maximum number of epochs to 10 and the Hyperband factor to 3.

[1] https://github.com/philipperemy/keras-tcn

Table 4.1: The list of hyperparameters explored for each neural network architecture.

Architecture	Parameter	Range	Step size
CNN	No. of Conv. layers	1 to 4	1
	Kernel size	3 to 15	2
	No. of filters	{16, 32, 64, 128}	-
	No. of FC layers	1 to 4	1
	No. of FC units	128 to 1024	128
	Dropout	0.0 to 0.5	0.05
B-LSTM & B-GRU	No. of layers	1 to 4	1
	No. of hidden units	20 to 260	20
TCN	No. of layers	1 to 4	1
	Kernel size	3 to 19	2
	No. of filters	{16, 32}	-
	No. of stacks	1 to 10	1
	No. of dilations	$\{2^0, 2^1, ..., 2^N\}$ N = 1 to 8	1
	Use skip connections	{True, False}	-
CRNN	No. of Conv. layers	1 to 4	1
	Kernel size	3 to 15	2
	No. of filters	{16, 32, 64, 128}	-
	No. of GRU layers	1 to 4	1
	No. of GRU Units	20 to 160	20

During hyperparameter tuning, we set the batch size to 32 to fit large models in the RAM. For each of the architectures, after an optimal network was chosen by Hyperband, we trained the networks for an extended number of epochs. Early-stopping (Yao et al., 2007) with a patience of 20 epochs was implemented during training and the model obtaining the lowest validation error was selected. Also, we decayed the learning rate by a factor of 0.84 after 10000 weight updates.

4.1.2 Results and Discussion

The parameters chosen by Hyperband can be found in table 4.2. Table 4.3 presents the results of different architectures on the validation and test set. For the overall F-measure, the CRNN network evidently outperformed other models. It also obtained the highest speech F-measure in both datasets. CNN obtained a high music F-measure on the validation set and interestingly the highest F-measure on

the test set. However, its speech F-measure was poorer. Studies have explained that music tends to have more stationary parts than speech (Seyerlehner et al., 2007). Moreover, Jang et al. (2019) adopted a CNN for music detection in radio broadcasts. Our results show that CNNs are excellent for music detection, but not for speech detection.

Table 4.2: The list of hyperparameters chosen by Hyperband each neural network architecture.

Architecture	**Parameter**	**Chosen Value**
CNN	No. of Conv. layers	3
	Kernel size	{9, 11, 11}
	No. of filters	{32, 128, 32}
	No. of FC layers	4
	No. of FC units	{256, 384, 896, 384}
	Dropout	0.45
B-LSTM	No. of layers	3
	No. of hidden units	{140, 160, 80}
B-GRU	No. of layers	3
	No. of hidden units	{60, 140, 100}
TCN	No. of layers	3
	Kernel size	{7, 13, 17}
	No. of filters	{32, 16, 32}
	No. of stacks	{9, 5, 2}
	No. of dilations	N = {3, 7, 2}
	Use skip connections	{False, True, True}
CRNN	No. of Conv. layers	3
	Kernel size	{3, 11, 11}
	No. of filters	{128, 128, 16}
	No. of GRU layers	2
	No. of GRU Units	{80, 40}

There was not much difference between the B-LSTM and ncTCN architectures. Bai et al. (2018) stated that TCNs perform better than recurrent architectures because they exhibit longer effective memory. However, the examples in our training set have a fixed duration of 8s. Probably, the longer memory of TCN was not advantageous for our task.

Unlike TCN, the CRNN model also performs convolutions on the frequency axis, which contributes to its better performance. During initial experiments, we

Table 4.3: The evaluation of different neural network architectures on the validation and test set. CRNN-small is the simplified version of the CRNN model, which has less number of filters for each convolutional layer.

	Validation			Test		
Architecture	$\mathbf{F_{overall}}$	$\mathbf{F_s}$	$\mathbf{F_m}$	$\mathbf{F_{overall}}$	$\mathbf{F_s}$	$\mathbf{F_m}$
CNN	95.87	94.75	96.78	95.23	89.62	**97.72**
B-LSTM	96.55	97.02	96.19	95.85	92.41	97.3
B-GRU	96.24	96.86	95.75	96.11	93.16	97.37
ncTCN	96.56	97.18	96.09	95.9	91.99	97.55
CRNN	**97.39**	**97.75**	**97.12**	96.37	**94**	97.37
CRNN-small	97.21	97.39	97.07	**96.46**	93.64	97.65

also observed that 2D convolutions were more effective than 1D convolutions.

The CRNN model that was selected by Hyperband had 2.4 million parameters. For further experiments in this chapter, the chosen model was unnecessarily large and was beyond our computational budget. We reduced the number of filters from 128 to 64 in the first two convolution layers. This reduced number of parameters to approximately 700 thousand and had a negligible impact on performance. Moreover, we increased the batch size from 32 to 128 to speed up training. Also, we updated the learning rate schedule to decay by a factor of 0.84 after 2500 weight updates. The smaller model is labelled as 'CRNN-small' and the results are presented in table 4.3. Furthermore, the smaller model obtained the highest overall F-measure on the test set. However, our decision was informed only by the validation set. We adopted this 'CRNN-small' as the network for experiments in sections 3.4, 3.5, and 3.6 within chapter 3.

4.2 Experiments with Raw Audio

After comparing state-of-the-architectures in the literature, I was interested in developing novel architectures for audio segmentation. Recently, end-to-end deep learning is gaining a lot of attention in the audio community. This includes training directly on raw audio without features such as MFCC or mel spectrograms. Studies

have explored training directly on raw audio for speech recognition (Hoshen et al., 2015), audio classification (Lee et al., 2017), and source separation (Stoller et al., 2018). This seemed like a promising research avenue to investigate because the literature has not yet explored raw audio for audio segmentation.

The CLDNN architecture was adopted by Sainath et al. (2015b) for speech recognition. In this neural network, the raw audio waveform is processed by temporal or 1-D convolutions. Subsequently, it is fed into LSTM layers. The study showed that the machine learning model using raw audio matched the performance of the previous models using mel spectrograms. The Convolutional Long Short-Term Memory Fully Connected Deep Neural Network (CLDNN) architecture for speech recognition outputs words. However, for audio segmentation, we need to perform segmentation-by-classification, which is classifying each audio sample in the time domain as music or speech.

Audio was downsampled to 8 kHz to manage computational cost. We did not expect the downsampling to lead to any significant data loss because similar sampling rates have been adopted in the literature. For instance, 8 kHz has been used by Meléndez-Catalán et al. (2017) for audio segmentation and Stoller et al. (2018) for source separation. Therefore, if we are adopting audio examples of 8 s, the input size would be 64000×1.

For the initial informal experiments, I developed a CRNN model that accepts raw audio. When tuning the network for hyperparameters, I experimented with filter maps of 32, 64, and 128 and kernel sizes of 7 to 15. After spending a few days exploring various hyperparameters, I was only able to obtain an overall F-measure of 85% on the validation set. This suggested that I need to explore a different network architecture. The CRNN architecture may have worked well for speech recognition because the output layer only produces words. However, in our case, we are performing segmentation-by-classification. Therefore, in the remainder of this section, I investigated the Wave-U-Net architecture for audio segmentation.

4.2.1 Experimental Set-Up

The Wave-U-Net architecture was developed by Stoller et al. (2018) for audio source separation. The structure of the network can be found in Figure 4.1. In this task of source separation, musical audio was separated into stems of *vocals*, *drums*, *bass*, and *other*. The code that accompanied the paper was originally implemented in Tensorflow version 1. Currently, Tensorflow has upgraded to version 2 and there are considerable differences between the two frameworks. Furthermore, the Wave-U-Net has many custom layers such as interpolation, difference output, cropping, and so on. Therefore, I re-implemented the network architecture in Tensorflow 2 and the code is openly available in this GitHub repository[2].

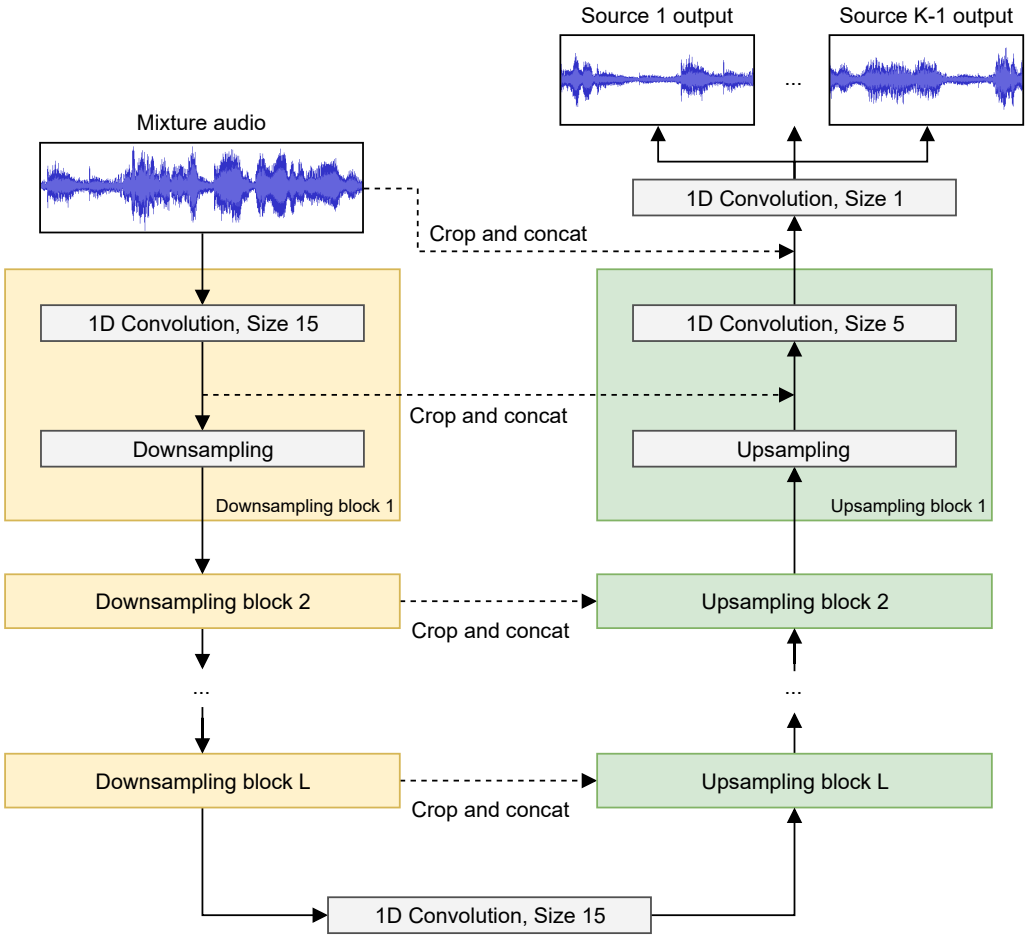


Figure 4.1: The wave-u-net architecture proposed by Stoller et al. (2018). The copyright of the figure belongs to Stoller et al. (2018) and was published under the creative commons attribution license.

The convolutional layers of Wave-U-Net architecture use a kernel size of

[2]https://github.com/satvik-venkatesh/Wave-U-net-TF2

15 (Stoller et al., 2018). As we are working directly with raw audio and not mel spectrograms, higher kernel sizes are expected to work better than smaller kernel sizes. Each convolutional layer is followed by a downsampling block, which reduces the sampling rate of the audio by half. Downsampling is done by parsing the array with a stride of 2. After multiple downsampling blocks, the audio is encoded into a low-dimensional vector space, such as 16. Subsequently, through learned upsampling blocks, the audio features are increased to the original resolution.

The audio was downsampled to 8kHz. Hence, for 8 s audio, the number of samples is 64000. The input size of the Wave-U-Net architecture needs to be a power of 2. Hence, the input size was 65,536. The audio was placed in the centre with zero padding on the left and right. The original output layer of the Wave-U-Net was a differential output layer, which was optimal for audio source separation. We replaced this layer with sigmoidal outputs for music-speech detection. The shape of the output layer was 65536×2, with one neuron for music and one neuron for speech at each audio sample.

For regularising the Wave-U-Net, Stoller et al. (2018) multiplied the audio signal by a random value between 0.7 and 1.0. This data augmentation technique randomly adjusts the gain of the audio, which improves the generalisation of the neural network. We adopted the same approach in our training pipeline.

In order to ensure that the performance of the model is not hindered by the reduced sampling rate, we trained another model on a sampling rate of 22050 kHz. However, due to the higher sampling rate, the duration of each audio example was reduced to 5.94 s to fit the input within the RAM. The new input size was 131,072.

4.2.2 Results

Table 4.4 compares the performance of the Wave-U-Net and the CRNN trained on mel spectrograms. First, we analysed the results on the validation set and observed that the CRNN trained on mel spectrograms significantly surpassed the Wave-U-Net trained on raw audio. The overall precision of the CRNN is 97.35%, compared to 95.92% and 95.15% for Wave-U-Net-8k and Wave-U-Net-22k respectively. The overall recall of the CRNN was 97.43%, compared to 93.71% and 94.1% for Wave-U-Net-8k and Wave-U-Net-22k respectively. Due to the unsatisfactory performance on the validation set, we did not evaluate the model on the test set.

Algo.	$\mathbf{F_{overall}}$	$\mathbf{F_m}$	$\mathbf{P_m}$	$\mathbf{R_m}$	$\mathbf{F_s}$	$\mathbf{P_s}$	$\mathbf{R_s}$
CRNN	97.39	97.1	97.4	96.8	97.7	97.2	98.3
Wave-U-Net-8k	94.81	93.6	96.3	91.1	96.3	95.5	97.1
Wave-U-Net-22k	94.62	93.9	96.5	91.4	95.5	93.5	97.6

Table 4.4: A comparison of CRNN trained on mel spectrograms and Wave-U-Net trained on raw audio. This results are on the validation set. Wave-U-Net-8k was trained on audio with a sampling rate of 8 kHz and Wave-U-Net-22k was trained audio with sampling rate of 22.05 kHz.

It is understood that training on raw audio requires more training data because there is no extraction of audio features. This makes the network more prone to overfitting. Although, training on raw audio has demonstrated advantages in audio classification (Lee et al., 2017) and speech recognition (Sainath et al., 2015b), it does not seem beneficial for music-speech detection in radio broadcast.

The performance for raw audio can be improved by training the neural network on a larger dataset. However, in this thesis, we have already explored artificially synthesising training sets and annotating real-world radio data provided by BBC Radio Devon. Annotating more audio examples is an expensive and time-consuming task. Therefore, this thesis concludes that the benefits of training on raw audio are limited for music-speech detection. However, techniques like transfer learning can be adopted, where the network is trained on larger datasets

for similar tasks such as speech recognition and audio classification. Subsequently, the network can be fine-tuned for music-speech detection.

4.3 You Only Hear Once (YOHO) Algorithm

There are many studies focusing on end-to-end deep learning. However, in the audio community, end-to-end deep learning is generally referred to the input audio. In other words, features like mel spectrograms are replaced with raw audio waveforms. There has been less attention given to the output of such networks. Traditionally, in segmentation-by-classification, the neural network classifies each audio frame. Subsequently, a post-processing step converts the neural network's output into human-readable labels. The disadvantage is that this post-processing is slow because each audio frame has to be serially processed. Therefore, in this section, I explore the idea of transforming audio segmentation from a classification problem to a regression one. This way, the neural network would output human-readable labels by directly predicting the boundaries of acoustic classes.

Phan et al. (2014) proposed random regression forests for sound event detection and classification. Xu et al. (2014) adopted a regression approach for speech enhancement. However, most studies in the literature adopt frame-based classification, where the neural network classifies each frame separately. In this section, we present a novel neural network architecture inspired by the You Only Look Once (YOLO) algorithm (Redmon et al., 2016). YOLO gained attention in the Computer Vision community for object detection. It transformed bounding box prediction from a classification problem to a regression one. Using this approach, it obtained speedups of around $3\times$ without compromising accuracy. We present a system called You Only Hear Once (YOHO) that predicts the boundaries of acoustic classes through regression.

YOLO has been adopted in the audio domain by visualising spectrograms

as images. Zsebők et al. (2019) adopted YOLO for automatic bird song and syllable segmentation. Segal et al. (2019) presented a system called SpeechYOLO which treated audio fragments as objects. They adopted YOLO for keyword spotting tasks. Algabri et al. (2020) investigated object detection techniques such as YOLO and CenterNet (Zhou et al., 2019) for phoneme recognition. However, the novelty of the YOHO paradigm is that it converts frame-based classification into a regression problem by gradually reducing the temporal dimension through many convolutional layers. This makes the output of the network closer to human-readable labels, therefore reducing the need for post-processing. Separate neurons were used to detect the onset and offset of audio classes. We apply our system to audio segmentation and sound event detection tasks, where the literature has predominantly used frame-based classification. Furthermore, we present a multi-output system, which detects acoustic classes that can overlap with each other.

Until now, in this thesis, we have used datasets only for music-speech detection. As we are proposing a novel algorithm, it would be interesting to evaluate YOHO on multiple audio event detection tasks. First, we explore music-speech detection in broadcast signals. We also compare our results with state-of-the-art algorithms on the Music Information Retrieval Evaluation eXchange (MIREX) competition dataset 2018 (Schlüter et al., 2018). Second, we test our model on the TUT sound event detection dataset, which represents common sounds related to human presence and traffic. It was the dataset used in the Detection and Classification of Acoustic Scenes and Events (DCASE) competition 2017 (Mesaros et al., 2017). Third, we evaluate our model on the Urban-SED dataset (Salamon et al., 2017), which is a synthetic dataset for environmental audio. In all three cases, the YOHO algorithm performed better and faster than the CRNN. All the code associated with this project is available in this GitHub repository (https://github.com/satvik-venkatesh/you-only-hear-once, accessed on 2 March 2022).

4.3.1 Motivation

In the YOHO system, we intend to make the neural network output labels closer to human-readable labels. This way we make the pipeline more end-to-end. Figure 4.2 illustrates a comparison between segmentation-by-classification and the YOHO paradigm. For both paradigms, a mel spectrogram of shape 801×64 is fed as input. In segmentation-by-classification, each time step is classified as music, speech, both, or none. Subsequently, these classifications are converted to human readable labels. However, in YOHO, each block of 0.307 s is processed through regression. One neuron detects the presence of an acoustic class. If the class is present, one neuron predicts the start point of the class and one neuron detects the end point of the class. Subsequently, during post-processing, these blocks of 0.307 s are merged to form a final prediction. Using this technique, the number of time steps is reduced from 801 to 26, which makes the network significantly faster, generalise better, and more end-to-end. More details on the implementation are gi

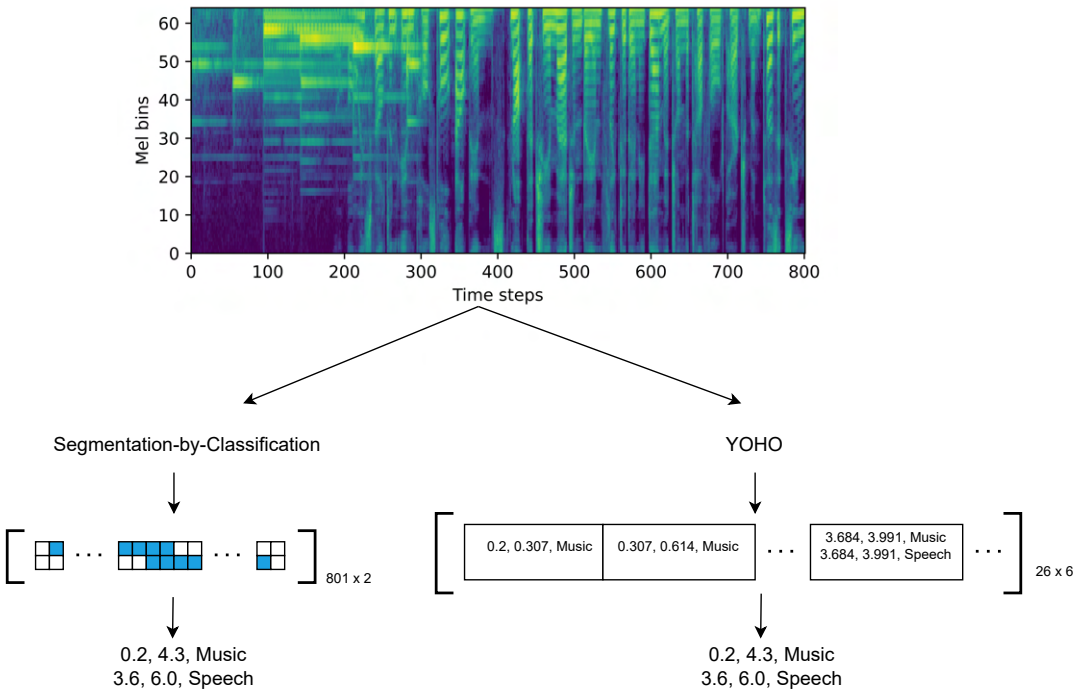


Figure 4.2: A comparison of segmentation-by-classification and YOHO.

4.3.2 Network Architecture

The network architectures used by different versions of YOLO (Redmon et al., 2016; Redmon & Farhadi, 2017) were large and not suitable for our smaller training datasets. Therefore, we adapted the MobileNet architecture (Howard et al., 2017) for our task. We modified the final layers of MobileNet to realize the YOHO algorithm. MobileNet has also been employed for audio classification by YamNet (Plakal & Ellis, 2020), which only detects audio classes, but not their segmentation boundaries.

As shown in Table 4.5, YOHO is purely a convolutional neural network (CNN). We divide the table into two parts — the upper half comprising the original layers of the MobileNet architecture and the bottom half containing the layers that we have added. We use log-mel spectrograms as input features. The input dimension depends on the duration of the audio example and specifications of the mel spectrogram. Here, we explain the network for music-speech detection, whose input contains 801 times steps and 64 frequency bins. After reshaping the mel spectrogram to $801 \times 64 \times 1$, we perform a 2D convolution with a stride of 2. Hence, the time dimension and frequency dimension are reduced by half. The MobileNet architecture uses many depthwise-separable convolutions (Sifre, 2014) with 3×3 filters followed by pointwise convolutions with 1×1 filters. All convolutions except the final layer were fitted with ReLu activations and batch normalization (Ioffe & Szegedy, 2015). Each time we adopt a stride of 2, there is a reduction in the time and frequency dimensions. As shown in the lower half of Table 4.5, we gradually reduce the number of filters from 1024 to 256.

Subsequently, we flatten the last two dimensions. The final layer is a 1D convolution with six filters. The output shape is 26×6, where 26 stands for the number of time steps. This layer is similar to a convolutional implementation of sliding windows (Sermanet et al., 2014) along the time dimension. At each time step, the first neuron performs a binary classification that detects the presence of

Table 4.5: The neural network architecture for YOHO. The upper half of the table comprises the original layers of MobileNet. The bottom half contains the layers that we have added. Conv2D and Conv1D stand for 2D and 1D convolutions, respectively. The convolutions use a stride of 1 unless mentioned otherwise and 'dw' stands for depthwise convolution.

Layer type	Filters	Shape / Stride	Output shape
Reshape	-	-	801 x 64 x 1
Conv2D	32	3 x 3 / 2	401 x 32 x 32
Conv2D-dw	-	3 x 3	401 x 32 x 32
Conv2D	64	1 x 1	401 x 32 x 64
Conv2D-dw	-	3 x 3 / 2	201 x 16 x 64
Conv2D	128	1 x 1	201 x 16 x 128
Conv2D-dw	-	3 x 3	201 x 16 x 128
Conv2D	128	1 x 1	201 x 16 x 128
Conv2D-dw	-	3 x 3 / 2	101 x 8 x 128
Conv2D	256	1 x 1	101 x 8 x 256
Conv2D-dw	-	3 x 3	101 x 8 x 256
Conv2D	256	1 x 1	101 x 8 x 256
Conv2D-dw	-	3 x 3 / 2	51 x 4 x 256
Conv2D	512	1 x 1	51 x 4 x 256
5x Conv2D-dw	-	3 x 3	51 x 4 x 256
5x Conv2D	512	1 x 1	51 x 4 x 256
Conv2D-dw	-	3 x 3 / 2	26 x 2 x 512
Conv2D	1024	1 x 1	26 x 2 x 1024
Conv2D-dw	-	3 x 3	26 x 2 x 1024
Conv2D	1024	1 x 1	26 x 2 x 1024
Conv2D-dw	-	3 x 3	26 x 2 x 1024
Conv2D	512	1 x 1	26 x 2 x 512
Conv2D-dw	-	3 x 3	26 x 2 x 512
Conv2D	256	1 x 1	26 x 2 x 256
Conv2D-dw	-	3 x 3	26 x 2 x 256
Conv2D	128	1 x 1	26 x 2 x 128
Reshape	-	-	26 x 256
Conv1D	6	1	26 x 6

an acoustic class. The second and third neurons perform regression for the start and endpoints for the respective acoustic class. Figure 4.3 illustrates the output layer of the YOHO algorithm.

In this context, we are dealing with two acoustic classes — music and speech. Therefore, the output has six neurons at each time step. For example, if the length of an audio example is 8 s, each time step in the output corresponds to 0.307 s

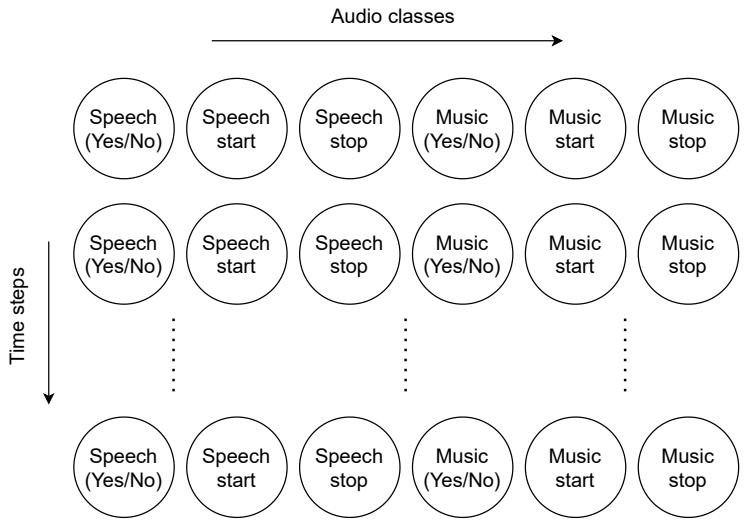


Figure 4.3: An illustration of the output layer of the YOHO algorithm. This network is for music-speech detection. To increase the number of audio classes, we add neurons along the horizontal axis.

because there are 26 divisions. We applied sigmoid activations for all neurons in the output layer. Hence, we normalized the regression outputs between 0 and 1. Moreover, even if the input shape of the neural network is different, for example, 257×40, the neural network and the parameters of convolutional layers still remain exactly the same. The only difference would be the output shape of the neural network, which depends on the number of time steps in the input and the number of unique audio classes in the output.

4.3.3 Loss Function

Generally, neural networks such as the CRNN that use segmentation-by-classification adopt binary cross-entropy as the loss function. As we modelled the problem as a regression one, we used the sum squared error. Equation 4.1 shows the loss function for each acoustic class c.

$$\mathcal{L}_c(\hat{y}, y) = \begin{cases} (\hat{y_1} - y_1)^2 + \\ (\hat{y_2} - y_2)^2 + (\hat{y_3} - y_3)^2, & \text{if } y_1 = 1 \\ (\hat{y_1} - y_1)^2, & \text{if } y_1 = 0 \end{cases} \tag{4.1}$$

where y and $\hat{y}$ are the ground-truth and predictions respectively. $y_1 = 1$ if the acoustic class is present and $y_1 = 0$ if the class is absent. y_2 and y_3, which are the start and endpoints for each acoustic class are considered only if $y_1 = 1$. In other words, $(\hat{y_1} - y_1)^2$ corresponds to the classification loss and $(\hat{y_2} - y_2)^2 + (\hat{y_3} - y_3)^2$ corresponds to the regression loss. The total loss $\mathcal{L}$ is summed across all acoustic classes.

4.3.4 Example of Labels

Table 4.6 shows an example of the output for the YOHO algorithm. The total length of the audio is 8 s. Within the example, Music occurs from 0.2 to 4.3 s and Speech occurs from 3.6 to 6.0 s. Note that each row in Table 4.6 corresponds to one time step, which is equal to 0.307 s. In addition, the regression values are normalized from 0 to 1. For example, if music starts at 0.2 s, the value is divided by 0.307 to get 0.65 as shown in the first row of Table 4.6.

4.3.5 Other Details

We trained the network with the Adam optimiser, a learning rate of 0.001, a batch size of 32, and early stopping (Yao et al., 2007). In some cases, we used L2 normalization, spatial dropout, and SpecAugment (Park et al., 2019). We used log-mel spectrograms as features for the neural network. The parameters of spectrograms were unique for each dataset. Section 4.5 contains the details for each case.

To evaluate the systems, we adopted the sed_eval toolbox (Mesaros et al., 2016), which is common in the literature for audio segmentation and sound event detection. The python toolbox is openly available (`https://tut-arg.github.io/sed_eval/`, accessed on 17 March 2022) and presents a convenient interface to calculate metrics such as overall F-measure, error rate, class-based F-measures, and

Table 4.6: An example of labels for the YOHO algorithm. Music occurs from 0.2 to 4.3 s and Speech occurs from 3.6 to 6.0 s. Note that start and stop values are considered only when the respective audio class is present. The dimensions of the output are 26 $\times$ 6. Note that each time step/row in the table corresponds to 0.307 s. The start and stop values are normalised on the range of 0 to 1. For instance, in the first time step, music's start point would be rescaled from 0.2 to 0.65.

Speech (Yes / No)	**Speech start**	**Speech stop**	**Music (Yes / No)**	**Music start**	**Music stop**
0	-	-	1	0.65	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
1	0.7	1.0	1	0.0	1.0
1	0.0	1.0	1	0.0	1.0
1	0.0	1.0	1	0.0	0.975
1	0.0	1.0	0	-	-
1	0.0	1.0	0	-	-
1	0.0	1.0	0	-	-
1	0.0	1.0	0	-	-
1	0.0	1.0	0	-	-
1	0.0	0.5	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-

so on. The specifications of segment-based metrics for experiments are mentioned along with the relevant results in Section 4.6.

4.3.6 Post-Processing

For music-speech detection, the output of the CRNN would be 801 $\times$ 2, corresponding to 801 times steps and two acoustic classes. On the other hand, the output for the YOHO network is 26 $\times$ 6. A post-processing step parses the output

of the neural network to create human-readable labels. Subsequently, smoothing is performed over the output to eliminate the occurrence of spurious audio events. Two smoothing approaches are common in the literature — median filtering (Gimeno et al., 2020) and threshold-dependent smoothing (Lemaire & Holzapfel, 2019). We adopted the latter approach. In this technique, if the duration of the audio event is too short or if the silence between consecutive events of the same acoustic class is too short, we remove the occurrence.

For music-speech detection, the minimum silence between consecutive music events or consecutive speech events was set to 0.8 s. The minimum duration for a music event was set to 3.4 s and for a speech event was 0.8 s. For environmental sound event detection, if the silence between consecutive audio events of the same acoustic class was less than 1.0 s, it was smoothed. We did not set any threshold for the minimum duration of an audio event for this task.

4.4 Models for Comparison

In this sub-section, we present two additional models, which are slight deviations from the YOHO architecture — CNN and CRNN. The motivation behind these models is to investigate which aspects of YOHO are actually advantageous. The feature-extraction for all these models is the same, which makes them directly comparable. The CNN model aims to create a segmentation-by-classification version of the YOHO architecture. As you can see in Table 4.5, some Conv2D-dw layers adopt a stride of [2, 2]. These strides were set to [1, 1] instead of [2, 2] and max-pooling was adopted to reduce the frequency dimension by half. This way, the time resolution of the network does not reduce through its depth. Note that using a stride of [1, 2] would have produced a similar effect of maintaining the time resolution and reducing the frequency resolution. However, TensorFlow currently does not support rectangular strides for depthwise convolutions and hence, we adopted max-pooling. The number of parameters in the CNN was 3.9

million, which is the same as the network for YOHO.

In the CRNN model, the first 13 layers were identical to the YOHO network. We skipped the convolutional layers where the number of filters became larger than 256 because the network became too large to fit into the RAM. Following the convolutional layers, we had two B-GRU layers with 80 units each. The number of parameters for the CRNN was 1.3 million, which is less than the YOHO network. Increasing the number of convolutional layers only worsened the performance of the CRNN. Therefore, it was optimal to have a CRNN with fewer parameters.

The output shape for the CNN and CRNN was 801×2, performing binary classification for music and speech at each time step. We compared the performance of YOHO with these two additional models on the in-house test set for music-speech detection. We also compared the inference times of these models. A summary of the architectures for comparison can be found in Table 4.7.

Table 4.7: Models for comparison on the in-house test set for music-speech detection.

Model	**Remarks**
YOHO	The architecture is explained in Section 4.3.2.
CNN	[2, 2] strides in convolutions are replaced by [1, 1] strides, followed by max-pooling of [1, 2] to maintain the time resolution.
CRNN	Only Conv2D and Conv2D-dw layers until 256 filters are included from Table 4.5. After this, two B-GRU layers with 80 units each are added.

4.5 Datasets

We evaluate the robustness of the YOHO algorithm on multiple datasets. This section explains the different datasets and how we adapt the YOHO algorithm for each of them.

4.5.1 Music-Speech Detection

Music-speech detection aims to detect the boundaries of music and speech in audio signals such as radio and TV programs. The neural network performs multi-output detection to allow the simultaneous occurrence of music and speech. The number of output neurons at each time step is six because we are detecting two acoustic classes. The dataset comprised 18 h of audio from BBC Radio Devon and 5 h from the MuSpeak dataset (MuSpeak Team, 2015). As done in chapter 3, both datasets were roughly split into 50% for training, 30% for validation, and 20% for testing. We artificially synthesised training data using the method explained in section 3.1. We included 46 h of synthetic examples in the training set. Table 4.8 shows a brief overview of the contents of each split in the dataset.

Table 4.8: Contents of train, validation, and test datasets for music-speech detection. Real-world radio data was collected from BBC Radio Devon. MuSpeak (MuSpeak Team, 2015) is an openly available dataset containing annotations for music and speech. 46 h of artificial radio-like examples were synthesised by the method presented in section 3.2.

Dataset Division	Contents
Train	46 h of synthetic radio data + 9 h from BBC Radio Devon + 1 h 30 min from MuSpeak
Validation	5 h from BBC Radio Devon + 2 h from MuSpeak
Test	4 h from BBC Radio Devon + 1 h 42 min from MuSpeak

All the audio files were resampled to 16 kHz. They were converted to mono by averaging the channels before pre-processing. Subsequently, we extracted 64 log-mel bins with a hop size of 10 ms and a window size of 25 ms. The frequencies for the mel spectrogram ranged from 125 Hz to 7.5 kHz. Note that the audio features used in the section are different from earlier experiments in this thesis. This was done to adopt audio features similar to those used by YamNet (Plakal & Ellis, 2020), which uses MobileNet as the basic framework. Note that we did not use any regularization such as L2 normalization, spatial dropout, or SpecAugment for this dataset because the training set is large.

We evaluate the model on two different test sets. The first one being our in-house test set, which contains approximately 4.5 h of audio from BBC Radio Devon and MuSpeak (MuSpeak Team, 2015). The second one was the MIREX music-speech detection dataset, which contains 27 h of audio from various TV programs.

4.5.2 TUT Sound Event Detection

The TUT Sound Event Detection dataset focuses on environmental sound detection (Mesaros et al., 2017). It was adopted for the third task of the DCASE challenge 2017. It primarily consists of street recordings with traffic and other activity. Each audio example is 2.56 s. There were six unique audio classes — Brakes Squeaking, Car, Children, Large Vehicle, People Speaking, and People Walking. Thus, to predict the existence of the six classes, plus start and end times, we required 18 output neurons. The more recent DCASE challenges use additional techniques such as semi-supervised learning and source separation, which is not the focus of this study. Hence, we used the dataset from 2017 that contains only strongly labelled data.

The total size of the dataset is approximately 1.5 h. The dataset comes with a four-fold cross-validation setup. The size of this dataset is significantly smaller than the one used for music-speech detection and may not be large enough for our deep learning architecture. Therefore, we applied L2 normalization of 0.001 on the first Conv2D layer. In addition, we included L2 normalization of 0.01 and spatial dropout of 0.1 on all the subsequent Conv2D layers. For data augmentation, we incorporated SpecAugment (Park et al., 2019), which randomly drops a sequence of frequency bins or time steps from the input. Note that there were slight differences in our implementation of SpecAugment. We did not use any time warping because it becomes complicated to redefine labels for audio events. In addition, we applied SpecAugment on batches instead of individual examples to save computational

time.

The database contained stereo audio files with a sampling rate of 44.1 kHz. These were downmixed to mono before pre-processing. Subsequently, we extracted 40 log-mel bands in the range of 0 to 22,050 Hz. The hop size was 10 ms and the window size was 40 ms. We adopted audio features similar to the baseline system (Mesaros et al., 2017) for the task, except that we used a smaller hop size. As the input of the network contains 2.56 s of audio, the input shape is 257×40 corresponding to 257 times steps and 40 mel bins. The output shape of the network is 9×18, corresponding to 9 times steps and 6 acoustic classes. Note that each time step, in this case, is 0.284 s, which is different from 0.307 s for music-speech detection. In both cases, we used the same network and the same sequence of convolutional layers. The convolutions with a stride of 2 reduce the temporal dimension by half. Hence, due to different input sizes, the number of time steps is 9 in one case and 26 in the other case. There were no special measures taken to estimate the duration of each time step beforehand. However, in most cases, it was somewhere around 0.3 s, due to the hop and window sizes selected for feature extraction.

4.5.3 Urban-SED

The Urban Sound Event Detection dataset is a purely synthetic dataset generated by using scaper (Salamon et al., 2017). Each audio example was 10 s. There were ten unique audio classes — Air Conditioner, Car Horn, Children Playing, Dog Bark, Drilling, Engine Idling, Gun Shot, Jackhammer, Siren, and Street Music. The total size of the dataset is about 30 h and contains pre-defined splits for training, validation, and testing. As there were ten audio classes, the number of output neurons in YOHO was 30.

We used the same audio features as explained in Section 4.5.2. For this dataset, we did not use any SpecAugment because the training set was larger. The L2

normalization and spatial dropout were identical to those used in Section 4.5.2. As the input of the network contains 10 s of audio, the input shape is 1001 $\times$ 40 corresponding to 1001 times steps and 40 mel bins. The output shape of the network is 32 $\times$ 30, corresponding to 32 times steps and ten acoustic classes.

4.6 Results

4.6.1 Music-Speech Detection

In-House Test Set

Table 4.9 shows the results on our in-house test set. F-measure was calculated using the sed_eval (Mesaros et al., 2016) module with a segment size of 10 ms. We compare the results of YOHO with the CNN and CRNN models explained in Section 4.4. In addition, we compare the performance with CNN and CRNN architectures published in previous research (Venkatesh et al., 2021a,b). All the deep learning models were trained using the same training set. YOHO obtains the highest F-measure for overall, music, and speech. YOHO significantly outperforms the CNN, which is the segmentation-by-classification version of the model. It is important to note that both models follow the same process for feature extraction and have the same number of parameters. This shows that our regression approach of predicting the acoustic boundaries directly is effective. The other CNN (Venkatesh et al., 2021b) used larger kernel sizes such as 9 and 11, which may have improved the F-measure of Speech.

YOHO also outperforms the three CRNN architectures. CRNN (Venkatesh et al., 2021a) used a kernel size of 7 and CRNN (Venkatesh et al., 2021b) used kernel sizes of 3, 11, and 11. In addition, CRNN (Venkatesh et al., 2021b) used layer normalisation (Ba et al., 2016) instead of batch normalisation (Ioffe & Szegedy, 2015). Therefore, we show that YOHO outperforms a variety of CRNN architectures in

the literature.

Table 4.9: Results on our in-house test set for music-speech detection. The F-measures for overall, music, and speech are presented as percentages. The values in bold indicate the largest number in each column.

Algorithm	$F_{overall}$	F_{music}	F_{speech}
YOHO	**97.22**	**98.20**	**94.89**
CRNN	96.79	97.84	94.26
CNN	93.89	97.96	85.13
CRNN (Venkatesh et al., 2021b)	96.37	97.37	94.00
CRNN (Venkatesh et al., 2021a)	96.24	97.30	93.80
CNN (Venkatesh et al., 2021b)	95.23	97.72	89.62

MIREX Music-Speech Detection

Table 4.10 shows the results on the MIREX music-speech detection dataset. YOHO obtains the highest overall F-measure, which makes it state-of-the-art for music-speech detection. The music F-measure for a CRNN (Venkatesh et al., 2021a) slightly surpassed the YOHO algorithm by 0.1%. However, YOHO obtained the highest F-measure for speech.

Table 4.10: Evaluation on the MIREX music-speech detection dataset 2018. The results of other studies were obtained from the MIREX website (Schlüter et al., 2018). The F-measures are presented as percentages. The values in bold indicate the largest number in each column.

Algorithm	$F_{overall}$	F_{music}	F_{speech}
YOHO	**90.20**	85.66	**93.18**
CRNN (Venkatesh et al., 2021a)	89.53	**85.76**	92.21
CRNN (Venkatesh et al., 2021b)	89.09	85.01	92.16
CNN (Marolt, 2018)	-	54.78	90.9
Logistic Regression (Marolt, 2018)	-	38.99	91.15
ResNet (Marolt, 2018)	-	31.24	90.86
MLP (Choi et al., 2018)	-	49.36	77.18

4.6.2 TUT Sound Event Detection

Table 4.11 shows the results on the TUT sound event detection dataset. It also contains the results of the top three performers in the competition. For this competition, they adopted error rate (Mesaros et al., 2016) as the main metric. Note

that a lower error rate indicates better performance of the algorithm. Furthermore, a segment size of 1 s was adopted to calculate segment-based metrics. The first place in the competition was the CRNN architecture (Adavanne & Virtanen, 2017). They used 3×3 kernels followed by B-GRU layers with 32 units. Their model was optimised by a random hyper-parameter search (Bergstra & Bengio, 2012) for the number of layers and units. The second place in the competition adopted a multi-input CNN with 3×3 kernels and a bespoke feature extraction process. The third place adopted a B-GRU model. Note that all three models adopt segmentation-by-classification. YOHO obtained a better error rate than the CRNN (Adavanne & Virtanen, 2017), CNN (Jeong et al., 2017), and B-GRU (Lu & Duan, 2017) models. To ensure that the improvement in performance was not attributed to data augmentation, we re-trained the best CRNN network (Adavanne & Virtanen, 2017) with SpecAugment. However, it worsened the performance of the algorithm. This may be because the CRNN uses segmentation-by-classification. Therefore, masking series of time steps leads to noise in the labels. However, the YOHO algorithm is relatively robust to this issue as it directly predicts boundaries through regression.

Our results are not state-of-the-art on this dataset. Vesperini et al. (2019) adopted a Capsule Neural Network (CapsNet) and binaural short-time Fourier transform (STFT) for feature extraction and obtained an error rate of 0.58. Luo et al. (2021) presented a Capsule Neural Network Recurrent Neural Network (CapsNet-RNN) that obtained an error rate of 0.57. However, these optimisations were beyond the scope of this study. It is important to note that YOHO is a paradigm and not an architecture. We show that regression outperforms segmentation-by-classification for multiple models. Future research can explore how YOHO can be optimised by adopting a CapsNet-style architecture.

Table 4.11: Results on the TUT sound event detection dataset. The value in bold indicates the algorithm with the lowest error rate.

Algorithm	Error Rate
CapsNet-RNN (Luo et al., 2021)	**0.57**
CapsNet (Vesperini et al., 2019)	0.59
YOHO	0.75
CRNN (Adavanne & Virtanen, 2017)	0.79
CNN (Jeong et al., 2017)	0.81
B-GRU (Lu & Duan, 2017)	0.83

4.6.3 Urban-SED

Table 4.12 shows the results on the Urban-SED dataset for overall F-measure. A comparison of class-wise performance is also presented in Figure 4.4. The YOHO algorithm is compared with the CRNN and CNN model presented by Salamon et al. (2017). YOHO obtains the highest overall F-measure. Among class-wise F-measures, YOHO obtains the highest for Children Playing, Dog Bark, Drilling, Gun Shot, Siren, and Street Music. CRNN obtains the highest for Air Conditioner and Engine Idling. CNN obtains the highest for Car Horn and Jackhammer.

Table 4.12: Segment-based overall F-measure on the Urban-SED dataset. The value in bold indicates the algorithm with the highest F-measure.

Algorithm	$F_{overall}$
CRNN with envelope estimation (Martín-Morató et al., 2019)	**64.70**
YOHO	59.50
CNN (Salamon et al., 2017)	56.88
CRNN (Salamon et al., 2017)	55.96

As you can see in Table 4.12, Martín-Morató et al. (2019) adopted sound event envelope estimation on a CRNN model to improve the overall F-measure to 64.7%, compared to 59.5% obtained by YOHO. In future research, YOHO's performance can be improved by incorporating techniques like envelope estimation. In addition, weakly supervised sound event detection with envelope estimation has further improved the performance of the CRNN on this dataset (Dinkel et al., 2021).

4.6.4 Speed of Prediction

In this section, we compare the inference times of YOHO, CNN and CRNN models for music-speech detection. This experiment was performed on the in-house test set explained in Section 4.5.1. To calculate the inference time, the prediction was made over the entire test set. Later, the inference time was divided by the number of hours of audio to obtain the average time taken per hour of audio. As we are adopting Google Colab for experiments, we ensured that all models were tested within the same runtime session. This way, we ensure that the same computing resources were given to YOHO, CNN, and CRNN. While training the models in earlier runtime sessions, we stored their weights on Google Drive. When running the experiment to calculate inference times, these weights were loaded from Google Drive. Note that separate runtime sessions were used to calculate inference times over CPU and GPU as shown in Figure 4.5, however, the same session was used for inter-model comparison. Some important aspects of the system configuration were — Intel Xeon CPU processor, 12 GB RAM, and Tesla P100 GPU (only when GPU was used).

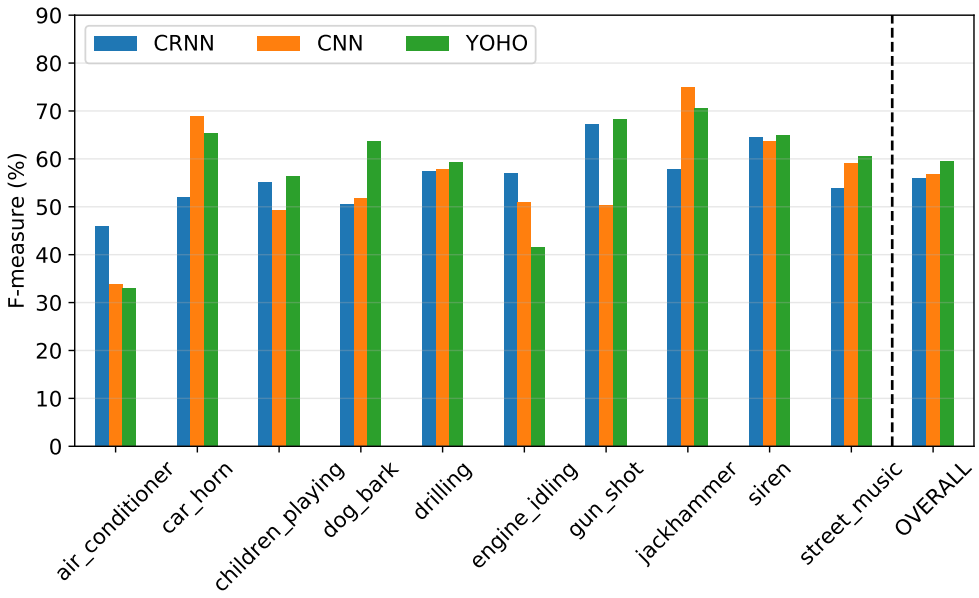


Figure 4.4: Segment-based F-measures for each class on the Urban-SED dataset calculated using segment-size of 1 s.

Figure 4.5 compares the inference times of YOHO, CNN and CRNN models

for music-speech detection on the in-house test set. The CNN and CRNN models were explained in Section 4.4. YOHO and the CNN had exactly the same number of parameters, which is 3.9 million. The only difference is that the CNN adopts frame-based classification instead of regression. The CRNN model had 1.3 million parameters, which was less than the CNN and YOHO. On the CPU, the prediction time of YOHO was 14 times faster than the CNN and 5 times faster than the CRNN. On the Graphical Processing Unit (GPU), the prediction time of YOHO was 6 times faster than the CNN and 4 times faster than the CRNN. The increase in prediction speed is because YOHO has to predict only 26×6 neurons, whereas the CNN and CRNN have to predict 801×2 neurons. Despite the CRNN having fewer parameters, YOHO is significantly faster.

As YOHO outputs acoustic boundaries directly, the post-processing and smoothing for YOHO was 7 times faster than the CNN and CRNN. Note that the smoothing is performed only on the CPU.

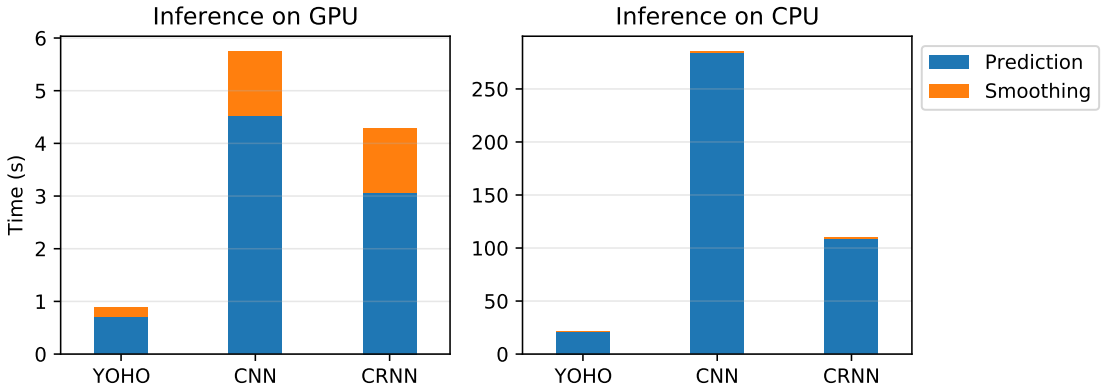


Figure 4.5: Average time taken to make predictions on 1 h of audio for music-speech detection. 'Prediction' refers to the time taken by the network to make predictions. 'Smoothing' is the post-processing step to parse the output of the network. The GPU used for inference was the Tesla P100.

4.7 Discussion

The results in Section 4.6 show that YOHO has multiple advantages over the state-of-the-art CRNN architecture. We examined the model for two different tasks —

music-speech detection and environmental sound event detection. Music-speech detection is relatively a simpler task because of a larger and diverse training set. Additionally, there are only two acoustic classes to predict. On the other hand, environmental sound event detection was harder because of smaller and lower quality training sets. In addition, the number of acoustic classes was greater. However, in both scenarios, YOHO generalised better than CRNN and CNN. YOHO obtained state-of-the-art performance for music-speech detection on the MIREX 2018 competition dataset. We understand that YOHO has not obtained state-of-the-art performance on TUT Sound Events and Urban-SED datasets. However, it is important to note that the purpose of this study is to shift the paradigm from frame-based classification to regression for audio segmentation and sound event detection. There is a vast body of research involving CNN and CRNN architectures. It is not within the capacity of this study to incorporate all these optimisations for YOHO. As this is the first study that explores this paradigm, we believe that optimisations such as weak label learning (Miyazaki et al., 2020b) and envelope estimation (Martín-Morató et al., 2019) will improve YOHO's performance.

We also explored the idea of creating a regression-based CRNN that adopts the YOHO paradigm. We replaced the Conv1D layer with a B-GRU block. However, this slightly worsened the performance of the algorithm. This is because the YOHO network has many convolutional layers that reduce the temporal resolution from 801 to 26. Hence, the B-GRU blocks may not be effective on such a small number of time steps. However, alternative structures such as CNN-transformers (Kong et al., 2020b) may be a promising avenue to explore.

YOHO was significantly quicker than the CNN and CRNN models because it had to predict fewer outputs and computationally cheaper post-processing. As explained earlier, the output produced by YOHO is more end-to-end. For example, the output dimensions in music-speech detection are 26×6 for YOHO versus 801×2 for the CRNN. This corresponds to 156 output neurons for YOHO and 1602 for the CRNN. Furthermore, the CRNN needs to convert frame-based classifications

to time boundaries. However, YOHO directly outputs the time boundaries. Due to the above reasons, YOHO is significantly quicker. Due to faster inference, YOHO is more suitable for real-time applications such as surveillance, self-driving automobiles, bioacoustic monitoring, and real-time remixing.

4.8 Publications, Code, and Contributions

- Venkatesh, S., Moffat, D., & Miranda, E. R. (2022b). You only hear once: a YOLO-like algorithm for audio segmentation and sound event detection. *Applied Sciences*, *12*(7), 3293. doi: 10.3390/app12073293.

  – In this journal paper, we presented the You Only Hear Once (YOHO) algorithm. The code associated with this study is open available at this GitHub repository (https://github.com/satvik-venkatesh/you-only-hear-once/). Shortly after we put out a pre-print on ArXiv, Tiwari et al. (2021) evaluated the YOHO algorithm on the VOICe dataset (Gharib et al., 2019), which contains audio files with noise at different sound-to-noise ratios. The study found that YOHO outperformed or at least matched the best performing CRNN architecture in multiple scenarios. As YOHO shifts the paradigm from a classification problem to a regression problem, it would be interesting to see how it performs in other audio tasks.

Chapter 5

Towards Domain Generalisation

Over the years, supervised learning algorithms have achieved remarkable success in multiple disciplines. However, it is often assumed that the training and tests come from the same data distributions. A domain distribution gap (Quinonero-Candela et al., 2008) between the training and test set makes it more challenging for the model to do well on *unseen* domains. In this chapter, we investigate domain generalisation to improve the robustness of audio segmentation algorithms. During my time as a PhD student, I pursued a five-month research internship at Mitsubishi Electric Research Laboratories (MERL). At MERL, we explored domain generalisation for anomalous sound detection. We developed some methods for domain generalisation, which can be applied to information retrieval in radio programmes. For instance, across different broadcasters, there is variation in audio quality, dynamic range compression, loudness difference between speech and background music, and so on. Therefore, in this chapter, we primarily investigate two methods for domain generalisation — transfer learning and domain-adversarial training.

5.1 Introduction

In chapter 3, we presented a method to artificially synthesise training examples that resemble radio broadcast. For the experiment in chapter 3, section 3.3, we adopted only synthetic examples for training and collected real-world data from BBC Radio Devon and MuSpeak (MuSpeak Team, 2015) for validation and testing. Note that in this case, the training set belongs to a different data distribution compared to the validation and test sets. In other words, there is a *domain shift* from synthetic data to real-world data. Another related example in Computer Vision is a domain shift from cartoons to real images. For example, if an animal-classifier only learns from drawings of lions, it may be unable to identify a real lion. Nevertheless, our music-speech detection algorithm obtained an overall F-measure of greater than 96% on our in-house test set by training only on synthetic data. This indicates that the data repository that we used for training set synthesis was diverse and comprised a variety of domains. For instance, under music, we included instrumental music, music with singing voice, a capella, different solo instruments, etc. This made the algorithm robust to domain shifts.

As mentioned earlier, there are broadly two ways to approach a machine learning problem — (1) data-centric and (2) model-centric. The above paragraph explained our data-centric approach by including various datasets in our data repository. However, the list of all possible domains is infinite. In section 3.6, we included an additional 9 h of real-world radio data from BBC Radio Devon to train the machine learning model. This improved the overall F-measure of the system by 0.5%, but it was a time-consuming process due to the time taken for annotating data. Incorporating more data from a different radio station may show further improvement in performance, but this makes the process more expensive. Therefore, researchers have also explored model-centric approaches that aim to address the limitations of the training set.

5.2 Methods to Address Domain Shifts

- **Multi-task learning** aims at optimising the same machine learning model for more than one task. As the same model is simultaneously optimising the loss for different sub-problems, it creates a shared representation of domains in the training set, which effectively improves domain generalisation. Bhattacharjee et al. (2022) developed a study on music-speech detection that adopts multi-task learning. It trains the music-speech detection model on auxiliary tasks such as harmonic-percussive source separation features to improve its generalisation.

- **Domain adaptation** refers to the scenario where there is a source domain and target domain. The source domain comprises the training set that is generally fully labelled. Subsequently, the model is tested on the target domain, which belongs to a different distribution. The literature has investigated numerous ways to perform domain adaptation. A couple of approaches include learning a mapping from the source domain to the target domain (Saenko et al., 2010; Kulis et al., 2011) or creating a shared latent space that is domain-invariant (Long et al., 2013; Baktashmotlagh et al., 2013). Many studies have explored the scenario where unlabelled data from the target domain is available (Ganin et al., 2016). This is referred to as unsupervised domain adaptation. The model is optimised to learn a domain-invariant representation using these unlabelled examples. Some studies have also approached domain adaptation as a few-shot learning problem, where there are only few labelled examples from the target domain (Motiian et al., 2017b). Note that few-shot domain adaptation is also commonly used in the context of learning novel classes in new domains. For instance, the training set does not contain certain acoustic classes, but needs to learn from few examples (Wang et al., 2020).

- **Transfer learning** adopts a pre-trained model on the source domain and

fine-tunes it to a new but related task on the target domain. An explanation is already detailed in section 2.7.5. This technique is also applicable for domain adaptation, where there is limited availability of target domain data. As the source domain enables the model to learn higher-level features, it improves the model's generalisation.

- **Meta-learning** is popularly referred to as learning-to-learn. It explores how systems can learn from experiences by performing suitable tasks (Vilalta & Drissi, 2002). Li et al. (2018) proposed a meta-learning framework that simulated train/test domain shift during training by synthesising virtual testing domains within each mini-batch. This made the model robust to domain shifts and thus, improving domain generalisation. Meta-learning has also been investigated in the audio field for sound event detection (Shi et al., 2020), source separation (Samuel et al., 2020) and anomaly detection (Wichern et al., 2021).

5.3 Domain-Adversarial Training

Ganin et al. (2016) proposed the idea of using a Gradient Reversal Layer (GRL) to perform domain-adversarial training of neural networks. The GRL negates the gradient by multiplying it with a negative number $-\lambda$, where λ is the domain adaptation parameter. The first few layers of such a network comprise the feature extractor. Subsequently, the network splits into two branches — a forward branch and the reverse branch, as shown in Figure 5.1. The forward branch is a normal label classifier, but the reverse branch is a domain classifier. The feature extractor is connected to the reverse branch through a GRL. This process of domain-adversarial training fools the network into developing a domain-invariant feature extractor.

Note that domain-adversarial training can be used in two scenarios — unsupervised domain adaptation and semi-supervised domain adaptation. In the former case, there are no labels for examples in the target domain. Therefore, the

label classifier is trained only on source domain examples and the domain classifier is trained on both source and target domain examples. In the semi-supervised scenario, some target domain examples are labelled and the other examples are unlabelled. Domain-adversarial approaches can be also used in this case, where the network is trained on all the known examples.

While the majority of the research into domain adversarial neural networks has been performed on images (Ganin et al., 2016; Matsuura & Harada, 2020; Sicilia et al., 2021), some studies have also explored it for sound event detection (Yang et al., 2020; Cornell et al., 2020) and sound localisation (He et al., 2019). For sound event detection, researchers have used a combination of strong and weak labels for adversarial training (Yang et al., 2020; Cornell et al., 2020). Strong labels mean that the audio file contains information about the audio class and its temporal information (onset and offset times). Weak labels contain information only on the audio class. For example, the training set may contain synthetic data that is strongly labelled and real-world data that is weakly labelled.

In this chapter, we investigate if domain-adversarial training is advantageous in a fully supervised learning setting. This scenario occurs in many studies where a combination of real-world and artificial examples is used for training. While studies have addressed the case where real-world data is weakly-labelled, less attention has been given to the case where the real-world data is also strongly labelled. In section 3.6, we showed that training the neural network on real-world and artificial radio examples obtained the best performance. However, no domain-information was exploited while training. Therefore, in this chapter, we train the neural network with domain-adversarial information to explore its advantages.

In addition to domain-adversarial training, we also investigate if transfer learning helps in domain generalisation. In section 4.3, our You Only Hear Once (YOHO) algorithm had adapted the MobileNet architecture. Another pre-trained model that adopts the MobileNet architecture is YamNet (Plakal & Ellis, 2020). YamNet is trained on AudioSet (Hershey et al., 2017; Gemmeke et al., 2017), which

has over 632 audio classes such as music, speech, vehicle, splinter, toothbrush, and so on. Hence, in this chapter, we initialise the YOHO model with YamNet weights and explore fine-tuning strategies to improve domain generalisation.

5.4 Experimental Setup

5.4.1 Domain Adversarial Neural Network

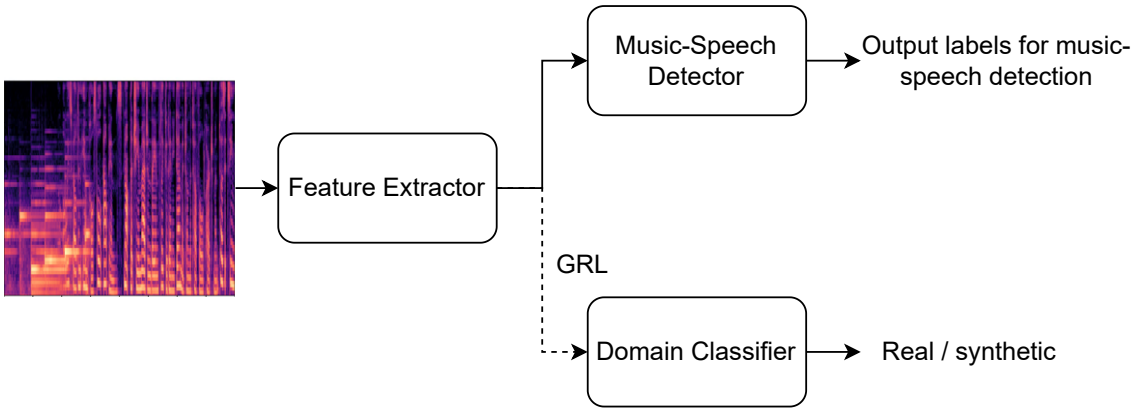


Figure 5.1: The domain adversarial neural network used for music-speech detection. GRL stands for the Gradient Reversal Layer. The music-speech detector and domain classifier are optimised during training. During inference, only the output of the music-speech detector is considered.

Figure 5.1 shows how we adapt our neural network architecture for domain-adversarial training. The structures of the feature extractor, music-speech detector, and domain classifier are clearly shown in table 5.1. We had adapted the MobileNet architecture to realise the YOHO algorithm in section 4.3. Table 5.1 adopts an identical structure for the feature extractor and the music-speech detector. The feature extractor is connected to the domain classifier through a GRL. The domain classifier has a series of convolutional layers followed by global average pooling. The output of the domain classifier uses a sigmoidal activation to detect whether the domain is real or synthetic.

Table 5.1: The neural network architecture for domain-adversarial training. The architecture has three parts — (1) Feature extractor (2) Music-Speech detector and (3) Domain classifier. The feature extractor and the music-speech detector are the same as the YOHO architecture explained in section 4.3.2. The domain classifier performs a binary classification with a sigmoid activation.

Branch	Layer type	Filters	Shape / Stride	Output shape
Feature Extractor	Reshape	-	-	801 x 64 x 1
	Conv2D	32	3 x 3 / 2	401 x 32 x 1
	Conv2D-dw	-	3 x 3	401 x 32 x 1
	Conv2D	64	1 x 1	401 x 32 x 1
	⋮	⋮	⋮	⋮
	Conv2D-dw	-	3 x 3	26 x 2 x 1024
	Conv2D	1024	1 x 1	26 x 2 x 1024
Music-Speech Detector	Conv2D-dw	-	3 x 3	26 x 2 x 1024
	Conv2D	512	1 x 1	26 x 2 x 512
	Conv2D-dw	-	3 x 3	26 x 2 x 512
	Conv2D	256	1 x 1	26 x 2 x 256
	Conv2D-dw	-	3 x 3	26 x 2 x 256
	Conv2D	128	1 x 1	26 x 2 x 128
	Reshape		-	26 x 256
	Conv1D	6	1	26 x 6
Domain Classifier	GRL	-	-	-
	Conv2D-dw	-	3 x 3	26 x 2 x 1024
	Conv2D	512	1 x 1	26 x 2 x 512
	Conv2D-dw	-	3 x 3	26 x 2 x 512
	Conv2D	256	1 x 1	26 x 2 x 256
	Conv2D-dw	-	3 x 3	26 x 2 x 256
	Conv2D	128	1 x 1	26 x 2 x 128
	GlobalAvgPool2D	-	-	128
	Dense	1	1	1

5.4.2 Audio Features and Training Strategy

As we are exploring transfer learning using YamNet, we need to use the same configuration for computing mel spectrograms. This was already done in section 4.5.1 — audio was resampled to 16 kHz, 64 log-mel bins between 125 Hz to 7.5 kHz with a hop size of 10 ms and a window size of 25 ms.

As mentioned earlier, all layers in the feature extractor are also present in YamNet. Hence, we initialise the feature extractor's weights to those of YamNet. The music-speech detector and domain classifier are randomly initialised from

a uniform distribution (Glorot & Bengio, 2010). We used the Adam optimiser to train the neural network. In the first phase of training the network, we freeze the weights of the feature extractor and only train the music-speech detector and the domain classifier with a learning rate of 10^{-3}. After this phase of training, we perform fine-tuning. Here, we unfreeze the weights of the entire neural network and train it with a learning rate of 10^{-4}. During the first phase of training, the domain adaptation parameter λ was set to 0 to independently train the music-speech detector and the label classifier without adversarial training. During the second phase of training, which is fine-tuning, λ was set to 0.31. Both phases of training were performed with early stopping and a patience of 15.

5.4.3 Datasets

The validation and test sets adopted for this experiment were similar to the setup in section 4.5.1. There was an in-house test set, which contains data from BBC Radio Devon and MuSpeak (MuSpeak Team, 2015). There was a second test set, which is the MIREX music-speech detection dataset, containing 27 h of audio from various TV programs. Note that the in-house test set is a *seen* domain because the training set also contains other data from BBC Radio Devon and MuSpeak. However, the second test set would provide us with valuable insights into how the model generalises to *unseen* domains because the network has never encountered this data distribution.

5.5 Results

5.5.1 In-house Test Set

Table 5.2 shows the results on the in-house test set. The table compares three models — YOHO, YOHO pre-trained with YamNet weights, and YOHO combined with adversarial training and with pre-trained weights. The results show that transfer learning and domain-adversarial training do not significantly improve the performance of the algorithm. YOHO with pre-trained weights obtained a slightly lower overall F-measure of 97.19%, compared to 97.22%. However, there is a slight improvement in speech F-measure from 94.89% to 95.10%. YOHO with pre-trained weights and adversarial training obtained the highest overall F-measure of 97.27%. However, such a slight improvement does convey any real benefit of domain-adversarial training. A potential reason for such a marginal improvement is that the test set is a *seen* domain. The training set contains examples from BBC Radio Devon and MuSpeak. Hence, the below sub-section evaluates the model on an *unseen* domain.

Table 5.2: Results on our in-house test set for music-speech detection. It compares performances of YOHO, YOHO pre-trained with YamNet weights, and YOHO combined with adversarial training and with pre-trained weights. The F-measures for overall, music, and speech are presented as percentages. The values in bold indicate the largest number in each column.

Algorithm	$F_{overall}$	F_{music}	F_{speech}
Pre-train with YamNet + Domain-adversarial training	**97.27**	98.07	**95.35**
Pre-train with YamNet	97.19	98.06	95.10
YOHO	97.22	**98.20**	94.89

5.5.2 MIREX Music-Speech Detection

Table 5.3 presents the results of three models on the MIREX music-speech detection competition dataset 2018. In this case, the differences in performance

are more noticeable. YOHO pre-trained with YamNet weights surpasses the YOHO model by more than 1% for the overall F-measure. Furthermore, including domain-adversarial training obtains an overall F-measure of 92.05%, which is an improvement of almost 2%. Considering class-wise metrics, the music F-measure improves from 85.66% to 88.77%. Furthermore, the speech F-measure improves from 93.18% to 94.13%. Therefore, we obtain the state-of-the-art performance on the MIREX competition dataset by combining transfer learning and domain-adversarial training. This demonstrates that both these techniques assist in generalising to *unseen* domains.

Table 5.3: Evaluation on the MIREX music-speech detection dataset 2018. It compares performances of YOHO, YOHO pre-trained with YamNet weights, and YOHO combined with adversarial training and with pre-trained weights. The F-measures for overall, music, and speech are presented as percentages. The values in bold indicate the largest number in each column.

Algorithm	$F_{overall}$	F_{music}	F_{speech}
Pre-train with YamNet + Domain-adversarial training	**92.05**	**88.77**	**94.13**
Pre-train with YamNet	91.34	87.48	93.18
YOHO	90.20	85.66	93.18

5.6 Discussion

In this chapter, we investigated two methods for domain generalisation — transfer learning and domain-adversarial training. We observed that both methods did not show improvement in the in-house test set because of similar data distributions in the training and test sets. However, we observed a 2% improvement in the overall F-measure on the MIREX competition dataset, which belongs to a different data distribution. Note that YamNet was originally trained for only audio classification. Yet, we found advantages in using it for audio segmentation. These findings contribute to answering the second research question — *"How can we advance state-of-the-art algorithms by investigating novel pattern recognition problems?"*.

We explored domain-adversarial training in a slightly different context com-

pared to most studies in the literature. We considered the case where synthetic and real-world data are fully labelled for training. There are multiple studies in the literature that adopt a training set with a combination of strongly labelled real-world data and synthetic data (Bhattacharjee et al., 2022; Lemaire & Holzapfel, 2019; Venkatesh et al., 2021b). Hence, the models in such scenarios can be improved by domain-adversarial training.

In addition to domain-adversarial training, I had also investigated other techniques in the literature. For instance, Motiian et al. (2017a) presented a novel loss function called Classification and Contrastive Semantic Alignment (CCSA), which is a unified framework for domain adaptation and generalisation. CCSA exploits the Siamese architecture (Koch et al., 2015) to learn an embedding subspace that combines the advantages of semantic alignment and separation between domains. In initial experiments, this technique was challenging to adapt for audio segmentation because of the temporal context. To perform semantic alignment, we need to find multiple examples with the same audio classes and transition points. However, this may be more suitable for simple audio classification, which does not require temporal context.

For future research, interesting techniques such as lifelong learning (Biesialska et al., 2020) may be relevant for radio signals. As the target domain can be accessed at each time step, it may improve the model's generalisation over time. However, we need to consider computational load if this model is being deployed for real-time applications.

5.7 Publications and Contributions

- Venkatesh, S., Wichern, G., Subramanian, A., & Le Roux, J. (2022c). Disentangled surrogate task learning for improved domain generalization in unsupervised anomalous sound detection. Tech. rep., Detection and Classifi-

cation of Acoustic Scenes and Events (DCASE) Challenge

– This is the technical report for our submission to the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge Task 2. We ranked 5th amongst 33 teams that participated in the challenge. The title of the task was 'Unsupervised Anomalous Sound Detection for Machine Condition Monitoring Applying Domain Generalisation Techniques'. We proposed a novel multi-task learning framework that disentangles domain-shared features and domain-specific features. Disentanglement leads to better latent features and also increases flexibility in post-processing due to the availability of multiple embedding spaces. In addition to disentanglement, we also investigated machine-specific domain generalisation methods. This included adversarial training (Ganin et al., 2016), simple multi-task learning, and additive angular margin loss (Deng et al., 2019).

Chapter 6

Intelligent Mixing

In this chapter, we address the third research question — *"To what extent can deep learning be used to improve intelligent mixing approaches?"*. The first part of this chapter fulfils the requirements of RadioMe to perform real-time remixing of radio programmes. This is achieved through mainly two features — (1) remix diary reminders in the radio stream to help them remember daily tasks and (2) in case they are undergoing an episode of agitation, play calming music to ease their symptoms.

The second part of this chapter investigates how individuals can communicate with intelligent mixing tools through spoken language or non-technical terms. Therefore, we explore a novel idea of using word embeddings to represent semantic descriptors. Using this technique, the machine learning model can also generate EQ settings for semantic descriptors that it has not seen before. We compare the parameters selected by humans with the predictions of the neural network to evaluate the quality of predictions. The results showed that the embedding layer enables the neural network to understand semantic descriptors. This study on using word embeddings for automatic EQ mixing was published in the *Journal of the Audio Engineering Society*.

6.1 Introduction to Intelligent Mixing

The process of audio mixing involves multiple tasks such as balancing sound levels and applying audio effects. It is applicable to making albums, radio programmes, TV shows, film-making, and many more tasks. Audio mixing can be performed in real-time for live shows and radio programmes where levels are adjusted by the DJ, or in a studio setting to create well-produced albums and films. A vast body of research has been exploring how this process can be automated through the use of intelligent tools (De Man et al., 2019; Moffat et al., 2018; Perez-Gonzalez & Reiss, 2009). Traditional AI approaches such as expert systems have been adopted to create autonomous mixing tools (De Man & Reiss, 2013). These systems are knowledge-engineered and adopt a set of rules for mixing depending on the scenario. However, recent research has grown towards using Machine Learning and Deep Learning for automatic mixing. On one hand, some studies have focused on specific areas, such as gain balancing (Perez-Gonzalez & Reiss, 2009; Moffat & Sandler, 2019a) and reverberation (Chourdakis & Reiss, 2017). On the other hand, some have explored building autonomous systems where the entire mixing process is carried out without human intervention (Ramirez et al., 2021; Moffat & Sandler, 2019b).

Researchers have developed tools to automate the role of a DJ. Thalmann et al. (2018) adopted the Web Audio API to develop an in-browser-based automatic DJ. It extracted structural representations such as harmonic similarity, tempo, and loudness to create mixes of songs with appropriate transitions. Bittner et al. (2017) emphasised the lack of time and expertise of users to curate thoughtful and well-sequenced playlists. Hence, they presented a method by combining graph models and optimisation to automatically sequence existing playlists and add DJ-style cross-fade transitions. Studies have also explored remixing from the perspective of audio enhancement, where gain levels of individual sources are re-adjusted after source separation (Torcoli et al., 2021). However, we are not aware of a study

that has adopted audio segmentation to improve intelligent mixing. Instead, most studies have used MIR tasks that analyse musical information such as tempo, pitch, etc (Thalmann et al., 2018). Therefore, in this chapter, we explore if audio segmentation can improve real-time remixing approaches.

6.2 RadioMe Audio Engine

There are two objectives behind the remixing — (1) play diary reminders to assist individuals in remembering their daily tasks and (2) play music from their personal playlist if they are undergoing an episode of agitation in order to calm them down. This remixing needs to be integrated with the live stream of radio broadcasts like BBC Radio Devon, Jazz FM, *Kerrang! Radio*, and so on. The focus of this thesis is to only build the audio technology and not directly work with people with dementia. The audio system is submitted to researchers at the Cambridge Institute for Music Therapy Research, Anglia Ruskin University, UK, for them to conduct tests with people. The purpose of this system is to demonstrate how intelligent remixing can be performed with underlying audio segmentation in real-time. Therefore, this technology is generic and can be applied to areas beyond people with dementia.

After a few meetings with broadcasters and understanding their thoughts, we learned that it would be difficult to utilise metadata from broadcasters. Most of them do not broadcast metadata along with the audio and even if they do, it may not be in sync with the live radio programme. This is because, there are always some minor deviations from the schedule, which makes it difficult to develop a straightforward temporal mapping between the metadata and the audio stream. Some researchers have also explored Object-based broadcasting (Armstrong et al., 2014), where media is represented by a set of individual assets. These individual assets are assembled at the user's device to create an improved user-experience. After discussions with broadcasters regarding using object-based media for the RadioMe system, we realised that this is still a developing concept and still not

adopted widely in practice. Furthermore, object-based media would be specific to radio stations, which would make it harder to develop a generic system applicable to multiple radio streams.

6.2.1 Streaming Radio

The RadioMe audio engine streams live internet radio through FFmpeg[1]. It stores data in chunks of 8 s, which was the duration of audio examples for experiments in earlier chapters. As we are using FFmpeg, we only require the target URL of the radio station to stream it. It can be any radio broadcaster. The audio engine also has a music-speech detector running in real-time, which analyses these chunks of 8 s. We have already demonstrated the speed and robustness of the YOHO algorithm in section 4.6.4, which made it suitable for real-time music-speech detection. In the main program, it took approximately 0.12 s to perform music-speech detection on 8 s of audio.

6.2.2 Remix Diary Reminders

The diary system was developed by our collaborators at the University of Glasgow. It was integrated with Google calendar, where the events of the day could be updated in the user's Google account. This way, the individual's family members or the carer can easily add and modify events in the calendar. There was a programme that fetches information from the calendar every minute. When the reminder for an event is due, it is flagged in the remixing programme.

These diary reminders require corresponding audio files that can be played by the audio engine. Initially, we adopted one of the state-of-the-art models for speech synthesis through TensorFlowTTS[2]. However, the general feedback from our collaborators was that the speech sounded artificial and may not be suitable

[1]FFmpeg: https://ffmpeg.org/
[2]https://github.com/TensorSpeech/TensorFlowTTS

for people with dementia. Therefore, it was appropriate to use the recording of a person speaking the diary reminder. This task is slightly harder because we need to manually record diary reminders for corresponding events. However, this was the preferred choice amongst the researchers because it made the radio sound more natural. Moreover, we included a bell sound that indicates a reminder is starting and announce — *"Hi! This is Jared.. I have a reminder for you.. Check the calendar."*. After the reminder, the bell sound is played again.

There are four possible cases that can be detected by the music-speech detector — music, speech, speech+music, and none. Below are the remixing rules adopted by the audio engine for each of the scenarios.

1. **Music**: If the radio stream has solely music going on, it is ducked such that there is a loudness difference of 15 dB between the diary reminder and music. The diary reminder is played over the ducked music. Figure 6.1 illustrates how diary reminders are inserted over music.

2. **Speech**: If the radio stream has someone speaking, we cannot adopt the same mixing procedure. A radio DJ speaking behind the diary reminder would be confusing. Therefore, we wait for a point until the music-speech detector finds a pause of greater than 0.4 s. Remixing at pauses of speech ensures that we do not cut-off a speaker in the middle of a sentence. After finding a pause, the radio stream is interrupted, the diary reminder is played, and the radio is resumed from where it was interrupted. Note that there is a slight delay (a few seconds) in the live radio stream after this occurs. However, we found this delay to be negligible and it would be reset each time the RadioMe system is switched off and switched on again.

3. **Speech+Music**: This scenario generally occurs in advertisements and radio jingles, where the radio DJ speaks over music. As these scenarios generally last for short durations, we decided it is best to not perform any remixing and wait until the radio plays only speech or music. Furthermore, it is tricky to remix diary reminders over the radio signal where there is music and

speech simultaneously playing.

4. **None**: The music-speech detector can output *none* if there is a period of silence or noise/sounds effects are being played. However, we found this scenario to be extremely rare. We followed the same mixing rules for *music*, where the radio signal is ducked and the diary reminder is played over it.

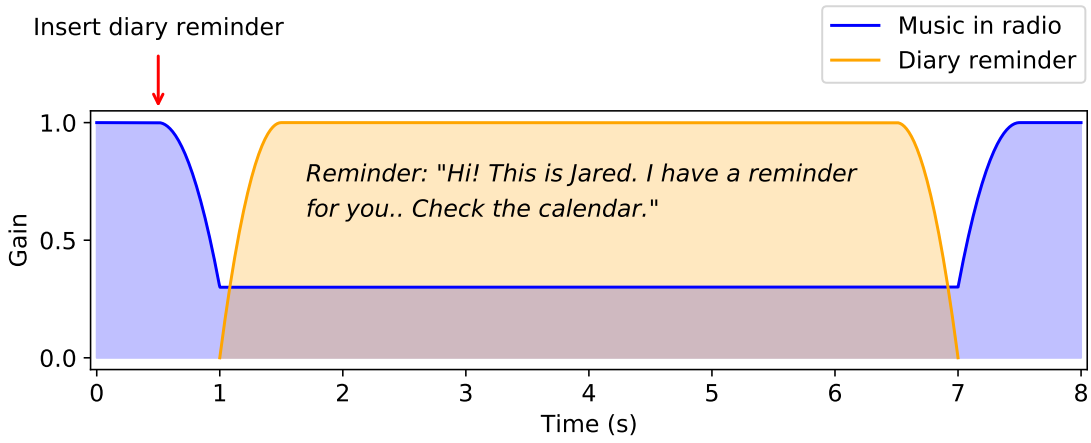


Figure 6.1: An illustration of how diary reminders are remixed in the radio programme when music is detected by the audio segmentation algorithm.

6.2.3 Remix Playlist

The second aspect of real-time radio remixing is to play calming music if the person is undergoing an episode of agitation. The agitation detection system was also developed by our collaborators at the University of Glasgow. They monitor heart rate through a smartwatch sensor to detect if a person is agitated. If agitation is detected, it is flagged to the RadioMe audio engine. As agitation needs to be addressed immediately, we did not analyse the output of the music-speech detector. Instead, the live stream of the radio is gradually faded out and music from the individual's personal playlist is played. Songs from the playlist are stored locally on the computer. This playlist is unique to each individual.

6.2.4 Discussion

The RadioMe project had planned a series of initial experiments to demonstrate the proof-of-concept. This was an offline version of RadioMe and the experiment was titled *Wizard of Oz*. For this, a 90-minute CD recording of RadioMe was to be deployed in the home of a person with dementia. In this setting, there were three diary reminders and two interruptions from the playlist as shown in table 6.1. The radio station was BBC Radio Cambridge, which was selected by the person with dementia. This experiment with the person with dementia was carried out by our collaborators at Anglia Ruskin University. The feedback from the individual mentioned that the diary reminders played during the radio were not obvious enough for them to pay sufficient attention. Although we had included a bell sound to alert the individual, this may not be sufficient when the person is performing their daily tasks while listening to the radio. Therefore, for future experiments it would be interesting to play a RadioMe jingle that captures the listener's attention to a greater extent, to make it obvious that a reminder is approaching.

Table 6.1: Schedule of the 90-minute CD recording demo.

Time	**Event**
10:40 am	Recording starts
10:44 am	Reminder: check the calendar
11:07 am	Song from playlist: Lady in Red (Chris De Burgh)
11:19 am	Reminder: Water the flowerpots
11:33 am	Song from playlist: Take the A train (Ted Heath)
11:55 am	Reminder: Make a cup of tea
12:10 pm	Recording ends

These initial experiments with RadioMe conveyed that people with dementia have their own preferences for intelligent remixing. Although the explanation of their preferences maybe non-technical, they were still able to make us understand in their own words. This motivated us to investigate how individuals can directly communicate with intelligent mixing tools with non-technical language. For example, a person may say — *"I cannot hear the diary reminder"*. However, if this is

put in technical terms, the loudness difference between the diary reminder and the background is not sufficient to facilitate clear speech.

Currently, in the RadioMe audio engine, we are mainly performing two audio effects — gain-balancing of audio tracks and applying audio fades. It would be interesting to go beyond these effects to develop AI tools for equalisation (EQ) and reverberation (Chourdakis & Reiss, 2017). For example, Torcoli et al. (2019) demonstrated that the elderly population preferred lower background music compared to a younger population. These preferences can be further addressed by incorporating bespoke EQ settings.

The use of words or semantic descriptors to describe audio effects is more common for EQ, dynamic range compression, and reverberation, compared to gain-balancing. Therefore, to study how non-technical descriptors can be used for automatic mixing, we mainly explore EQ because the literature has datasets for this task.

6.3 Word Embeddings for EQ

An equalizer (EQ) is an audio effect created by cascading multiple filters in series (Tarr, 2018). Timbral adjectives often have a correlation with the parameter setting for the equalizer. Some examples include, *add air*, *make it warmer*, and *make it less muddy* (Spyridon, 2019). Kulka (1972) associated adjectives such as *warmth*, *honk*, *crunch*, and *sibilance* with frequencies of 125, 500, 2000, and 8000 Hz respectively. For example, according to the Kulka rule, if the mix sounds honky, cut the region around 500 Hz.

When clients such as instrumentalists and musical directors work with mixing engineers, they often use semantic descriptors to describe their goals. For example, *"make the violin sound warmer"* (Cartwright & Pardo, 2013). It is the role of the mixing engineer to understand these descriptors mentioned by the client.

Popular semantic descriptors such as *warm* and *bright* are easily understood by the mixing engineer (Bromham et al., 2019). To expand the vocabulary of such descriptors, studies have also tried to create a thesaurus with synonyms and antonyms. For example, significant synonyms of *boom* are *boxy, dull, and fat* and significant antonyms of *boom* are *air, bright, and crisp* (Spyridon, 2019). However, the problem arises when individuals without training in audio production describe their creative goals (Zheng et al., 2016). They may have ideas that cannot be directly translated into a studio engineer's vocabulary.

To address this issue of non-technical descriptors, Cartwright & Pardo (2013) presented a dataset called SocialEQ, which is a web-based project that adopts crowd-sourcing to learn a vocabulary of audio descriptors. As it is crowd-sourced, the study focuses on aggregating a vocabulary to enable non-technical individuals to describe their sonic goals. Crowd-sourcing was also adopted to build the datasets for other effects like reverberation (Seetharaman & Pardo, 2014) and dynamic range compression (Zheng et al., 2016).

There is a growing interest in adopting natural language processing (NLP) methodologies to develop semantically-controlled audio effects (Zacharakis et al., 2012; Williams & Brookes, 2007, 2009; Antoine et al., 2016; Miranda, 1995). Stables et al. (2014) presented a system called Semantic Audio Feature Extraction (SAFE), which focused on extracting semantic descriptions for equalization from a digital audio workstation (DAW). Stasis et al. (2016) investigated the idea of mapping the descriptors to a reduced dimensionality space, to enable users to interact with the system in a more intuitive way. Chourdakis & Reiss (2019) explored tagging and retrieval of room impulse responses for reverberation. They adopted word embeddings to assign impulse responses to tags that match their short descriptions.

In this chapter, we explore the novel idea of adopting word embeddings to automatically predict EQ settings. We present a methodology to translate words from a semantic vector space to a vector space representing the parameters

of an equalizer. Word embeddings are representations of words that capture lexical semantics in language (Bakarov, 2018). An embedding layer is often used as the first layer in a neural network that performs NLP tasks such as machine translation, caption generation, and automatic speech recognition (Goldberg, 2017). Although word embeddings are commonly used to understand natural language, we investigate if they would be of any benefit to descriptors for EQ settings. We adopt this approach to translate words to predict values of a parametric equalizer. This way, the neural network has the ability to understand non-technical words and even descriptors that it has not seen before. This finding is significant because artists without training in audio production can express their creative goals directly to the AI-powered mixing engine. To our knowledge, this is the first study that investigates how EQ settings can be predicted for *unseen* semantic descriptors. We demonstrate that the neural network is capable of learning a direct translation from the text domain to the EQ domain.

6.4 Experimental Setup

6.4.1 Dataset

We adopted the SocialEQ dataset (Cartwright & Pardo, 2013), which crowd-sources semantic descriptors for EQ settings. In the raw format, each sample in the dataset contains a semantic descriptor, language of the descriptor, audio id, a consistency rating, and 40 values for EQ parameters. During the data collection, each participant was asked to enter a word in their preferred language. For example, *warm* in English, *claro* in Spanish, or *grave* in Italian. Subsequently, they pick a sound file which will be modified by the EQ plugin. There were three sound files — electric guitar, piano, and drums. Each sound file had a unique audio id.

After selecting a descriptive term and audio file, the participant was presented

with 40 different modifications of the sound file made by different EQ settings. Suppose the user has selected *warm*, they are asked to rate *how warm that sound is*. Out of the 40 modifications, there are 15 repetitions to test for consistency. Consistency score was calculated using Pearson correlation between the ratings of the test and repeated examples. The system processes the ratings of the user and develops a relative boost/cut for 40 different frequency bands. Refer to the study by Cartwright & Pardo (2013) for more details on the dataset.

The dataset has 1595 samples in it. For simplicity, we considered only descriptors in English. The number of examples in English was 918. It is important to note that the dataset contained examples with different EQ parameter settings for the same word. Thus, the number of unique descriptors in English was 388.

6.4.2 Train-Test Split

An important hypothesis we wanted to test in this chapter is that a word embedding layer helps a model predict EQ parameter settings for semantic descriptors it has not seen before. Therefore, words in the test set should not appear in the training set. We adopted a four-fold cross-validation setup (Forman & Scholz, 2010) and the strategy is explained below.

We aggregated a list of semantic descriptors that are common in the audio mixing literature. We labelled these as High Quality (HQ) words. In order to avoid bias and objectively choose these words, we selected those that were already listed in table 4.8 in the study by Spyridon (2019). Additionally, we included semantic descriptors that fell under the hierarchical ontology presented by Pearce et al. (2016). The list of HQ words is presented in bold in table 6.2. There are 32 HQ words present in the SocialEQ dataset.

We also aggregated a list of words that were Highly-Rated (HR). HR words need be not semantically meaningful, but have a high consistency score in the

Table 6.2: Four cross-validation folds from the dataset. The test words from each fold are presented in the table. For each fold, the training set consists of words that are not in the test set. High Quality (HQ) words are indicated in bold, as explained in section 6.4.2.

Fold 1	Fold 2	Fold 3	Fold 4
smooth, muffled, crisp, punch, clean, brittle, muddy, soothing, clear, brassy, caring, mellow, throbbing, cooing, fluffy, good, excited, squeaking, punchy, funky, whispered, disgusting, beautiful, reserved, serene, thumpy, pleasurable, whispering, gentle, energetic, peace	**crunchy, woody, flat, metallic, dull, tinny, cold, booming, deep,** energizing, heart-warming, edgy, heavy, edge, strong, enchanting, cheerful, plodding, quiet, radiant, biting, brass, pleasing, light, taco, gruff, exciting, love, heat, techno, solemn	**sweet, warm, airy, full, boxy, bright, boom, fat, shrill,** calm, velvety, hard, rich, noisy, down, rumble, sloppy, relaxing, peaceful, romantic, low, hot, thunderous, frigid, happy, poor, cool, tense, jagged, forceful, aggressive	**sharp, big, dark, hollow, harsh, smooth, muffled, crisp, punch,** mournful, clarity, genius, bold, twangy, soft, splash, slow, wistful, brash, fancy, cute, rousing, loud, breezy, large, passionate, baseball, huge, icy, brassy, caring

dataset. Words that have a consistency score greater than 0.7 were selected as HR words. As these words have a high consistency score, the user strongly associated the semantic word with a particular EQ setting. Words in table 6.2 that are not formatted as bold text are HR words. Totally, 86 HR words were present in the SocialEQ dataset.

Each test fold contained 9 HQ words and 22 HR words. We ensured that every HQ and HR word was tested at least once. In the last test fold, there may be a few repetitions of words from the first test fold. There was no overlap between the training set and test set. The test set only contained words that were not present in the training set. Note that the network for each fold is trained as a separate experiment. In other words, the network is totally trained four times and tested four times on different folds and we report the average performance.

As mentioned earlier, each word can have multiple EQ settings. Each setting is a separate example and can have different consistency scores. In the test set, we only included examples that had a consistency score of greater than 0.7. In the training set, we did not exclude any words based on the consistency score.

6.4.3 Word Embeddings

A vocabulary consists of all the possible words that the neural network can understand. Generally, a word is converted into a one-hot encoded vector before passing into the neural network. For instance, in the SocialEQ dataset, there are 388 unique words, which means that the size of the vocabulary is 388. Therefore, the dimensions of the one-hot encoded vector are 1×388. Each position within the vector is assigned to a unique word. Thus, the respective position of the word is labelled as 1 and the remaining elements are 0. However, it is important to note that the Euclidean distance between any pair of words is equal. As each word is equidistant from each other, the neural network would not develop the capability of handling words that is not present in the training set. For example, let us consider the semantic descriptor *bright* and assume that it is present in the training set. Let us also assume that *clear* and *boom* are words in the test set. According to Spyridon (2019), *clear* is a synonym of *bright* and *boom* is an antonym of *bright*. Thus, we expect similar EQ settings for *clear* and *bright*, but considerably different EQ settings *boom* and *bright*. However, the neural network cannot perceive this understanding unless it has seen all three words because each word is equidistant from each other. Furthermore, this issue becomes exaggerated if a non-technical user is utilising a semantic descriptor that is not common in the audio mixing literature.

The purpose of a word embedding layer is to convert a one-hot encoded representation into a vector space of reduced dimensionality. They are useful for NLP tasks such as machine translation. Large vocabularies with millions of words can be reduced to a 300-dimensional vector representation (Pennington et al., 2014). The distances between words in the embedding space are governed by some form of semantic correlation. Examples include synonyms or two words frequently occurring together. There are different algorithms to train word embedding models. Some of them include Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), ConceptNet (Speer & Lowry-Duda, 2017), and Dict2Vec (Tissier et al., 2017).

Each of these algorithms presents unique methods to train on large corpora of text such as Wikipedia. Effectively, they try to learn semantic relationships between words and represent them through an embedding vector.

For this study, we investigated four different embedding models — GloVe-6B, GloVe-840B, Tok2Vec, and Dict2Vec. GloVe is an unsupervised learning algorithm developed to obtain vector representations for words (Pennington et al., 2014). GloVe-6B refers to the model that was trained on Wikipedia 2014 and Gigaword 5. It includes 6B tokens and a vocabulary size of 400k (B, M, k stand for Billion, Million, and thousand respectively). On the other hand, GloVe-840B uses 840B tokens and a vocabulary size of 2.2M. It trains on the World Wide Web using Common Crawl, which is a larger corpus of text. Tok2Vec is a word embedding model provided by a company called spaCy (Honnibal et al., 2020). We did not find the entire details regarding its implementation, but the model is publicly available and free to use. It is important to note that word embeddings are used for NLP tasks, which are designed to accept sentences. In our application, we are considering only one word, which is the semantic descriptor. As GloVe and Tok2Vec also focus on the ordering of words in sentences, we thought it is a good idea to consider another embedding model called Dict2Vec (Tissier et al., 2017). Dict2Vec is an embedding model that uses lexical dictionaries. It builds new word pairs from dictionary entries so that semantically-related words are closer to each other in the embedding space (Tissier et al., 2017). Similar to GloVe-6B, it was trained on the Wikipedia corpus.

6.4.4 Machine Learning Architecture

Word Embedding Layer

We evaluated four different pre-trained word embedding models in the study — GloVe-6B, GloVe-840B, Tok2Vec, and Dict2Vec. All the models represent words

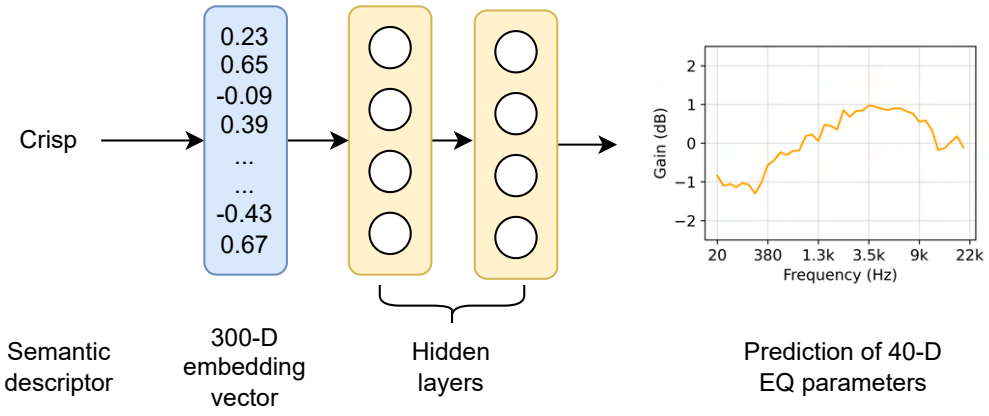


Figure 6.2: A schematic diagram of how the network learns a translation from semantic descriptors to EQ parameters.

Table 6.3: The neural network architecture

Layer type	Units	Activation	Output shape
Embedding	-	-	300
Dense	300	ReLu	300
Dense	200	ReLu	200
Dense	100	ReLu	100
Dense	80	ReLu	80
Dense	60	ReLu	60
Dense	40	Sigmoid	40

with 300-D semantic vectors. This is convenient because we can adopt the same neural network architecture to compare different embeddings. Initially, a word is converted into a one-hot encoded representation. Subsequently, an embedding matrix converts this one-hot encoded representation into a 300-D semantic vector. Then, this vector is connected to hidden layers in the network. Note that the weights of the embedding matrix are frozen and the layer is not trainable. We did not consider setting this to trainable because of the limited data we have.

Hidden Layers

The neural network aims to translate a representation of word embeddings to a prediction of equalizer parameters. Therefore, our network needs to be deep enough to learn the translation between two domains. Deeper networks apply the non-linear activation more number of times on the input and therefore have the

advantage of learning more complex translations. However, it is important to note that our dataset is relatively small for our task.

All the layers in the neural network were fully connected layers. Table 6.3 shows an overview of the architecture. After the embedding layer, we had a series of fully connected layers. The number of hidden units in these layers were 300, 200, 100, 80, and 60 respectively. Finally, it was connected to an output layer with 40 units. Excluding the final layer, all the hidden layers were fitted with ReLu activations and a dropout of 0.1. The output layer is explained in section 6.4.4. The code and trained models associated with this study can be found in this GitHub repository[3].

Normalisation

Traditional min-max normalisation by calculating the maximum and minimum in the training set was not appropriate for our dataset. This is because if there exist any outliers amongst the values in the test set, specific features may get magnified or diminished. Furthermore, as we are predicting values for 40 EQ bands, this issue becomes more crucial. Therefore, we fixed the minimum and maximum value for each EQ parameter to -4 dB and +4 dB respectively. In other words, the highest cut/boost within each EQ band was 4 dB. The values were linearly normalised to the range of 0 to 1. Hence, -4 dB would correspond to 0 and +4 dB would correspond to 1 in the output layer.

Note that normalisation is performed only on the output of the neural network, which is EQ predictions. The input of the neural network is only semantic descriptors, which is directly connected to the word embedding layer.

[3]https://github.com/satvik-venkatesh/word-eq

Table 6.4: The error calculated across four folds. The smallest error in the column is indicated in bold.

Word Embedding	Error
Tok2Vec	**0.760 ± 0.055**
Glove-840	0.770 ± 0.032
Dict2Vec	0.792 ± 0.058
Glove-6B	0.798 ± 0.046
No Embedding	0.836 ± 0.016

Output Layer and Loss Function

The output layer of the network contained 40 neurons, with each of them predicting a value for one EQ band. As we normalised the data within the range of 0 to 1, we used sigmoid activation functions for the output neurons. As we are working with a regression problem, we adopted the mean absolute error loss function. All EQ bands were given equal importance when averaging the error for the loss function. In future work, it would be interesting to weigh the EQ bands based on perceptual frequency band weights. However, that is beyond the scope of this study.

The network was trained using Stochastic Gradient Descent (SGD) with an initial learning rate of 0.1. The learning rate was scaled by 0.96 after every 10,000 weight updates.

6.5 Results

6.5.1 Error

Table 6.4 shows the mean absolute error for different embedding models calculated across four test folds. As we can see, Tok2Vec obtains the lowest error rate of 0.76, followed by GloVe-840 with an error rate of 0.77. GloVe-840 obtains an error lower than GloVe-6B, which conveys that it benefited from training on a larger

corpus. Dict2Vec and GloVe-6B were trained on similar dataset sizes and the former obtained a better error rate. This suggests that the performance of Dict2Vec can be improved with training on a larger corpus of text.

The 'No Embedding' model in table 6.4 means that no word embedding layer was used in the neural network. This can be considered to be the baseline system. As this is the first study that investigates a translation from *unseen* semantic descriptors to EQ settings, there are no state-of-the-art approaches for comparison. The input of the network was a direct one-hot encoded representation. All the neural networks with word embeddings performed better than the model without word embeddings. However, the difference was not huge. The best model was Tok2Vec with an error rate of 0.76 vs 'No Embedding' with an error rate of 0.836. This is possibly due to two reasons. Firstly, error may not be the best metric for our task. For example, the semantic word *warm* may have a boost of 1.2 dB at 260 Hz. But, the neural network may predict a boost at the adjacent EQ band, such as 317 Hz. Although, the error rate in this case is high, the EQ effect applied to the audio may still be semantically meaningful. Secondly, the test set contains many semantic descriptors that occur only once. These examples may be highly subjective to one individual, despite having a high consistency score. Therefore, in the next subsection, we evaluate the top two performing models using Partial Curve Mapping (PCM) (Witowski & Stander, 2012), which is a method to quantify the similarity between two curves. For instance, this technique is generally adopted to analyse similarities between hysteresis curves pertaining to a magnetic field. Although this technique may not be ideal for our task, it would give us a better understanding of our model's performance compared to mean absolute error.

6.5.2 Partial Curve Mapping

In this section, we evaluate the models using PCM. PCM was implemented using this Python package developed by Jekel et al. (2019). We also compare our model

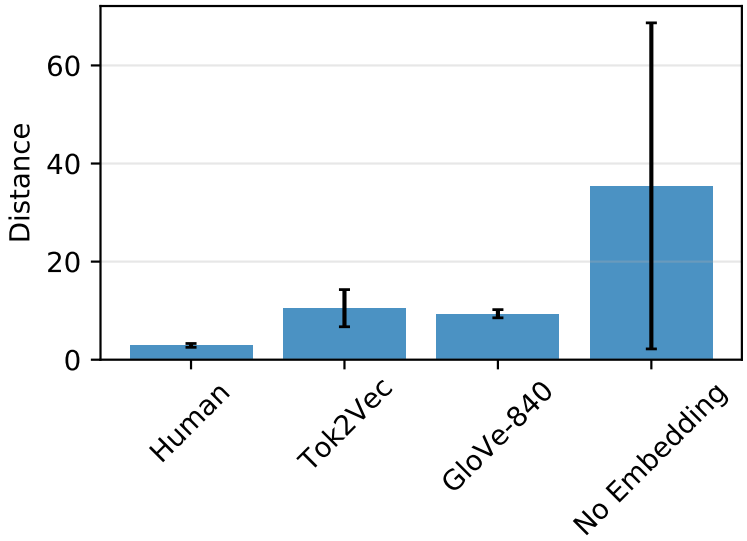


Figure 6.3: Distances obtained by different models calculated by using Partial Curve Mapping (PCM). An ideal algorithm would have a distance of zero.

to human labels. As mentioned earlier, each semantic descriptor had multiple EQ settings in the dataset. To calculate the error in human labels, we considered the mean of the different EQ settings as the ground truth. However, words that occur only once in the dataset would not have an error associated with them. These words would artificially reduce the average error. Hence, we only included words that occur at least twice in the dataset. Figure 6.3 shows the distances for different models. An ideal algorithm would obtain a distance of zero. Human labels obtain the smallest distance of 2.9, which is an expected observation. GloVe and Tok2Vec obtain similar distances with the former performing slightly better. The distances were 9.3 and 10.5 respectively. Note that for this experiment, we only considered words that occur at least twice, which is different from the results presented in section 6.5.1. The mean distance of the model with no embeddings was 35.4, which was considerably higher. In addition, there was a much larger standard deviation for this model, which suggests that it was randomly guessing.

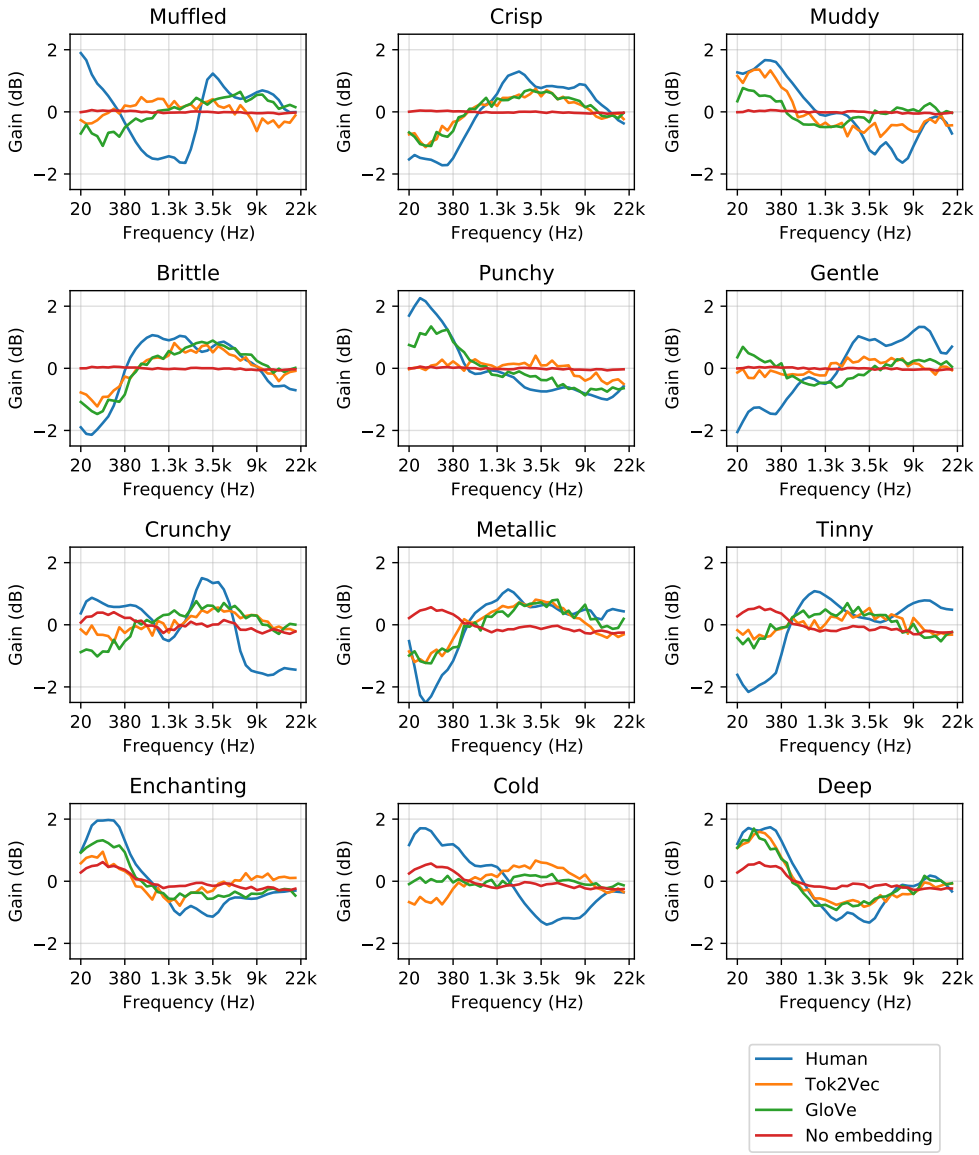


Figure 6.4: Plots of human labels alongside EQ parameters predicted by GloVe-840, Tok2Vec, and no embedding. These are for words in test folds 1 and 2. Note that that each word in the test set does not occur in the training set. The first two rows occur in fold 1 and the last two rows occur in fold 2. The human label plotted for a semantic word was the EQ settings with the highest consistency score in the dataset.

6.5.3 Plots of EQ Parameters

In this section, we perform an error analysis of predictions made by the machine learning models. We look at individual test words to investigate if the neural network is actually learning semantic meanings of EQ settings. We predominantly look at HQ words as they are common in the audio mixing literature and would be more intuitive to evaluate. In figures 6.4 and 6.5, we plot the EQ settings of human labels alongside the predictions of Tok2Vec, Glove-840B, and 'no embedding'. As

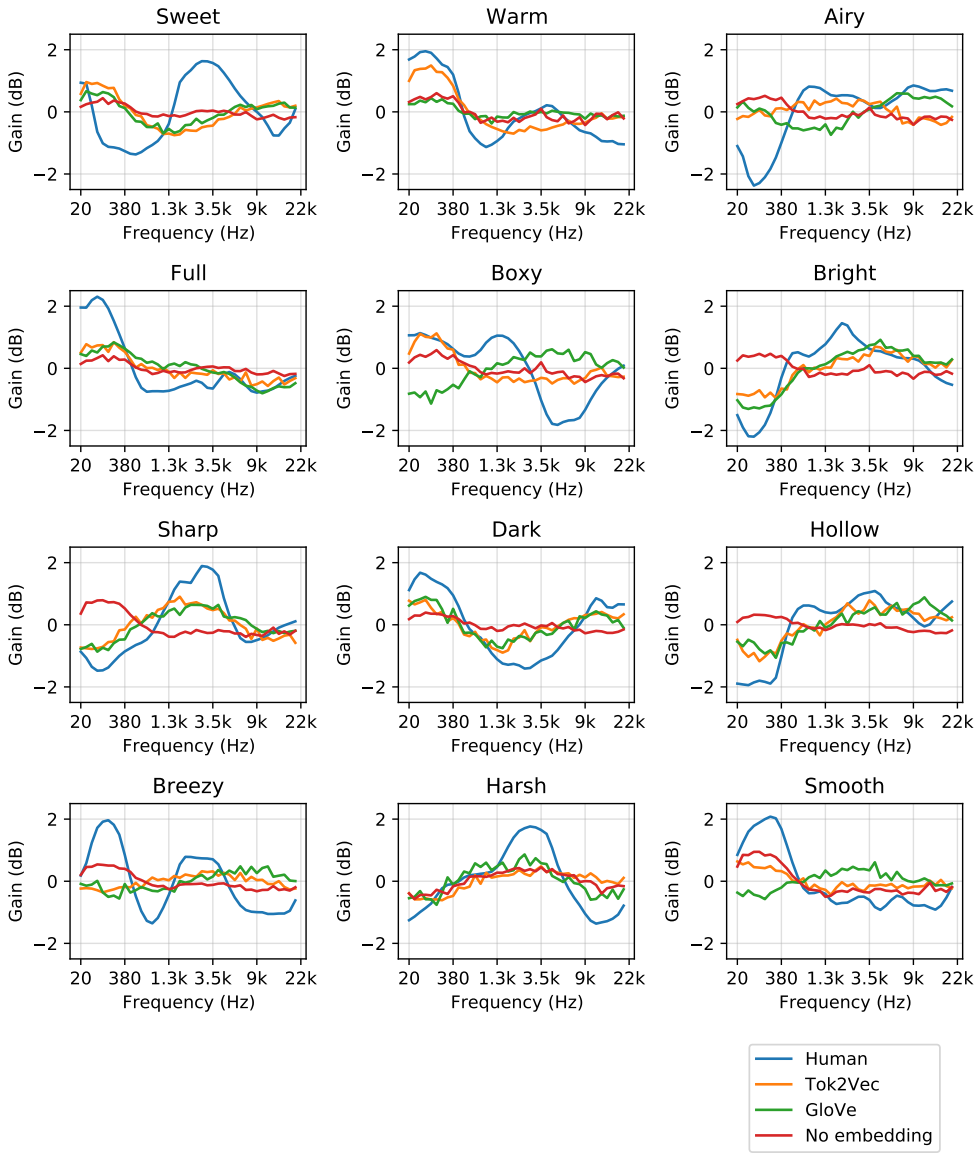


Figure 6.5: Plots of human labels alongside EQ parameters predicted by GloVe-840, Tok2Vec, and no embedding. These are for words in test folds 3 and 4. Note that that each word in the test set does not occur in the training set. The first two rows occur in fold 3 and the last two rows occur in fold 4. The human label plotted for a semantic word was the EQ settings with the highest consistency score in the dataset.

the literature does not comprise an 'ideal' metric for our task of predicting EQ parameters, we thought it is a good idea to plot graphs and actually visualise the predictions of the algorithms. Figure 6.4 plots the graphs for words selected from the test folds 1 and 2. Figure 6.5 plots the graphs for words selected from the test folds 3 and 4. Note that for each word in the test folds, the neural network has not encountered the word in the training set. The human label chosen for each semantic word in the plots was the EQ setting with the highest consistency score in the dataset.

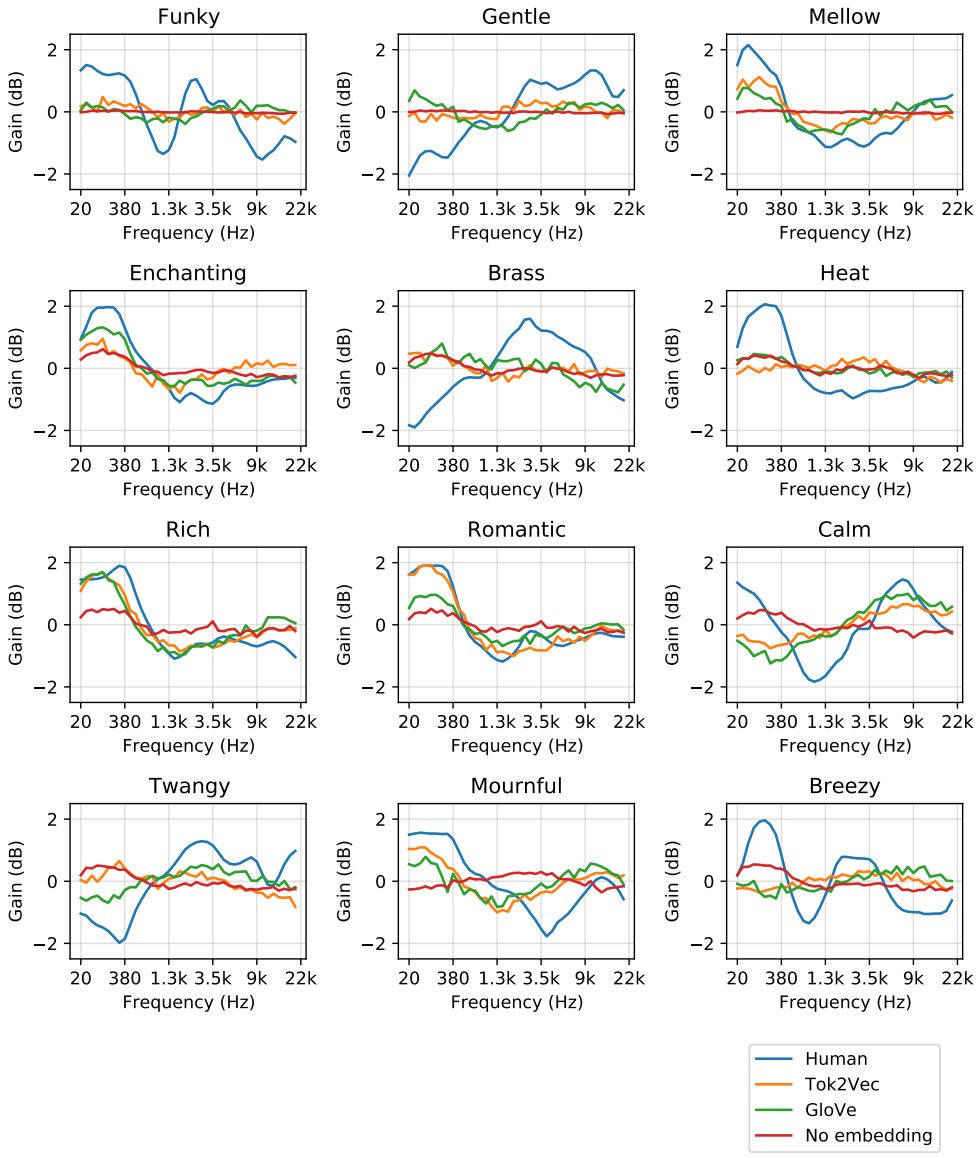


Figure 6.6: Plots of EQ parameters for highly-rated (HR) words as explained in section 6.4.2. These are non-technical words that may be highly subjective to a user.

In Figure 6.4, human labels for *muffled* had boosts at 20 Hz and 3.5 kHz. For Tok2Vec and GloVe, we saw slight boosts in the mid-range and high-range respectively, which may convey that the neural networks did not interpret this word correctly. We also observed that the predictions made by Tok2Vec and GloVe are considerably different from each other. This can be due to two reasons — (1) Tok2Vec and GloVe are different algorithms and therefore, learn different semantic meanings from text (2) there may be a higher degree of randomness in their predictions because the embeddings are trained only on natural text from the Word Wide Web, which is different from EQ descriptors. Hence, the neural network would require more training examples containing EQ descriptors. The

network with 'no embedding' was basically a flat curve for all the words in the first fold. For *crisp*, interestingly, the predictions of Tok2Vec and GloVe did follow a similar pattern as the human labels. In the human labels, we saw boosts at 2.1k and 9k. For GloVe and Tok2Vec, we saw a gradual boost at 3k, which lifts the high-range of the frequency spectrum. Some semantic synonyms of *crisp* present in the training set for this respective fold include *bright*, *harsh*, *hollow*, and *sharp*. This means that the word embedding has delineated a relationship between the semantic word and EQ predictions. Again, as mentioned earlier, we did not observe a meaningful pattern in the neural network with 'no embedding' because the curves were flat.

Muddy had a gradual boost from 200 to 380 Hz in the human labels. Tok2Vec follows a very similar pattern in its prediction by boosting the lows and cutting the highs. GloVe's prediction has slightly boosted lows and highs, which is not convincing for the semantic word *muddy*. Some semantic synonyms in the training set include *boom*, *muddled*, *dark*, *dull*, and *fat*. The next test word, *brittle* was well-understood by both Tok2Vec and GloVe. There was considerable overlap with the human labels. The synonyms for *brittle* in the training set would be similar to those listed for *crisp*. *Punchy* was understood by GloVe, but not by Tok2Vec. *Gentle* was not understood by both embedding models[4].

Crunchy had boosts in the low and high-frequency range in the human labels. We observe a boost for GloVe and Tok2Vec in the high range. The 'no embedding' model has a boost in the low range. However, if you observe, it has made the same prediction for all the test words in the second fold. GloVe and Tok2Vec correctly understood the semantic descriptor *metallic* and have significant overlap with human labels. For *tinny*, human labels have boosts at 1.3k and 9k. Whereas, the neural networks with embeddings have a gradual boost around 3k. We are not certain if these predictions would have a *tinny* effect. For test words *enchanting* and *deep*, we observed a noticeable overlap with human labels. However, for *cold*,

[4]If the reader is interested in more semantic synonyms present in the training set, please refer to table 6.2. If fold 1 is selected as the test set, folds 2, 3, and 4 are included in the training set.

it seems as though GloVe and Tok2Vec predicted the antonym.

In test fold 3, *sweet* was not understood by the networks at all. For *warm*, Tok2Vec has a noticeable overlap with the human labels because both have a boost of approximately 2 dB in the low-frequency range. *Airy* was partially convincing because GloVe recognised a boost at 9 kHz. Although, the networks have boosted the lows for *full*, it seems like a random guess as the prediction significantly overlaps with the one made by 'no embedding'. The predictions made by the networks for *boxy* were not convincing. *Bright* seemed plausible with Tok2Vec and GloVe boosting the high-frequency range.

In test fold 4, we saw reasonable overlap for *sharp*, *dark*, *hollow*, and *harsh*. We did not observe a reasonable pattern for *breezy* and *smooth*. Interestingly, the network with 'no embedding' predicted the EQ settings for *harsh* correctly. This is a chance occurrence because the 'no embedding' model predicted a standard template of settings for all the other words.

In Figure 6.6, we analyse the predictions on non-technical words. These non-technical words are the same as the HR words explained in section 6.2. Although these words may have a high consistency score in the SocialFX dataset, they may be highly subjective to the user. However, we compared the predictions of GloVe and Tok2Vec to the human labels. There was considerable overlap for *mellow*, *enchanting*, *rich*, and *romantic*. For *mournful* and *calm*, there were similar patterns between the predictions of the word embedding models and human labels. However, for *heat* and *brass*, the word embedding models did not predict a relevant pattern. Although the training set contained semantically similar words like *warm* and *brassy*, the embeddings did not perceive these similarities. This conveys that the algorithms to learn word embeddings can be further optimised for EQ mixing.

6.6 Discussion

The results presented in the previous section show that a word embedding layer is helpful for automatic mixing. We analysed the error of models in section 6.5.1. All the models with an embedding layer obtained lower errors than the one without an embedding layer. We further analysed the performance of GloVe-840B and Tok2Vec by using partial curve mapping. The mean distances obtained by human labels, Tok2Vec, GloVe, and 'no embedding' were 2.9, 10.5, 9.3, and 35.4 respectively. This objectively demonstrates that the embedding models perform better than models without an embedding layer, but definitely not as good as human labels.

In section 6.5.3, we conducted an error analysis of predictions made by GloVe and Tok2Vec. We observed that the machine learning model was able to understand semantic descriptors that it had not encountered before. This is a promising step towards understanding semantic descriptors from non-technical users. It is important to note the word embedding layers used in the networks were trained on corpora of written text. This concludes that there exists some common ground for semantic relationships between words in written text and for those adopted in EQ mixing.

Considering the fact that we have adopted such a small training dataset, this performance is reasonable. The SocialFX dataset comprises only 388 unique English words. Additionally, many of the high-quality and highly rated words were used for testing in each fold. As our study has demonstrated that word embeddings are helpful for automatic EQ mixing, we hope to encourage researchers the build larger datasets with semantic descriptors. In the literature, another dataset called SAFE (Stables et al., 2014) focused on extracting semantic descriptions for equalization from a DAW. We were unable to include the dataset within this study for two reasons. Firstly, as these are extracted directly from the DAW without post-processing, some labels can be noisy. Although the dataset contains many

examples with meaningful descriptors, some words are randomly typed letters such as 'xy', which have no semantic meaning. Perhaps, this noise may not matter when training the network with large-scale data. The second reason is that both datasets use different EQ plugins. The SocialFX dataset uses a 40-band EQ, whereas the SAFE dataset uses a five-band EQ. We are not certain if additional noise would be induced in mapping one EQ format to the other.

In this study, we analysed the performance of the machine learning model using objective metrics. However, it is important to perform listening tests with human participants to obtain subjective evaluations of the system. We need to investigate if users are satisfied with the way the machine learning model understands their semantic descriptors. After aggregating a larger dataset for this task, this could be a potential future pathway.

In this chapter, we demonstrated the feasibility of adopting word embeddings for automatic EQ mixing. We showed that the word embedding layer is capable of providing relationships between semantic descriptors, which assists in predicting EQ parameters. Using this technique, the machine learning model can predict EQ settings for words it has not seen before. This is a step towards bridging the gap between artists explaining their creative goals and mixing engineers understanding them.

In this study, we looked at EQ parameters as a separate entity. This may not be ideal in some scenarios. For example, the EQ settings for a drum track may differ from a vocal track. In other words, the EQ settings for *"make the vocals sound brighter"* maybe different from *"make the drums sound brighter"*. Moreover, the number of EQ bands predicted was 40. This number is pretty large for a network that performs regression. Future research could explore how the neural network architecture can be optimised and regularised better. Furthermore, it may be interesting to augment the size of training sets by adopting well-known synonyms and antonyms in the mixing engineer's vocabulary.

For some words, Tok2Vec captured relationships, but GloVe did not and vice versa. For example, GloVe captured the meaning of *punchy* as shown in figure 6.4 and Tok2Vec captured the meaning of *warm* as shown in figure 6.5. This may be simply because of limited data in the training set. Otherwise, different embedding models may capture different aspects of semantic relationships. Therefore, an ensemble of different embedding models will improve performance in this case. Furthermore, in our study, we discarded non-English words for simplicity. Word embedding models such as ConceptNet (Speer & Lowry-Duda, 2017) use a knowledge graph to connect words from different languages. This may be an interesting avenue to explore.

6.7 Publications, Code, and Contributions

- Venkatesh, S., Moffat, D., & Miranda, E. R. (2022a). Word embeddings for automatic equalization in audio mixing. *Journal of the Audio Engineering Society*, *70*(9), 753–763. doi: 10.17743/jaes.2022.0047

  – This journal paper presents our approach on using word embeddings to represent semantic descriptors for automatic EQ mixing. It has been submitted to a journal and is under peer review. The code associated with this study is openly available on GitHub (https://github.com/satvik-venkatesh/word-eq/).

- Venkatesh, S., Moffat, D., & Miranda, E. (2019). Radiome: Artificially intelligent radio for people with dementia. In *Proceedings of DMRN+14: Digital Music Research Network One-Day Workshop, London, UK*

  – We attended the DMRN one-day workshop to present a poster on RadioMe. As the project was just at the initial stage, we presented an overview of the overarching aims of the project, envisioning how intelligent remixing would be performed. At this point we had developed a basic music-speech detector, explored source separation of speech and music, and implemented some

remixing approaches.

- Di Campli San Vito, P., Brewster, S., Venkatesh, S., Miranda, E., Kirke, A., Moffat, D., Banerjee, S., Street, A., Fachner, J., & Odell-Miller, H. (2022). Radiome: Supporting individuals with dementia in their own home... and beyond? In *CHI Conference on Human Factors in Computing Systems (CHI '22) Workshop 32, New Orleans, Louisiana, USA, 30 Apr 2022*. doi: 10.36399/gla.pubs.267520
  – This was a paper presented by Di Campli San Vito et al. (2022) at the CHI '22 workshop on *Designing Ecosystems for Complex Health Needs*. I was involved through the RadioMe project in developing machine learning for audio segmentation and intelligent remixing in the RadioMe system.

- Shakeri, G., Brewster, S., Venkatesh, S., Moffat, D., Kirke, A., Miranda, E., Banerjee, S., Street, A., Fachner, J., & Odell-Miller, H. (2021). Radiome: challenges during the development of a real time tool to support people with dementia. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 08–13, 2021, Yokohama, Japan*. doi: 10026.1/17584
  – This paper was presented by Shakeri et al. (2021) at the CHI '22 workshop on *Designing Interactions for the Ageing Populations – Addressing Global Challenges*. I was involved thorugh the RadioMe project in developing machine learning for audio segmentation and intelligent remixing in the RadioMe system.

Chapter 7

Conclusion

In this thesis, we developed data-centric and model-centric approaches to improve state-of-the-art algorithms for audio segmentation. In chapter 3, we presented a method to synthesise large scale training sets to train deep learning models. Using this method, we obtained high accuracies on in-house and external test sets. On the MIREX dataset, there was a relative improvement of 56.55% and 1.15% for Music and Speech F-measures respectively. In chapter 4, this thesis proposed a novel algorithm called the You Only Hear Once (YOHO). It converted audio segmentation into a regression problem and generalised better than the CRNN across multiple datasets for audio segmentation and sound event detection. In addition, the speed of inference and post-processing were significantly faster. In chapter 5, we explored how the machine learning algorithm can be more robust shifts in data distributions. Using domain generalisation techniques, we improved the performance of the algorithm from 90.20% to 92.05% on the MIREX competition dataset.

In chapter 6, we demonstrated how audio segmentation can be performed on a live radio broadcast to intelligently remix the stream. Furthermore, we presented an approach for individuals to communicate with intelligent mixing systems through non-technical language. The following section summarises the

conclusions associated with each research question.

7.1 Research Conclusions

RQ1: *What would be an effective way to train machine learning models for audio segmentation with limited access to domain data?*

In chapter 3, we proposed a novel method to artificially synthesise training sets for music-speech detection. We replicated the pipeline of a mixing engineer by incorporating principles like audio ducking and fade curves. As broadcast audio is copyrighted material and cannot be shared, this method presented a promising alternative to spending resources on annotating real-world training sets.

Contribution 1.1 — *Artificially synthesising training sets is highly effective to generate large datasets to train deep neural networks for audio segmentation.*

In section 3.3, we investigated the robustness of the data synthesis procedure. After training a CRNN model on synthetic data, we tested it on two test sets — the in-house test set and the MIREX music-speech detection dataset. Although the training set and the test tests belong to different data distributions, it obtained high F-measures on both test sets. On the in-house test set, it obtained an overall F-measure of 96.69%. On the MIREX competition dataset, it obtained state-of-the-art performance for music and speech F-measure of 85.76% and 92.21% respectively.

Contribution 1.2 — *Synthetic training sets perform as good or better than real-world data for music-speech detection.*

In section 3.6, we compared the performance of artificial training sets and real-world training sets. In table 3.5, we observed that the model trained on artificial data obtained an overall F-measure of 96.89% on the in-house test set. Whereas, the model trained on real-world data obtained an overall F-measure of 96.64%.

Considering that the real-world training set and the in-house test set come from the same data distribution, artificial data is interestingly more effective. This difference is further exaggerated when we test the model on a different data distribution, which is the MIREX dataset. However, combining artificial and real-world data obtained the best performance.

Contribution 1.3 — *The optimal parameters for audio ducking selected by the machine learning algorithm were similar to the preferences of human listeners.*
In section 3.4, we investigated how different levels of audio ducking impact the performance of the machine learning algorithm. We found that the maximum Loudness Difference (LD) should lie between 23 and 27 LU for optimal performance in F-measure. We also observed that it is meaningless to have an LD greater than 40 LU because the recall stopped improving beyond this point. Furthermore, the precision of music kept decreasing as the LD kept increasing. This is because it becomes harder to precisely detect music at quieter volumes.

The minimum LD should lie between 2 and 8 LU for optimal F-measure performance for speech. Interestingly, we found that a minimum LD of 10 LU is necessary to maximise the precision of speech. In the study by Torcoli et al. (2019), human listeners also preferred a minimum LD of 10 LU. This similarity between the preferences of machines and humans is an interesting phenomenon.

Contribution 1.4 — *Synthetic data is more robust to disagreements between annotators of different datasets.*
On the in-house test set, the model trained on real-world data obtained a music F-measure of 97.81%. Whereas, the model trained on artificial data obtained a music F-measure of 97.77%. Furthermore, the model trained on both real and artificial data obtained a music F-measure of 98.3%. However, we did not observe the same pattern on the MIREX dataset. The models trained on real, artificial, and combined obtained 84.3%, 85.51%, and 85.01% respectively. In other words, artificial data

obtained the highest music F-measure on external data distributions. Moreover, real-world data actually hindered the F-measure of music, indicating that synthetic data is more robust to human-error and bias induced by the annotators.

RQ2: *How can we advance state-of-the-art algorithms by investigating novel pattern recognition problems?*

This research question was investigated in chapters 4 and 5, where we presented model-centric approaches to improve the performance of audio segmentation algorithms. First, we compared state-of-the-art architectures in the literature, then proposed a novel algorithm called YOHO, and later made it generalise better to unseen domains.

Contribution 2.1 — *Amongst the neural network architectures in the literature for audio segmentation, CRNN was the best performing network.*

In section 4.1, we compared CNN, GRU, LSTM, LSTM, TCN, and CRNN architectures for audio segmentation. We adopted an automatic hyperparameter tuning method called Hyperband (Li et al., 2017) to present an unbiased comparison of the architectures. We found that CRNN was the best performing model for music-speech detection. As CRNN combines the advantages of both convolutional and recurrent layers, it is well-suited for this task. Furthermore, the CRNN performs 2D convolutions, which is on the temporal and frequency dimensions, which makes it learn better high-level features.

Contribution 2.2 — *Machine learning models trained directly on raw audio did not perform as well as those using mel spectrograms.*

In section 4.2, we explored how end-to-end deep learning can be adopted for raw audio. First, we investigated a CRNN model for the task, which performed considerably poorer than using mel spectrograms. Later, we adopted the Wave-U-net architecture, which showed improvements from the CRNN model. However,

the best performing Wave-U-net model obtained 94.81%, which was lower than the CRNN model obtaining 97.39% on the validation set. These conclusions showed that using raw audio actually hindered the performance of audio segmentation. In future research, collecting more training data may be a potential pathway to reduce overfitting. However, as our training set was already diverse and large (46 h of synthetic data and 10.5 h of real-world data), we did not see the benefits of collecting more data.

Contribution 2.3 — *Proposed a novel algorithm called YOHO that generalises better than the CRNN architecture and is significantly faster than frame-based classification*
In section 4.3, we proposed a novel paradigm called You Only Hear Once (YOHO) for audio segmentation and sound event detection. YOHO presents sound event detection differently from the traditional segmentation-by-classification approach. It converted sound event detection from a frame-based classification problem to a regression problem. It obtained state-of-the-art performance for music-speech detection and surpassed the CRNN and CNN's performance for environmental audio. As this approach is more end-to-end and predicts acoustic boundaries directly, it is significantly quicker during post-processing and smoothing, which makes it more suitable for real-time applications.

Contribution 2.4 — *Demonstrated the benefits of transfer learning and domain-adversarial training for improved domain generalisation*
In chapter 5, we investigated how our model can generalise better to unseen domains. It is desirable that the music-speech detector performs well in audio from different broadcasters, languages, countries, and so on. Therefore, we demonstrated how transfer learning can be applied on YamNet (Plakal & Ellis, 2020) to improve domain generalisation. Furthermore, we investigate domain-adversarial training where a domain classifier predicts if the audio is real or synthetic. Using this approach, we obtained state-of-the-art performance on the MIREX music-

speech detection dataset, which is an *unseen domain*. YOHO with pre-training and domain-adversarial training obtained an overall F-measure of 92.05%, compared to simply YOHO which obtained 90.20%. Furthermore, we observed that domain-adversarial training did not help much in the in-house test set because it is a *seen domain*.

RQ3: *To what extent can deep learning be used to improve intelligent mixing approaches?* This research question was addressed in chapter 6. Within this research question, we primarily investigated two aspects — (1) *To what extent can audio segmentation improve existing real-time radio remixing approaches?* (2) *How can non-experts communicate with intelligent mixing systems?*

Contribution 3.1 — *Demonstrated how radio remixing can be performed with underlying segmentation*

In section 6.2, we presented a real-time radio remixing framework that assists people with dementia. There were two features for the remixing — (1) diary reminders and (2) music from the playlist. The remixing was governed by the music-speech detector, which was running in real-time in the background. This remixing aims to be seamlessly integrated with the radio stream. If music is detected in the stream, the background music is ducked and a diary reminder is played over it. Moreover, if speech is detected, we wait for a pause to occur in the speech, so that the diary reminder can be inserted. Initial tests have been conducted in homes of people with dementia through the RadioMe project. However, in the future, minor adjustments to the system can be easily incorporated based on their feedback.

Contribution 3.2 — *Demonstrated the benefits of using word embeddings to represent semantic descriptors in automatic mixing systems*

In sections 6.3 to 6.6, we investigated how individuals can communicate with

automatic mixing systems with non-technical language. We demonstrated that a word embedding layer enables the mixing system to understand high-level representations of semantic descriptors. It was also shown the machine learning model was able to predict mixing settings for words that it has not seen before. We presented this system to translate words to EQ parameters. This is a proof-of-concept to perform other mixing operations such as audio ducking and dynamic range compression.

7.2 Future Work

The conclusions in the previous section indicate many avenues that can be explored through future research. Below are some potential pathways.

Pathway 1: *Novel methods to generate artificial training data and data augmentation techniques*

There is a growing interest in adopting GANs for generating artificial training data (Goodfellow et al., 2014; Kong et al., 2020a). It would be interesting to explore how GANs can generate artificial training sets for audio segmentation. However, is important to generate label-preserving examples. As the temporal context is crucial for audio segmentation, the labels should not lose relevant information while generating new examples. Furthermore, there is a considerable difference in audio quality between different radio stations. For example, the audio signals in some stations have higher dynamic range compression or there are some frequency bands that are filtered. Data augmentation techniques that address these differences could improve the performance of music-speech detection algorithms.

In this thesis, we observed that it becomes increasingly harder to detect music at very low volumes. Furthermore, the OpenBMAT dataset found lower agreement scores between different annotators for very low volumes of back-

ground music (Meléndez-Catalán et al., 2019). This emphasises that the task of background music-detection needs to be defined in greater detail by using threshold-related variables. It would be interesting to explore the similarities between the OpenBMAT ontology for relative music loudness estimation and artificially synthesised data.

Pathway 2: *Improvements to audio segmentation as a regression problem*
For the YOHO algorithm in section 4.3, we primarily adapted the MobileNet architecture (Howard et al., 2017). Future developments in the network architecture for YOHO would lead to improvements in performance. For instance, adding skip connections through ResNets (He et al., 2016) or by including Inception blocks (Szegedy et al., 2015). Furthermore, there is scope to create hybrid architectures such as CNN-transformers (Kong et al., 2020b) by adopting the YOHO paradigm.

Although YOHO's output is more end-to-end by predicting acoustic boundaries directly, it is limited by the time-resolution of the input, which is the mel spectrogram. It would be interesting to explore YOHO with raw audio, which would make the sound event detection pipeline completely end-to-end. Moreover, the YOHO approach is relevant to related tasks such as singing voice detection. Furthermore, recent studies have successfully combined sound event detection with source separation and semi-supervised learning (Turpault et al., 2019, 2021). Future work could explore how YOHO would perform in these scenarios.

Pathway 3: *Investigate semi-supervised and unsupervised learning for audio segmentation*
Most of this thesis formulated audio segmentation as a supervised learning problem. When we explored concepts like domain generalisation, we still adopted labelled examples of real-world and synthetic examples. However, semi-supervised learning is gaining attention in the audio community. For instance, the mean-teacher approach (Tarvainen & Valpola, 2017) has been explored by researchers

for sound event detection (Yan et al., 2020; Lin et al., 2020). Furthermore, unsupervised domain adaptation (Ganin et al., 2016) can also be investigated where we have a lot of unlabelled real-world data.

Another interesting problem that can be investigated on radio signals is increasing the number of audio classes from just music and speech to higher-level structures like travel news, weather report, interviews, musical structures (chorus, verse, etc), genre recognition, and so on. This can be performed by clustering approaches like k-nearest neighbours. For example, we can use the embeddings from the music-speech detector to cluster radio signals and evaluate patterns.

Pathway 4: *Explore other domain generalisation techniques*
In chapter 5, we explored transfer learning and domain-adversarial training to improve music-speech detection. Although there are many well-established domain adaptation techniques for Computer Vision, they do not seem to work as well for audio (Lopez et al., 2021). This emphasises that there need to be more audio-specific methods developed for domain adaptation and generalisation. Some potential avenues include meta-learning, few-shot domain adaptation, and auxiliary task learning.

Pathway 5: *Curate datasets to facilitate research in using semantic descriptors for automatic mixing systems*
In chapter 6, we showed that a word embedding layer helps a machine learning model understand the representation of semantic descriptors. However, this word embedding layer was trained purely on natural language. Subsequently, the total number of unique descriptors in the Social EQ dataset was only 388 (only English was considered in this study). Hence, the performance of the machine learning model could be significantly improved if we adopt a large enough dataset for this task.

Acronyms

AI Artificial Intelligence. 42, 126

ANOVA Analysis of variance. 66–70, 72, 73

B-GRU Bidirectional Gated Recurrent Unit. 28, 82–87, 101, 107, 108, 111

B-LSTM Bidirectional Long Short-Term Memory. 25–27, 82–87

BIC Bayesian information criterion. 16, 24, 28

BN Batch Normalisation. 36, 65, 77, 83

CCSA Classification and Contrastive Semantic Alignment. 123

CLDNN Convolutional Long Short-Term Memory Fully Connected Deep Neural Network. 23, 88

CNN Convolutional Neural Network. 18–22, 25–28, 36, 42, 81–83, 85–87, 95, 100, 101, 105–111, 156

CRNN Convolutional Recurrent Neural Network. xi, xiii, xiv, 22, 23, 26, 28, 36, 58, 59, 81, 82, 84–88, 91, 93, 97, 99–101, 105–112, 153, 154, 156, 157

DCASE Detection and Classification of Acoustic Scenes and Events. 5, 19, 28, 93, 103, 124

EPSRC Engineering and Physical Sciences Research Council. iv

GANs Generative Adversarial Networks. 79, 159

GPU Graphical Processing Unit. 37

GRL Gradient Reversal Layer. 116, 118

GRU Gated Recurrent Unit. 21, 36, 81, 83, 85, 86, 156

HMM Hidden Markov Model. 18, 24, 27

ICCMR Interdisciplinary Centre for Computer Music Research. iv

LD Loudness Difference. xi, xii, 51–53, 62, 63, 65–73, 78, 155

LN Layer Normalisation. 65, 77, 83

LSTM Long Short-Term Memory. 23, 36, 81, 83, 88, 156

LU Loudness Units. xii, 51, 52, 65–73, 78, 155

LUFS Loudness Units relative to Full Scale. 52

MERL Mitsubishi Electric Research Laboratories. iii, 113

MFCC Mel Frequency Cepstral Coefficients. 12, 24, 87

MIR Music Information Retrieval. 3, 15, 127

MIREX Music Information Retrieval Evaluation eXchange. xiv, 5, 7, 14, 25, 26, 63, 93, 103, 106, 111, 120–122, 153

MLP Multilayer Perceptrons. 27

ncTCN non-causal Temporal Convolutional Network. 22, 25–27, 84, 86, 87

ResNet Residual Networks. 25, 27

RNN Recurrent Neural Network. xi, 21, 22, 33

SGD Stochastic Gradient Descent. 36, 141

STFT Short-time Fourier transform. 14

SVM Support Vector Machine. 18, 24, 28

TCN Temporal Convolutional Network. 18, 22, 36, 81, 82, 84–86, 156

YOHO You Only Hear Once. i, xii, xiv, 6, 9, 81, 82, 92–101, 104–112, 117–119, 153, 156–158

YOLO You Only Look Once. i, 6, 92, 93, 95

Bibliography

Adavanne, S., Fayek, H., & Tourbabin, V. (2019). Sound event classification and detection with weakly labeled data. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), New York, NY, USA*.

Adavanne, S., & Virtanen, T. (2017). A report on sound event detection with different binaural features. Tech. rep., Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge.

Algabri, M., Mathkour, H., Bencherif, M. A., Alsulaiman, M., & Mekhtiche, M. A. (2020). Towards deep object detection techniques for phoneme recognition. *IEEE Access*, *8*, 54663–54680.

Antoine, A., Williams, D., & Miranda, E. R. (2016). Towards a timbral classification system for musical excerpts. In *Proceedings of the 2nd AES Workshop on Intelligent Music Production, London, UK*, vol. 13.

Armstrong, M., Brooks, M., Churnside, A., Evans, M., Melchior, F., & Shotton, M. (2014). Object-based broadcasting-curation, responsiveness and user experience. In *International Broadcasting Convention, Amsterdam, Netherlands*. IET.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Bakarov, A. (2018). A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536*.

Baktashmotlagh, M., Harandi, M. T., Lovell, B. C., & Salzmann, M. (2013). Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia*, (pp. 769–776).

Bartsch, M. A., & Wakefield, G. H. (2001). To catch a chorus: Using chroma-based representations for audio thumbnailing. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (WASPAA), NY, USA*, (pp. 15–18).

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(2).

Bhattacharjee, M., Prasanna, S. M., & Guha, P. (2022). Clean vs. overlapped speech-music detection using harmonic-percussive features and multi-task learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. doi: 10.1109/TASLP.2022.3164199.

Biesialska, M., Biesialska, K., & Costa-jussà, M. R. (2020). Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain*, (pp. 6523–6541).

Bittner, R. M., Gu, M., Hernandez, G., Humphrey, E. J., Jehan, T., McCurry, H., & Montecchio, N. (2017). Automatic playlist sequencing and transitions. In *18th International Society for Music Information Retrieval Conference (ISMIR), Suzhou, China*, (pp. 442–448).

Black, D. A. A., Li, M., & Tian, M. (2014). Automatic identification of emotional cues in chinese opera singing. In *13th International Conference on Music Perception and Cognition and the 5th Conference for the Asian-Pacific Society for Cognitive Sciences of Music, Seoul, South Korea*.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, New York, NY, USA*, (pp. 92–100).

Bosch, J. J., Janer, J., Fuhrmann, F., & Herrera, P. (2012). A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *13th International Society for Music Information Retrieval Conference (ISMIR), Porto, Portugal*, (pp. 559–564).

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Bromham, G., Moffat, D., Barthet, M., Danielsen, A., & Fazekas, G. (2019). The impact of audio effects processing on the perception of brightness and warmth. In *ACM Audio Mostly Conference, Nottingham, UK*.

Butko, T., & Nadeu, C. (2011). Audio segmentation of broadcast news in the albayzin-2010 evaluation: overview, results, and discussion. *EURASIP Journal on Audio, Speech, and Music Processing*, *2011*(1), 1.

Cakir, E., Heittola, T., Huttunen, H., & Virtanen, T. (2015). Multi-label vs. combined single-label sound event detection with deep neural networks. In *23rd European Signal Processing Conference (EUSIPCO), Nice, France*, (pp. 2551–2555). IEEE.

Cakır, 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.

Cartwright, M. B., & Pardo, B. (2013). Social-EQ: Crowdsourcing an equalization descriptor map. In *14th International Society for Music Information Retrieval Conference (ISMIR), Curitiba, Brazil*, (pp. 395–400).

Castán, D., Ortega, A., Miguel, A., & Lleida, E. (2014). Audio segmentation-by-classification approach based on factor analysis in broadcast news domain. *EURASIP Journal on Audio, Speech, and Music Processing*, *2014*(1), 34.

Chen, S., Gopalakrishnan, P., et al. (1998). Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, Virginia*, vol. 8, (pp. 127–132). Citeseer.

Chen, Y., Dinkel, H., Wu, M., & Yu, K. (2020). Voice activity detection in the wild via weakly supervised sound event detection. In *Interspeech, Shanghai, China*, (pp. 3665–3669).

Cheuk, K. W., Agres, K., & Herremans, D. (2020). The impact of audio input representations on neural network based music transcription. In *2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK*, (pp. 1–6). IEEE.

Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing, Doha, Qatar*.

Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2017a). A tutorial on deep learning for music information retrieval. *arXiv preprint arXiv:1709.04396*.

Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017b). Convolutional recurrent neural networks for music classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, Louisiana, USA*, (pp. 2392–2396).

Choi, M., Lee, J., & Nam, J. (2018). Hybrid features for music and speech detection. Tech. rep., Music Information Retrieval Evaluation eXchange (MIREX) Challenge.

Chourdakis, E., Ward, L., Paradis, M., & Reiss, J. D. (2019). Modelling experts' decisions on assigning narrative importances of objects in a radio drama mix. In *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx), Birmingham, UK, September 2–6, 2019*.

Chourdakis, E. T., & Reiss, J. D. (2017). A machine-learning approach to application of intelligent artificial reverberation. *Journal of the Audio Engineering Society*, *65*(1/2), 56–65.

Chourdakis, E. T., & Reiss, J. D. (2019). Tagging and retrieval of room impulse responses using semantic word vectors and perceptual measures of reverberation. In *146th Convention of the Audio Engineering Society, Dublin, Ireland*.

Cornell, S., Olvera, M., Pariente, M., Pepe, G., Principi, E., Gabrielli, L., & Squartini, S. (2020). Domain-adversarial training and trainable parallel front-end for the dcase 2020 task 4 sound event detection challenge. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), Tokyo, Japan*.

Coşkun, M., YILDIRIM, Ö., Ayşegül, U., & Demir, Y. (2017). An overview of popular deep learning methods. *European Journal of Technique (EJT)*, *7*(2), 165–176.

De Man, B., & Reiss, J. D. (2013). A knowledge-engineered autonomous mixing system. In *135th Convention of the Audio Engineering Society, New York, NY, USA*.

De Man, B., Stables, R., & Reiss, J. D. (2019). *Intelligent Music Production*. Focal Press.

de Sa, V. R. (1994). Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems*, (pp. 112–119). Citeseer.

DeepLearning.AI (2021). A chat with Andrew on MLOps: From model-centric to data-centric AI. `https://youtu.be/06-AZXmwHjo` [Last accessed on 02-11-2021].

Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA*, (pp. 4690–4699).

Dhanalakshmi, P., Palanivel, S., & Ramalingam, V. (2009). Classification of audio signals using SVM and RBFNN. *Expert Systems with Applications*, *36*(3), 6069–6075.

Di Campli San Vito, P., Brewster, S., Venkatesh, S., Miranda, E., Kirke, A., Moffat, D., Banerjee, S., Street, A., Fachner, J., & Odell-Miller, H. (2022). Radiome: Supporting individuals with dementia in their own home... and beyond? In *CHI Conference on Human Factors in Computing Systems (CHI '22) Workshop 32, New Orleans, Louisiana, USA, 30 Apr 2022*. doi: 10.36399/gla.pubs.267520.

Diment, A., & Virtanen, T. (2017). Transfer learning of weakly labelled audio. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA*, (pp. 6–10).

Dinkel, H., Wu, M., & Yu, K. (2021). Towards duration robust weakly supervised sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *29*, 887–900.

Docio-Fernandez, L., Lopez-Otero, P., & Garcia-Mateo, C. (2010). The UVigo-GTM speaker diarization system for the albayzin'10 evaluation. In *Proceedings of FALA 2010: VI Jornadas en Tecnología del Habla and II Iberian SLTech Workshop, Vigo, Spain*.

Doukhan, D., & Carrive, J. (2017). Investigating the use of semi-supervised convolutional neural network models for speech/music classification and segmentation. In *The Ninth International Conferences on Advances in Multimedia (MMEDIA), Venice, Italy*.

Fenton, S. (2018). Automatic mixing of multitrack material using modified loudness models. In *145th Convention of the Audio Engineering Society, New York, NY, USA*. Audio Engineering Society.

Forman, G., & Scholz, M. (2010). Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *ACM SIGKKD Explorations Newsletter*, *12*(1), 49–57.

Gal, Y., & Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. *Advances in Neural Information Processing Systems*, *29*, 1019–1027.

Gallardo Antolín, A., & San Segundo Hernández, R. (2010). UPM-UC3M system for music and speech segmentation. In *Proceedings of FALA 2010: VI Jornadas en Tecnología del Habla and II Iberian SLTech Workshop, Vigo, Spain*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, *17*(1), 2096–2030.

Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., & Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, Louisiana, USA*, (pp. 776–780).

Gharib, S., Drossos, K., Fagerlund, E., & Virtanen, T. (2019). Voice: A sound event detection dataset for generalizable domain adaptation. *arXiv preprint arXiv:1911.07098*.

Gimeno, P., Viñals, I., Ortega, A., Miguel, A., & Lleida, E. (2020). Multiclass audio segmentation based on recurrent neural networks for broadcast domain data. *EURASIP Journal on Audio, Speech, and Music Processing*, *2020*(1), 1–19.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy*, (pp. 249–256). PMLR.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, *10*(1), 1–309.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, (pp. 2672–2680).

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, Canada*, (pp. 6645–6649).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, Nevada, USA*, (pp. 770–778).

He, W., Motlicek, P., & Odobez, J.-M. (2019). Adaptation of multiple sound source localization neural networks with weak supervision and domain-adversarial training. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK*, (pp. 770–774).

Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., et al. (2017). CNN architectures for large-scale audio classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, Louisiana, USA*, (pp. 131–135).

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

Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). spaCy: Industrial-strength natural language processing in python. doi: 10.5281/zenodo.1212303.

Hoshen, Y., Weiss, R. J., & Wilson, K. W. (2015). Speech acoustic modeling from raw multichannel waveforms. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia*, (pp. 4624–4628).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Huang, J., Dong, Y., Liu, J., Dong, C., & Wang, H. (2009). Sports audio segmentation and classification. In *IEEE International Conference on Network Infrastructure and Digital Content*, (pp. 379–383).

Huang, R., & Hansen, J. H. (2006). Advances in unsupervised audio classification and segmentation for the broadcast news and NGSW corpora. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(3), 907–919.

Humphrey, E. J., Salamon, J., Nieto, O., Forsyth, J., Bittner, R. M., & Bello, J. P. (2014). JAMS: A JSON annotated music specification for reproducible MIR research. In *15th International Society for Music Information Retrieval Conference (ISMIR), Taipei, Taiwan*, (pp. 591–596).

Hussain, M., Haque, M. A., et al. (2018). Swishnet: a fast convolutional neural network for speech, music and noise classification and segmentation. *arXiv preprint arXiv:1812.00149*.

IBM Corporation (2017). SPSS statistics for windows. Version 25.0.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICLR), San Diego, CA, USA*, (pp. 448–456).

ITU-R (2017). ITU-R Rec. BS.1770-4: Algorithms to measure audio programme loudness and true-peak audio level. *BS Series*.

Jang, B.-Y., Heo, W.-H., Kim, J.-H., & Kwon, O.-W. (2019). Music detection from broadcast contents using convolutional neural networks with a mel-scale kernel. *EURASIP Journal on Audio, Speech, and Music Processing*, *2019*(1), 11.

Jekel, C. F., Venter, G., Venter, M. P., Stander, N., & Haftka, R. T. (2019). Similarity measures for identifying material parameters from hysteresis loops using inverse analysis. *International Journal of Material Forming*.

Jeong, I.-Y., Lee, S., Han, Y., & Lee, K. (2017). Audio event detection using multiple-input convolutional neural network. Tech. rep., Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge.

Jiang, N., Grosche, P., Konz, V., & Müller, M. (2011). Analyzing chroma feature types for automated chord recognition. In *Audio Engineering Society Conference: 42nd International Conference: Semantic Audio, Ilmenau, Germany*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA*.

Koch, G., Zemel, R., Salakhutdinov, R., et al. (2015). Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, vol. 2, (p. 0). Lille.

Kong, J., Kim, J., & Bae, J. (2020a). Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, *33*, 17022–17033.

Kong, Q., Xu, Y., Sobieraj, I., Wang, W., & Plumbley, M. D. (2019). Sound event detection and time–frequency segmentation from weakly labelled data. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *27*(4), 777–787.

Kong, Q., Xu, Y., Wang, W., & Plumbley, M. D. (2020b). Sound event detection of weakly labelled data with CNN-transformer and automatic threshold optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 2450–2460.

Kos, M., Grasic, M., Vlaj, D., & Kacic, Z. (2009). On-line speech/music segmentation for broadcast news domain. In *16th International Conference on Systems, Signals and Image Processing, Chalkida, Greece*, (pp. 1–4). IEEE.

Kotti, M., Benetos, E., & Kotropoulos, C. (2008). Computationally efficient and robust bic-based speaker segmentation. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(5), 920–933.

Kukačka, J., Golkov, V., & Cremers, D. (2017). Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*.

Kulis, B., Saenko, K., & Darrell, T. (2011). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 1785–1792).

Kulka, L. d. G. (1972). Equalization-the highest, most sustained expression of the recordist's heart. *Recording Engineer/Producer*, *3*(6), 17–24.

Kum, S., & Nam, J. (2019). Joint detection and classification of singing voice melody using convolutional recurrent neural networks. *Applied Sciences*, *9*(7), 1324.

Kumar, A., & Raj, B. (2016). Audio event detection using weakly labeled data. In *Proceedings of the 24th ACM International Conference on Multimedia, Amsterdam, Netherlands*, (pp. 1038–1047).

Kwon, S., et al. (2020). A CNN-assisted enhanced audio signal processing for speech emotion recognition. *Sensors*, *20*(1), 183.

Lee, J., Park, J., Kim, T., & Nam, J. (2017). Raw waveform-based audio classification using sample-level CNN architectures. In *Machine Learning for Audio Signal Processing Workshop, Neural Information Processing Systems (NeurIPS)*.

Lee, Y., Lim, S., & Kwak, I.-Y. (2021). CNN-based acoustic scene classification system. *Electronics*, *10*(4), 371.

Lemaire, Q., & Holzapfel, A. (2019). Temporal convolutional networks for speech and music detection in radio broadcast. In *20th International Society for Music Information Retrieval Conference (ISMIR)*.

Li, D., Yang, Y., Song, Y.-Z., & Hospedales, T. (2018). Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA*, vol. 32.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, *18*(1), 6765–6816.

Lidy, T. (2015). Spectral convolutional neural network for music classification. Tech. rep., Music Information Retrieval Evaluation eXchange (MIREX) Challenge.

Lin, L., Wang, X., Liu, H., & Qian, Y. (2020). Guided learning for weakly-labeled semi-supervised sound event detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain*, (pp. 626–630).

Liu, C., Jin, Z., Gu, J., & Qiu, C. (2017). Short-term load forecasting using a long short-term memory network. In *IEEE PES innovative smart grid technologies conference Europe (ISGT-Europe)*, (pp. 1–6).

Lo, H.-Y., Wang, J.-C., & Wang, H.-M. (2010). Homogeneous segmentation and classifier ensemble for audio tag annotation and retrieval. In *IEEE International Conference on Multimedia and Expo, Singapore*, (pp. 304–309).

Long, M., Ding, G., Wang, J., Sun, J., Guo, Y., & Yu, P. S. (2013). Transfer sparse coding for robust image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 407–414).

Lopez, J. A., Stemmer, G., Lopez Meyer, P., Singh, P., Del Hoyo Ontiveros, J., & Cordourier, H. (2021). Ensemble of complementary anomaly detectors under domain shifted conditions. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), Barcelona, Spain*, (pp. 11–15).

Lu, R., & Duan, Z. (2017). Bidirectional GRU for sound event detection. Tech. rep., Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge.

Luo, L., Zhang, L., Wang, M., Liu, Z., Liu, X., He, R., & Jin, Y. (2021). A system for the detection of polyphonic sound on a university campus based on CapsNet-RNN. *IEEE Access*, *9*, 147900–147913.

Marolt, M. (2015). Music/speech classification and detection submission for mirex 2015. Tech. rep., Music Information Retrieval Evaluation eXchange (MIREX) Challenge.

Marolt, M. (2018). Music/speech classification and detection submission for MIREX 2018. Tech. rep., Music Information Retrieval Evaluation eXchange (MIREX) Challenge.

Martín-Morató, I., Mesaros, A., Heittola, T., Virtanen, T., Cobos, M., & Ferri, F. J. (2019). Sound event envelope estimation in polyphonic mixtures. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK*, (pp. 935–939).

Masuyama, Y., Yatabe, K., Koizumi, Y., Oikawa, Y., & Harada, N. (2020). Phase reconstruction based on recurrent phase unwrapping with deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain*, (pp. 826–830).

Matsuura, T., & Harada, T. (2020). Domain generalization using a mixture of multiple latent domains. In *Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA*, vol. 34, (pp. 11749–11756).

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference, Austin, Texas, USA*, vol. 8, (pp. 18–25). Citeseer.

Meléndez-Catalán, B., Molina, E., & Gomez, E. (2018). Music and/or speech detection MIREX 2018 submission. Tech. rep., Music Information Retrieval Evaluation eXchange (MIREX) Challenge.

Meléndez-Catalán, B., Molina, E., & Gómez, E. (2019). Open broadcast media audio from TV: A dataset of TV broadcast audio with relative music loudness annotations. *Transactions of the International Society for Music Information Retrieval*, *2*(1).

Meléndez-Catalán, B., Molina, E., & Gómez Gutiérrez, E. (2017). BAT: An open-source, web-based audio events annotation tool. In *Web Audio Conference, London, UK*.

Meléndez-Catalán, B., SL, B. L., Molina, E., & Gómez, E. (2020). Relative music loudness estimation using temporal convolutional networks and a CNN feature extraction front-end. In *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx), Vienna, Austria*, vol. 5, (pp. 273–280).

Mesaros, A., Heittola, T., Diment, A., Elizalde, B., Shah, A., Vincent, E., Raj, B., & Virtanen, T. (2017). DCASE 2017 challenge setup: Tasks, datasets and baseline system. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), Munich, Germany*.

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

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Miranda, E. R. (1995). An artificial intelligence approach to sound design. *Computer Music Journal*, *19*(2), 59–75.

Miyazaki, K., Komatsu, T., Hayashi, T., Watanabe, S., Toda, T., & Takeda, K. (2020a). Convolution-augmented transformer for semisupervised sound event detection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), Tokyo, Japan*, (pp. 100–104).

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

Moffat, D., & Sandler, M. (2019a). Machine learning multitrack gain mixing of drums. In *147th Convention of the Audio Engineering Society, New York, NY, USA*.

Moffat, D., & Sandler, M. B. (2019b). Approaches in intelligent music production. In *Arts*, vol. 8, (p. 125). Multidisciplinary Digital Publishing Institute.

Moffat, D., Thalmann, F., & Sandler, M. B. (2018). Towards a semantic web representation and application of audio mixing rules. In *Proceedings of the 4th Workshop on Intelligent Music Production, Huddersfield, UK*.

Motiian, S., Jones, Q., Iranmanesh, S., & Doretto, G. (2017a). Few-shot adversarial domain adaptation. *Advances in Neural Information Processing Systems*, *30*.

Motiian, S., Piccirilli, M., Adjeroh, D. A., & Doretto, G. (2017b). Unified deep supervised domain adaptation and generalization. In *IEEE International Conference on Computer Vision, Venice, Italy*, (pp. 5715–5725).

MuSpeak Team (2015). MIREX muspeak sample dataset. `http://mirg.city.ac.uk/datasets/muspeak/` [Last accessed on 26-02-2021].

Ng, A. Y., et al. (1997). Preventing "overfitting" of cross-validation data. In *International Conference on Machine Learning (ICML)*, vol. 97, (pp. 245–253).

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). LibriSpeech: an ASR corpus based on public domain audio books. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia*, (pp. 5206–5210).

Papadopoulos, H., & Peeters, G. (2007). Large-scale study of chord estimation algorithms based on chroma representation and hmm. In *International Workshop on Content-Based Multimedia Indexing, Bordeaux, France*, (pp. 53–60). IEEE.

Papakostas, M., & Giannakopoulos, T. (2018). Speech-music discrimination using deep visual feature extractors. *Expert Systems with Applications*, *114*, 334–344.

Parascandolo, G., Huttunen, H., & Virtanen, T. (2016). Recurrent neural networks for polyphonic sound event detection in real life recordings. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China*, (pp. 6440–6444).

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech, Graz, Austria*, (pp. 2613–2617).

Parvat, A., Chavan, J., Kadam, S., Dev, S., & Pathak, V. (2017). A survey of deep-learning frameworks. In *International Conference on Inventive Systems and Control (ICISC), Coimbatore, Tamil Nadu, India*, (pp. 1–7). IEEE.

Pearce, A., Brookes, T., & Mason, R. (2016). Audio Commons: Hierarchical ontology of timbral semantic descriptors. Tech. rep. `https://www.audiocommons.org/assets/files/AC-WP5-Surrey-D5.1%20Hierarchical%20ontology%20of%20timbral%20semantic%20descriptors.pdf` Accessed on 26-11-21.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing, Doha, Qatar*, (pp. 1532–1543).

Perez-Gonzalez, E., & Reiss, J. (2009). Automatic gain and fader control for live mixing. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA*, (pp. 1–4).

Pestana, P. D., & Barbosa, A. (2012). Accuracy of itu-r bs. 1770 algorithm in evaluating multitrack material. In *133rd Convention of the Audio Engineering Society, San Francisco, CA, USA*. Audio Engineering Society.

Phan, H., Koch, P., Katzberg, F., Maass, M., Mazur, R., & Mertins, A. (2017). Audio scene classification with deep recurrent neural networks. In *Interspeech, Stockholm, Sweden*, (pp. 3043–3047). International Speech Communication Association.

Phan, H., Maaß, M., Mazur, R., & Mertins, A. (2014). Random regression forests for acoustic event detection and classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(1), 20–31.

Piczak, K. J. (2015). ESC: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia, Brisbane Australia*, (pp. 1015–1018).

Plakal, M., & Ellis, D. (2020). Yamnet. `https://github.com/tensorflow/models/tree/master/research/audioset/yamnet/` [Last accessed on 08-10-2021].

Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2008). *Dataset shift in machine learning*. MIT Press.

Ramirez, M. M., Stoller, D., & Moffat, D. (2021). A deep learning approach to intelligent drum mixing with the wave-u-net. *Journal of the Audio Engineering Society*, *69*(3), 142–151.

Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(2), 92–102.

Ravelli, E., Richard, G., & Daudet, L. (2010). Audio signal representations for indexing in the transform domain. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(3), 434–446.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, Nevada, USA*, (pp. 779–788).

Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 7263–7271).

Saenko, K., Kulis, B., Fritz, M., & Darrell, T. (2010). Adapting visual category models to new domains. In *European Conference on Computer Vision, Heraklion, Crete, Greece*, (pp. 213–226). Springer.

Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015a). Convolutional, long short-term memory, fully connected deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia*, (pp. 4580–4584).

Sainath, T. N., Weiss, R. J., Senior, A. W., Wilson, K. W., & Vinyals, O. (2015b). Learning the speech front-end with raw waveform CLDNNs. In *Interspeech, Dresden, Germany*.

Sajjadi, M., Javanmardi, M., & Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in Neural Information Processing Systems*, *29*, 1163–1171.

Salamon, J., & Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, *24*(3), 279–283.

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

Samuel, D., Ganeshan, A., & Naradowsky, J. (2020). Meta-learning extractors for music source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain*, (pp. 816–820).

Scheirer, E., & Slaney, M. (1997). Construction and evaluation of a robust multifeature speech/music discriminator. In *IEEE International Conference on Acoustics, Speech, and Signal processing (ICASSP), NW Washington, DC, United States*, vol. 2, (pp. 1331–1334).

Schlüter, J., Doukhan, D., & Meléndez-Catalán, B. (2018). MIREX challenge: Music and/or speech detection. `https://www.music-ir.org/mirex/wiki/2018:Music_and/or_Speech_Detection` [Last accessed on 14-08-2021].

Schlüter, J., & Grill, T. (2015). Exploring data augmentation for improved singing voice detection with neural networks. In *16th International Society for Music Information Retrieval Conference (ISMIR), Málaga, Spain*, (pp. 121–126).

Schlüter, J., & Sonnleitner, R. (2012). Unsupervised feature learning for speech and music detection in radio broadcasts. In *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx), Trondheim, Norway*.

Seetharaman, P., & Pardo, B. (2014). Crowdsourcing a reverberation descriptor map. In *Proceedings of the 22nd ACM international conference on Multimedia, New York, NY, United States*, (pp. 587–596).

Segal, Y., Fuchs, T. S., & Keshet, J. (2019). SpeechYOLO: Detection and localization of speech objects. In *Interspeech, Graz, Austria*, (pp. 4210–4214).

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *2nd International Conference on Learning Representations (ICLR), Banff, Alberta, Canada*.

Seyerlehner, K., Pohle, T., Schedl, M., & Widmer, G. (2007). Automatic music detection in television productions. In *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx), Bordeaux, France*.

Shakeri, G., Brewster, S., Venkatesh, S., Moffat, D., Kirke, A., Miranda, E., Banerjee, S., Street, A., Fachner, J., & Odell-Miller, H. (2021). Radiome: challenges during the development of a real time tool to support people with dementia. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 08–13, 2021, Yokohama, Japan*. doi: 10026.1/17584.

Shi, B., Sun, M., Puvvada, K. C., Kao, C.-C., Matsoukas, S., & Wang, C. (2020). Few-shot acoustic event detection via meta learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain*, (pp. 76–80).

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 1–48.

Sicilia, A., Zhao, X., & Hwang, S. J. (2021). Domain adversarial neural networks for domain generalization: When it works and how to improve. *arXiv preprint arXiv:2102.03924*.

Sifre, L. (2014). Rigid-motion scattering for image classification. PhD thesis, Ecole Normale Superieure.

Snyder, D., Chen, G., & Povey, D. (2015). Musan: A music, speech, and noise corpus. *arXiv preprint arXiv:1510.08484*.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, *45*(4), 427–437.

Speer, R., & Lowry-Duda, J. (2017). Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Vancouver, Canada*, (pp. 85–89).

Spyridon, S. (2019). *Audio equalisation using natural language*. Ph.D. thesis, Birmingham City University.

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.

Stables, R., Enderby, S., De Man, B., Fazekas, G., & Reiss, J. D. (2014). SAFE: A system for extraction and retrieval of semantic audio descriptors. In *15th International Society for Music Information Retrieval (ISMIR) Conference, Taipei, Taiwan*.

Stasis, S., Stables, R., & Hockman, J. (2016). Semantically controlled adaptive equalisation in reduced dimensionality parameter space. *Applied Sciences*, *6*(4), 116.

Steinmetz, C. J., & Reiss, J. D. (2021). pyloudnorm: A simple yet flexible loudness meter in python. In *150th Convention of the Audio Engineering Society*.

Stoller, D., Ewert, S., & Dixon, S. (2018). Wave-U-Net: A multi-scale neural network for end-to-end audio source separation. In *19th International Society for Music Information Retrieval Conference (ISMIR), Paris, France*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA*, (pp. 1–9).

Tarr, E. (2018). *Hack Audio: An Introduction to Computer Programming and Digital Signal Processing in MATLAB*. Routledge.

Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA*, (pp. 1195–1204).

Thalmann, F., Thompson, L., & Sandler, M. (2018). A user-adaptive automated dj web app with object-based audio and crowd-sourced decision trees. In *Web Audio Conference, Berlin, Germany*.

Theodorou, T., Mporas, I., & Fakotakis, N. (2014). An overview of automatic audio segmentation. *International Journal of Information Technology and Computer Science (IJITCS)*, *6*(11), 1.

Thieling, L., Wilhelm, D., & Jax, P. (2021). Recurrent phase reconstruction using estimated phase derivatives from deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Ontario, Canada*, (pp. 7088–7092).

Tissier, J., Gravier, C., & Habrard, A. (2017). Dict2vec : Learning word embeddings using lexical dictionaries. In *Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark*, (pp. 254–263). Association for Computational Linguistics.

Tiwari, S., Lakhotia, K., & Mulimani, M. (2021). Evaluating robustness of you only hear once (YOHO) algorithm on noisy audios in the voice dataset. *arXiv preprint arXiv:2111.01205*.

Torcoli, M., Freke-Morin, A., Paulus, J., Simon, C., & Shirley, B. (2019). Background ducking to produce esthetically pleasing audio for TV with clear speech. In *146th Convention of the Audio Engineering Society, Dublin, Ireland*.

Torcoli, M., Paulus, J., Kastner, T., & Uhle, C. (2021). Controlling the remixing of separated dialogue with a non-intrusive quality estimate. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA*, (pp. 91–95).

Turpault, N., Serizel, R., Shah, A., & Salamon, J. (2019). Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), New York, NY, USA*, (pp. 253–257).

Turpault, N., Serizel, R., Wisdom, S., Erdogan, H., Hershey, J. R., Fonseca, E., Seetharaman, P., & Salamon, J. (2021). Sound event detection and separation: a benchmark on desed synthetic soundscapes. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Ontario, Canada*, (pp. 840–844).

Tzanetakis, G., & Cook, P. (2000). Marsyas: A framework for audio analysis. *Organised sound*, *4*(3), 169–175.

Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio processing*, *10*(5), 293–302.

UK Digital Production Partnership (DPP) (2017). Technical specification for the delivery of television programmesas as-11 files v5.0. *Sec. 2.2.1. Loudness Terms*.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*(2), 373–440.

Venkatesh, S. (2021). Data generators with keras and tensorflow on google colab. `https://satvikvenkatesh.medium.com/data-generators-with-keras-and-tensorflow-on-google-colab-65fbd60a941a` [Last accessed on 09-03-2022].

Venkatesh, S., Moffat, D., Kirke, A., Shakeri, G., Brewster, S., Fachner, J., Odell-Miller, H., Street, A., Farina, N., Banerjee, S., et al. (2021a). Artificially synthesising data for audio classification and segmentation to improve speech and music detection in radio broadcast. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Ontario, Canada*, (pp. 636–640). doi: 10.1109/ICASSP39728.2021.9413597.

Venkatesh, S., Moffat, D., & Miranda, E. (2019). Radiome: Artificially intelligent radio for people with dementia. In *Proceedings of DMRN+14: Digital Music Research Network One-Day Workshop, London, UK*.

Venkatesh, S., Moffat, D., & Miranda, E. R. (2021b). Investigating the effects of training set synthesis for audio segmentation of radio broadcast. *Electronics*, *10*(7), 827. doi: 10.3390/electronics10070827.

Venkatesh, S., Moffat, D., & Miranda, E. R. (2022a). Word embeddings for automatic equalization in audio mixing. *Journal of the Audio Engineering Society*, *70*(9), 753–763. doi: 10.17743/jaes.2022.0047.

Venkatesh, S., Moffat, D., & Miranda, E. R. (2022b). You only hear once: a YOLO-like algorithm for audio segmentation and sound event detection. *Applied Sciences*, *12*(7), 3293. doi: 10.3390/app12073293.

Venkatesh, S., Wichern, G., Subramanian, A., & Le Roux, J. (2022c). Disentangled surrogate task learning for improved domain generalization in unsupervised anomalous sound detection. Tech. rep., Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge.

Vesperini, F., Gabrielli, L., Principi, E., & Squartini, S. (2019). Polyphonic sound event detection by using capsule neural networks. *IEEE Journal of Selected Topics in Signal Processing*, *13*(2), 310–322.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial intelligence review*, *18*(2), 77–95.

Wang, D., Vogt, R., Mason, M., & Sridharan, S. (2008). Automatic audio segmentation using the generalized likelihood ratio. In *2nd International Conference on Signal Processing and Communication Systems, Gold Coast, Australia*, (pp. 1–5). IEEE.

Wang, Y., Salamon, J., Bryan, N. J., & Bello, J. P. (2020). Few-shot sound event detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain*, (pp. 81–85).

Wichern, G., Chakrabarty, A., Wang, Z.-Q., & Le Roux, J. (2021). Anomalous sound detection using attentive neural processes. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA*, (pp. 186–190).

Wichern, G., Wishnick, A., Lukin, A., & Robertson, H. (2015). Comparison of loudness features for automatic level adjustment in mixing. In *139th Convention of the Audio Engineering Society, New York, NY, USA*. Audio Engineering Society.

Williams, D., & Brookes, T. (2007). Perceptually-motivated audio morphing: Brightness. In *122nd Convention of the Audio Engineering Society, Vienna, Austria*.

Williams, D., & Brookes, T. (2009). Perceptually-motivated audio morphing: softness. In *126th Convention of the Audio Engineering Society, Munich, Germany*.

Witowski, K., & Stander, N. (2012). Parameter identification of hysteretic models using partial curve mapping. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Indianapolis, Indiana*, (p. 5580).

Wright, P. S. (1999). Short-time fourier transforms and wigner-ville distributions applied to the calibration of power frequency harmonic analyzers. *IEEE Transactions on Instrumentation and Measurement*, *48*(2), 475–478.

Wu, C.-H., Chiu, Y.-H., Shia, C.-J., & Lin, C.-Y. (2005). Automatic segmentation and identification of mixed-language speech using delta-BIC and LSA-based GMMs. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(1), 266–276.

Xu, Y., Du, J., Dai, L.-R., & Lee, C.-H. (2014). A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(1), 7–19.

Xue, H., Li, H., Gao, C., & Shi, Z. (2010). Computationally efficient audio segmentation through a multi-stage BIC approach. In *3rd International Congress on Image and Signal Processing, Yantai, China*, vol. 8, (pp. 3774–3777). IEEE.

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, *9*(4), 611–629.

Yan, J., Song, Y., Dai, L.-R., & McLoughlin, I. (2020). Task-aware mean teacher method for large scale weakly labeled semi-supervised sound event detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain*, (pp. 326–330).

Yang, J. H., Kim, N. K., & Kim, H. K. (2018). SE-ResNet with GAN-based data augmentation applied to acoustic scene classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), Surrey, UK*.

Yang, L., Hao, J., Hou, Z., & Peng, W. (2020). Two-stage domain adaptation for sound event detection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), Tokyo, Japan*, (pp. 230–234).

Yao, Y., Rosasco, L., & Caponnetto, A. (2007). On early stopping in gradient descent learning. *Constructive Approximation*, *26*(2), 289–315.

Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR), San Diego, CA, USA*.

Zacharakis, A., Pastiadis, K., Reiss, J. D., & Papadelis, G. (2012). Analysis of musical timbre semantics through metric and non-metric data reduction techniques. In *Proceedings of the 12th International Conference on Music Perception and Cognition (ICMPC12) and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music (ESCOM 08), Thessaloniki, Greece*, (pp. 1177–1182).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision, Zurich, Switzerland*, (pp. 818–833). Springer.

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). Mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR), Vancouver, Canada*.

Zhang, X., Yu, Y., Gao, Y., Chen, X., & Li, W. (2020). Research on singing voice detection based on a long-term recurrent convolutional network with vocal separation and temporal smoothing. *Electronics*, *9*(9), 1458.

Zhang, Z., & Schuller, B. (2012). Semi-supervised learning helps in sound event classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan*, (pp. 333–336).

Zheng, T., Seetharaman, P., & Pardo, B. (2016). Socialfx: Studying a crowdsourced folksonomy of audio effects terms. In *Proceedings of the 24th ACM international conference on Multimedia*, (pp. 182–186).

Zhou, X., Wang, D., & Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*.

Zhou, Y.-T., & Chellappa, R. (1988). Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks, San Diego, CA, USA*, (pp. 71–78).

Zsebők, S., Nagy-Egri, M. F., Barnaföldi, G. G., Laczi, M., Nagy, G., Vaskuti, É., & Garamszegi, L. Z. (2019). Automatic bird song and syllable segmentation with an open-source deep-learning object detection method–a case study in the collared flycatcher. *Ornis Hungarica*, *27*(2), 59–66.