Speech Processing in Real-Time

Connor Boyd-Lyon

MSc Data Science

Supervised by Dr. Phil Smith

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

2024-25

Abstract

This is the abstract...
this is still the abstract...

Declarations

I certify that this project is my own work. Debugging was assisted by GPT-4 & GPT-5. The report structure and initial drafts are mine; final editing was assisted by GPT-5 for clarity and consistency. All outputs have been reviewed and edited to accurately reflect my work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Speech is the primary means of human communication, and the ability of a computer to analyse and understand speech has been a longstanding challenge in computer science [44]. Advances in machine learning and the availability of large-scale datasets have enabled substantial progress in speech processing tasks [25], enabling new technologies such as virtual assistants [17] and translation services [15] to real-time communication platforms, where robust and efficient models are increasingly essential [27, 2].

Despite significant progress, many existing approaches remain impractical for real-world deployment. A key limitation is their reliance on multi-channel input [16, 36] or high-capacity models and large-scale computational resources [32], which are rarely available in consumer or embedded devices. Datasets place strong emphasis on spatial information derived from microphone arrays, which cannot be exploited in mono-channel environments. This creates a mismatch between research benchmarks and the constraints of applications that demand compact, low-latency, and robust solutions.

Addressing this gap requires approaches that remain effective in mono-channel conditions while retaining efficiency in both training and inference. Lightweight convolutional neural networks (CNNs) [3] operating on spectrogram representations provide one promising direction, as they capture relevant time–frequency structure without the computational burden of recurrent or transformer architectures [20]. Yet, relatively little research has focused on developing models that jointly satisfy the requirements of mono-channel input, low-latency inference, and resilience to overlapping speakers, leaving a clear opportunity for further work.

This thesis investigates the development of real-time transcription and speaker recognition methods for multi-speaker audio under mono-channel, resource-constrained conditions. The work explores spectrogram-based convolutional architectures as a foundation for lightweight, robust pipelines that can support overlapping speech and adapt to noisy environments. By evaluating dataset suitability, optimising model design, and examining trade-offs between accuracy, latency, and deployment feasibility, this project aims to contribute towards bridging the gap between research-driven advances in speech processing and the demands of practical, real-world usage.

Chapter 2

Literature Review

2.0.1 Speech Processing

Speech processing is the field focused on analysing and modelling human vocal communication signals [26]. A primary task in speech processing is Automated Speech Recognition (ASR), which aims to convert raw speech audio into text [30]. ASR consists of many components, each a task in itself; examples include noise cancellation [42], speaker diarisation [4] and speaker counting [13]. Modern speech processing solutions are often implemented as pipelines for this reason; Pyannote [9, 8], Speechbrain [34] and NeMo [24] are leading Python packages which use "modular neural building blocks" to solve these tasks. Figure 2.0.2 illustrates one potential structure of such a pipeline.

2.0.2 Machine Learning for Speech Processing

Modern deep learning techniques have revolutionised speech processing, with models achieving state-of-the-art results on large-scale datasets [20]. However, successes in controlled or offline settings often do not generalise well to noisy, real-world, or resource-constrained environments - this has been understood since early research [18].

Voice activity detection (VAD) and speaker counting are essential upstream components in the speech processing pipeline [33]. VAD identifies whether speech is present in an audio segment, supporting segmentation and reducing computational load for subsequent modules. Speaker counting estimates the number of distinct concurrent speakers, which is critical in overlapped speech scenarios [6] where diarisation and ASR performance degrade significantly without accurate overlap detection.

Progress in VAD has been strong, with modern model architectures achieving near-human performance in controlled conditions [37]. Speaker counting, however, remains challenging due to the complexity of overlapping voices [13]. Existing methods use recurrent neural networks for temporal modelling [45] and (more recently) transformer-based methods [20]. Despite these advances, most approaches operate offline and assume access to multi-channel or far-field microphone arrays, limiting their applicability in consumer or embedded contexts [40]. This is because real-time processing introduces additional constraints: latency, robustness to dynamic overlap, and computational efficiency, all of which degrade as model complexity increases [2]. Current systems often struggle to meet these requirements simultaneously. Many models rely on heavy architectures unsuited for deployment on embedded hardware, or are trained on datasets containing multiple microphones which is unrealistic in real-world applications [12]. This leaves a gap between academic research and production-level deployment.

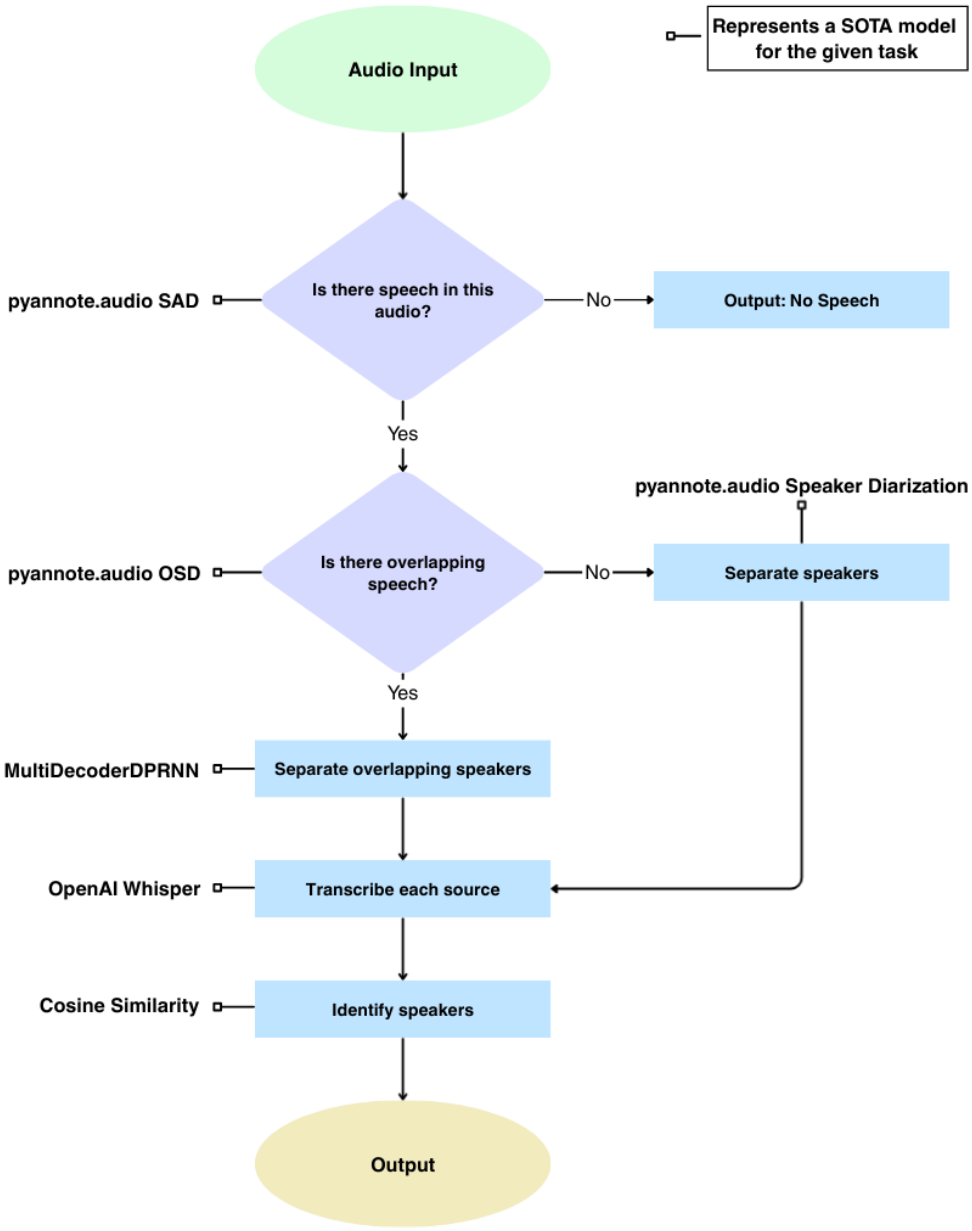


Figure 2.1: A typical speech processing pipeline, making use of SOTA models & methods [9, 47, 31, 38]. The end-goal of speech processing is to fully seperate and transcribe overlapping audio in a busy environment; each piece of the pipeline attempts a solution of one piece of the problem by assuming that previous parts are solved.

In contrast to current state-of-the-art (SOTA) solutions, convolutional neural networks (CNNs) [3] provide a promising approach for lightweight, real-time speech processing. CNNs are a deep learning architecture built on convolutional layers; these layers can break down images into abstract features, which are learnable by an artificial neural network [19]. These convolutional layers can be stacked to progressively capture higher-level representations. Unlike recurrent and transformer based architectures, CNNs typically require fewer parameters, as convolution and pooling reduce input dimensionality. This behaviour means that CNNs are well-suited for real-time deployment, especially as edge devices become more popular [11]. Initially popularised in computer vision tasks such as image classification and object detection [35], CNNs are also well-suited to learning from spectrograms, which represent speech in a two-dimensional time-frequency space [43]. Spectrograms encode frequency content over time, with intensity representing amplitude (Appendix A). This resembles an image, making them naturally compatible with convolutional feature extractors.

2.0.3 Datasets in Speech Processing

Several datasets support research in speech processing, including LibriSpeech and its derivatives, LibriMix, and LibriCSS [12]. LibriCSS is particularly popular; it's derived from LibriSpeech via playing and re-recording in a controlled setting using microphone arrays to simulate realistic room acoustics. Many models exploit multi-channel datasets to maximise spatial information available during training, but this causes major issues as models fail to generalise in a mono-channel environment. This limitation is critical for consumer hardware deployment, which requires compact, low-latency, and robust models. Dataset size is also important: large datasets are needed for SOTA models, but overly large datasets may be impractical to train in reasonable time.

Table 2.1: Summary of key datasets relevant to speech processing.

Dataset	**Size**	**Details**
LibriSpeech [29]	1000h	Widely used ASR benchmark; clean read English speech; single-channel; limited real-world noise.
LibriCSS [12]	∼10h	7-channel multi-speaker recordings derived from LibriSpeech; conversational-style; seminal for overlap and diarisation research.
RIR & Noise [23]	Various	Room Impulse Response and noise recordings; used for data augmentation and simulation of acoustic environments.
AMI Meeting Corpus [10]	100h	Multi-channel, multi-speaker meeting recordings; natural conversational speech; annotated for diarisation and ASR.
CHiME Challenges (1–6) [5]	Varies (5–100h)	Series of corpora for robust ASR in noisy, far-field, real-world environments; includes multi-channel recordings.
LibriMix [41]	500h+	Mixtures of LibriSpeech utterances; designed for source separation and speech enhancement; synthetic overlap.
VoxCeleb (1 & 2) [28, 14]	2000h+	Large-scale speaker recognition dataset from YouTube interviews; diverse conditions; single-channel speech.

Summary

Voice activity detection and speaker counting are critical upstream components of modern speech processing pipelines. While transformer-based and recurrent models achieve state-of-the-art performance on large-scale datasets, their reliance on multi-channel input and heavy architectures limits applicability in real-time, resource-constrained environments. Existing datasets such as LibriCSS further reinforce this gap by emphasising spatial information that's unavailable to consumer hardware. In contrast, CNN-based approaches offer a lightweight, spectrogram-driven alternative that better suits real-world deployment constraints. However, there remains a shortage of both datasets and models designed for the combined requirements of mono-channel input, low-latency inference, and robustness to overlapping speakers, highlighting a clear opportunity for further research.

Chapter 3

Methodology

3.1 Data Collection

The primary dataset used for model training was LibriCSS [12], a corpus designed to evaluate continuous speech separation systems. The corpus comprises over ten hours of speech from 682 distinct speakers, in clips with varying levels of overlap (Table 3.1). Given its overlapping structure and multi-channel recordings, LibriCSS provides a suitable foundation for training a speaker-counting model under pseudo-realistic acoustic conditions.

Table 3.1: The types of audio present in LibriCSS [12].

Audio Type	**Meaning**
0L (zero-long)	No overlap, inter-utterance gap of ~ 3 seconds.
0S (zero-short)	No overlap, inter-utterance gap of ~ 0.5 seconds.
OV10 (overlap-10)	$\sim 10\%$ overlap on utterances.
OV20 (overlap-20)	$\sim 20\%$ overlap on utterances.
OV30 (overlap-30)	$\sim 30\%$ overlap on utterances.
OV40 (overlap-40)	$\sim 40\%$ overlap on utterances.

As the majority of LibriCSS samples contain overlapping speech, the dataset provided an insufficient number of zero-speaker 'noise' samples. To address this imbalance the Room Impulse Response and Noise (RIR&N) database [23] was incorporated into the training set. RIR&N contains approximately 45 minutes of non-speech audio, including white noise, reverberation, and ambient recordings. In addition, supplementary noise clips were recorded to further expand the zero-speaker class. These recordings were collected under varied conditions to capture a broader range of acoustic characteristics (Table 3.2), resulting in approximately 125 minutes of new data. In total the combined corpus amounted to roughly 13 hours of audio for pre-processing (Table 3.3).

Table 3.2: A description of each manually recorded audio file. Different noise levels and sources were incorporated to give the greatest possible variance in the data.

Audio File	**Duration**	**Description**	**Average Noise Level**
Downstairs 1	00:20:00	Walking around, indoor ambience	Low
Downstairs 2	00:05:00	Doing chores	High
Upstairs 1	00:02:00	Working at a desk	Medium
Upstairs 2	00:07:00	Doing chores, eating a snack	Medium
Upstairs 3	00:01:00	White noise	High
Upstairs 4	01:30:00	Left by a window; occasional cars passing & animal sounds	Low

Table 3.3: The amount of audio collected from each source.

Data Source	**Duration**
LibriCSS	∼10 hours
Room Impulse Response & Noise (RIR&N)	∼45 minutes
Author added samples	∼2 hours
Total	12:53:13

3.2 Data Pre-Processing

Each recording was downmixed to a single channel (mono) to emulate the conditions of a single-microphone recording. While this increased the difficulty of the learning task, it enhanced the model's applicability to real-world scenarios where multi-microphone arrays are typically unavailable. The LibriCSS corpus was originally captured using a seven-channel setup [12], providing rich spatial information that facilitates separation and recognition; such configurations are impractical in real-world applications.

Following downmixing, the audio was segmented into fixed-length clips of one second. The use of a consistent clip duration was necessary to standardise input lengths for the neural network, ensuring comparability across samples and maximising learnability. A duration of one second was selected as it provided sufficient temporal context for distinguishing between speakers whilst also maintaining a manageable input size for efficient training and inference.

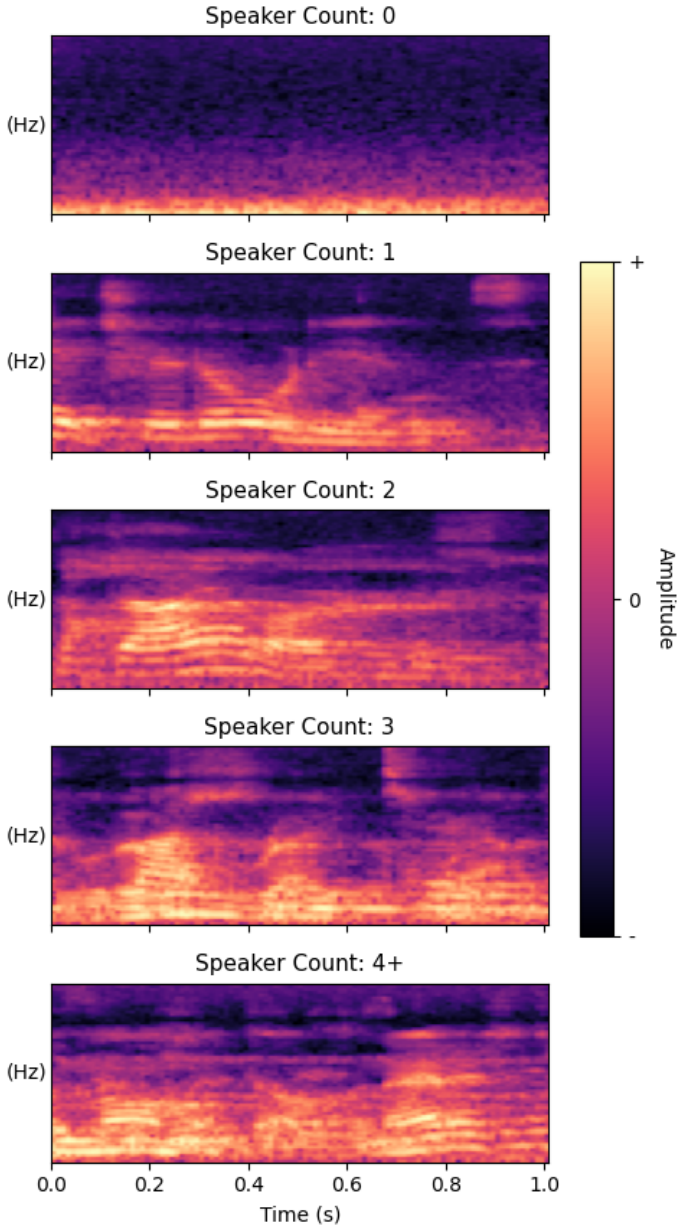


Figure 3.1: A series of spectrograms created with the `SpectrogramExtractor()` class on speech audio. Speaker count was found by manually cross-referencing the audio metadata. It's clear that spectral activity increases with speaker count. With a low number of speakers ($n = 1, 2$) the spectral patterns of each speaker are visible, whereas the result becomes increasingly obscure and noisy as speaker count increases. This suggests that learnability diverges as n increases.

To create the training samples, a custom pre-processing class `SpectrogramExtractor()` was implemented. This class transforms an audio clip into its corresponding spectrogram representation, which stores frequency, amplitude and temporal information in a single sample. By converting the signal into the time-frequency domain, the resulting spectrograms are learnable based on their . The `SpectrogramExtractor()` class was used on each clip to create 46,393 spectrograms (Figure 3.1).

After creating the spectrograms for each sample, the libricss metadata files (Appendix B) were scraped to find the speaker count and speaker ID(s) for each spectrogram. The number of speakers was used as the supervision label for each spectrogram in model training. The noise samples were labelled with Speaker Count = 0. The resulting labelled dataset was formatted as a DataFrame (Table 3.4).

Table 3.4: An extract of the resulting dataset from using the LibriCSS metadata (Appendix B) to find the number of speakers for each observation.

Spectrogram Directory	Clip Directory	Speaker Count	Speakers
.../OV40_session9_clip90.pt	.../OV40_session9_clip90.wav	2	['1995', '2961']
.../OV40_session9_clip91.pt	.../OV40_session9_clip91.wav	1	['2961']
.../OV40_session9_clip92.pt	.../OV40_session9_clip92.wav	1	['2961']
.../OV40_session9_clip93.pt	.../OV40_session9_clip93.wav	1	['2961']
.../OV40_session9_clip94.pt	.../OV40_session9_clip94.wav	1	['2961']
.../OV40_session9_clip95.pt	.../OV40_session9_clip95.wav	1	['2961']
.../OV40_session9_clip96.pt	.../OV40_session9_clip96.wav	1	['2961']
.../OV40_session9_clip97.pt	.../OV40_session9_clip97.wav	2	['2961', '7176']
.../OV40_session9_clip98.pt	.../OV40_session9_clip98.wav	2	['2961', '7176']
.../OV40_session9_clip99.pt	.../OV40_session9_clip99.wav	2	['2961', '7176']

Evaluating the Dataset

Two primary issues were identified in the created dataset. First, it contained only three speaker count classes [0, 1, 2], preventing a model trained on this data from generalising to scenarios involving three or more speakers. Second, the dataset exhibited substantial class imbalance (Table 3.5), which could have biased model training toward the more frequently occurring classes. To resolve these issues the dataset was augmented.

Table 3.5: Number of samples per speaker count before augmentation.

Speaker Count	Samples
0	11719
1	29309
2	5251

3.2.1 Data Augmentation

Data augmentation was employed to address the class imbalance present in the dataset, as well as the lack of samples representing higher speaker counts. Two methods were considered to address the imbalance; increase samples in underrepresented classes or reduce samples in overrepresented classes. The latter meant that valuable information could be lost, so alternative data sources were considered to increase class sizes. To create higher speaker audio, synthetic samples were generated by selecting pairs of clips and combining them. The clips were overlayed and normalised to produce realistic multi-speaker recordings. The spectrograms were extracted from the new clips via the usual process with `SpectrogramExtractor()`. Finally, the speaker counts for the synthetic samples were created by summing the counts of the original clips, and the observations were added to the dataset.

Clip selection was pseudo-random; clips were separated into groups by their number of speakers first, and pairs were then drawn randomly from two chosen groups. This meant that the number of samples for each class could be curated, and class balance was possible (Table 3.6). Additionally, checks were made using speaker ID to ensure that clips of the same speaker weren't overlayed, as combining two clips of the same voice could be interpreted as a single speaker. It was noticed that if a speaker began talking in the first or last milliseconds of a clip the speech wouldn't be audibly significant, so counting the speech would be inappropriate; a minimum utterance length of 0.2 seconds was required for labelling, so that insignificant speech couldn't distort the speaker count labels.

Table 3.8: A comparison of the label structure for the Speaker Counting model and the Voice Activity Detection (VAD) model.

Number of Speakers	Speaker Count Label	VAD Label
0	0	0
1	1	1
2	2	1
3	3	1
4	4+	1
5	4+	1

Table 3.6: Summary of data augmentation for multi-speaker classes. The original dataset was split into '0-speaker', '1-speaker' and '2-speaker' samples. These could then be overlayed to create a clip with any number of speakers; for example, overlaying '2-speaker' clips with '1-speaker' clips produced synthetic '3-speaker' clips.

Base Clips	Overlay Clips	Resulting Speaker Count	Samples Created
1-speaker	1-speaker	2	17,500
0-speaker	2-speaker	2	2,500
2-speaker	1-speaker	3	25,000
2-speaker	2-speaker	4+	10,000
3-speaker	1-speaker	4+	7,500
3-speaker	2-speaker	4+	7,500

Table 3.7: Number of samples per speaker count after augmentation. These class sizes were curated to reflect both the difficulty of the class to learn and it's occurrence in the world. For example, the '0' class, where observations are only silence and noise, was theoretically easier to learn than the higher classes, so less observations were needed. In contrast the '1' speaker class is the most common class in the real-world, so there are more observations for this class than the others. With this in mind, a strategically weighted dataset was created.

Speaker Count	Samples
0	11,719
1	29,309
2	25,251
3	25,000
4+	25,000

Changes for the Voice Activity Detection (VAD) task

Due to the binary nature of the VAD task the speaker labels needed to be altered (Table 3.2.1).

3.3 Model Architecture

An artificial neural network with convolutional layers was employed to extract feature maps from the spectrogram inputs. Convolutional layers efficiently captured local patterns in both time and frequency, making them well-suited for processing audio representations. Long Short-Term Memory (LSTM) and transformer-based architectures were considered, but were deemed less suitable due to higher computational cost and slower inference; convolutional neural networks (CNNs) achieved efficient feature extraction with substantially lower overhead, making them a suitable choice for real-time inference.

3.3.1 Architecture for the Voice Activity Detection (VAD) model

Voice Activity Detection (VAD) was implemented using a compact CNN, designed to classify short audio segments as either speech or non-speech (Figure 3.2). The network consists of two convolutional layers, with max-pooling applied after each to reduce spatial dimensions. Dropout was required to mitigate overfitting. The fully connected layers map the extracted features to the two output classes. The size is

intentionally small for efficient inference; modern CNNs are usually much more complex and are larger in size [46]. A detailed model architecture can be found in Appendix C.1.

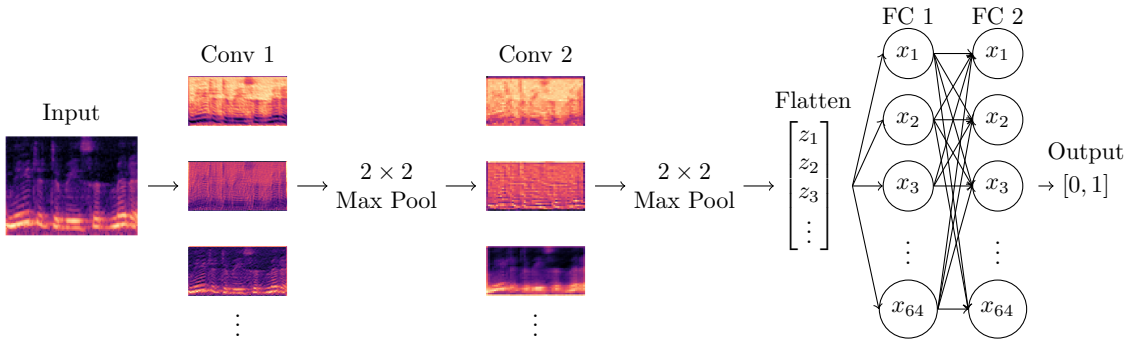


Figure 3.2: CNN architecture for the VAD model. An input spectrogram is fed through two convolutional layers with 3×3 kernels, applying 2×2 max pooling after each. This provides a series of 'feature maps' that represent the various features of the original image. These feature maps are flattened into a column vector, z, and learned by two fully connected layers with $n = 64$ nodes each. The output is a binary classification; '0' for no speech, '1' for speech.

3.3.2 Architecture for the Speaker Count Model

Speaker counting was implemented using a multi-class CNN, enabling the model to predict the number of concurrent speakers in an audio segment. Given the increased complexity of distinguishing multiple overlapping speakers compared to voice activity detection, a more complex architecture was required (Figure 3.3.2). Four convolutional layers were employed to extract increasingly abstract features and each fully-connected layer had 128 neurons (twice that of the VAD model). The network otherwise maintained a structure similar to the VAD model; a detailed model architecture can be found in Appendix C.2.

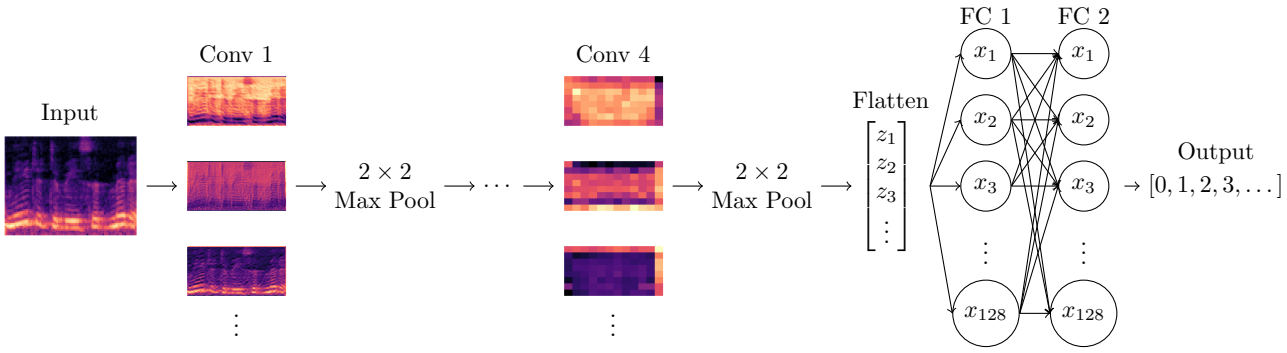


Figure 3.3: CNN architecture for the speaker count model. An input spectrogram is fed through four convolutional layers with 3×3 kernels, applying 2×2 max pooling after each. In comparison to Figure 3.2 the resulting feature maps are much smaller in spatial resolution but richer in representational detail. The feature maps are flattened into a column vector z and learned by two fully-connected layers with $n = 128$ nodes each. The output is a multi-class classification of how many distinct speakers are present in the audio.

3.4 Hyperparameter Tuning with Optuna

Each CNN architecture required careful hyperparameter (HP) selection to maximise precision in evaluation. Initial exploration was done via random search, where categorical values were drawn from a defined grid of candidate parameters (Table 3.9). Random search was successful in identifying promising regions of the hyperparameter space, but expectedly struggled beyond that due to high compute over a very large search space. Grid search was then considered as it could cover the entire search space, but was too slow and expensive to be practical.

Table 3.9: HP search space for the Speaker Count model's random search. These values were drawn from intuition and intentionally cover a very large parameter space.

Hyperparameter	Candidate values
Learning rate (η)	{1e-4, 1e-3, 1e-2}
Dropout probability	{0.1, 0.3, 0.5}
FC hidden units	{64, 128, 256}
Conv1 output channels	{8, 16, 32}
Conv2 output channels	{16, 32, 64}
Conv3 output channels	{32, 64, 128}
Conv4 output channels	{32, 64, 128}

To improve HP search efficiency, the Optuna package [1] was employed. Optuna contains wrappers to easily perform a guided random search with the ability to prune poor-performing trials early, reducing wasted computation. Each trial sampled a configuration of hyperparameters (Table 3.10) and trained a model for up to three epochs. Validation accuracy was monitored after each epoch, and trials that would not outperform the current best were terminated. A stratified subset of the dataset was used during tuning to reduce runtime while preserving class balance. The subset contained 25% of the total dataset making runtime $\sim 4\times$ faster. In total, 150 trials were run, covering a substantial portion of the 324 possible configurations, which was sufficient to identify a near-optimal configuration.

Training used the Adam optimiser with cross-entropy loss. Adam was chosen for its robustness in handling sparse gradients and its ability to adapt learning rates per parameter, making it well suited to CNNs trained on audio spectrogram data [22]. Cross-entropy was selected as the objective function because the tasks under study were classification problems. Together, these choices provided stable convergence and compatibility with the Optuna optimisation framework.

Table 3.10: Hyperparameter search space explored by Optuna for the Speaker Count model. The space was informed by the previous random search, for example setting fully-connected layer size of 128, as it was a strong performer in the random search. There were 324 total possible configurations, but it wasn't deemed necessary to try all configurations; most configurations provided insignificant changes, and a near-optimal configuration was considered sufficient.

Hyperparameter	Candidate values
Learning rate (η)	{0.01, 0.001, 0.005}
Dropout probability	{0.1, 0.2, 0.3}
FC hidden units	128 (fixed)
Conv1 output channels	{8, 16}
Conv2 output channels	{32, 64}
Conv3 output channels	{32, 64, 128}
Conv4 output channels	{32, 64, 128}

3.5 Real-Time Evaluation with Streamlit

After training, both the Voice Activity Detection and Speaker Count models were deployed in lightweight real-time applications. Each application was implemented using the Streamlit library [39], which provided a straightforward way to combine live microphone input, model inference, and interactive visualisation in a single interface. In both cases the process was identical; the trained model was loaded with its saved weights and connected to a preprocessing pipeline. The preprocessing module resampled audio captured from the system microphone to 16 kHz, segmented it into fixed one-second windows, and converted each segment into spectrograms suitable for model input.

Streaming audio capture was managed by the sounddevice library [7]. One-second clips were placed into a queue and processed continuously by a background thread, which applied preprocessing and forwarded the results to the model for inference. Predictions were stored in a rolling buffer, enabling real-time display and short-term history tracking. This allowed the models to be tested under realistic streaming conditions. Because both applications shared the same architecture, only differing in the loaded model and output interpretation, the methodology was consistent. Streamlit was chosen to minimise development overhead and make results accessible without requiring a separate deployment framework.

Chapter 4

Results

4.1 Results of the Optuna Studies

The final hyperparameter (HP) configurations for the three models are summarised in Table 4.1. The VAD model uses smaller fully connected layers and 2 convolutional layers for its binary classification task. The speaker counting model employs a 2× deeper convolutional stack with larger channel counts and a higher dropout, reflecting the increased complexity of distinguishing five classes. The post-VAD model maintains the same convolutional architecture as the speaker counting model but reduces the output classes to four, following the removal of the silence class.

Table 4.1: Hyperparameter configurations for the three models. Each was derived from an initial random search followed by an Optuna [1] study over a smaller search space (Section 3.4).

Hyperparameter	**VAD (2-class)**	**Speaker Count (5-class)**	**Post-VAD (4-class)**
Learning rate (lr)	0.001	0.001	0.001
Dropout probability	0.2	0.3	0.3
FC hidden units	64	128	128
Conv1 output channels	32	8	8
Conv2 output channels	64	32	32
Conv3 output channels	–	128	128
Conv4 output channels	–	128	128
Number of classes	2	5	4

4.2 Model Results

Each model was assessed on per-class accuracy, precision, recall & F1-Score, as is customary in modern classification tasks. This allows the direct comparison of each model despite architectural differences (Table 4.2). The VAD model was trained over a binary representation of the dataset (Section 3.2.1). Figure 4.2 shows the results of VAD model validation. The 5-Class Speaker Count model was trained and validated three times, over various levels of stratification (Figure 4.2). The 4-Class model was trained once over the entire dataset (Figure 4.2).

Table 4.2: Full Summary for the VAD, 5-class, and 4-class models. The key metrics used to assess a given model were Accuracy, Precision, Recall and F1-Score.

Metric / Class	**VAD (2-class)**	**5-Class Count**	**4-Class Count**
Validation Accuracy	99.43%	65.09%	61.43%
Per-Class Accuracy			
Class 0	0.970	0.996	–
Class 1	1.000	0.989	0.989
Class 2	–	0.635	0.503
Class 3	–	0.300	0.384
Class 4	–	0.466	0.510
Precision			
Class 0	1.000	0.986	–
Class 1	0.993	0.906	0.919
Class 2	–	0.410	0.423
Class 3	–	0.414	0.387
Class 4	–	0.765	0.702
Recall			
Class 0	0.970	0.996	–
Class 1	1.000	0.989	0.989
Class 2	–	0.635	0.503
Class 3	–	0.300	0.384
Class 4	–	0.466	0.510
F1 Score			
Class 0	0.985	0.991	–
Class 1	0.997	0.946	0.953
Class 2	–	0.498	0.460
Class 3	–	0.348	0.386
Class 4	–	0.579	0.591

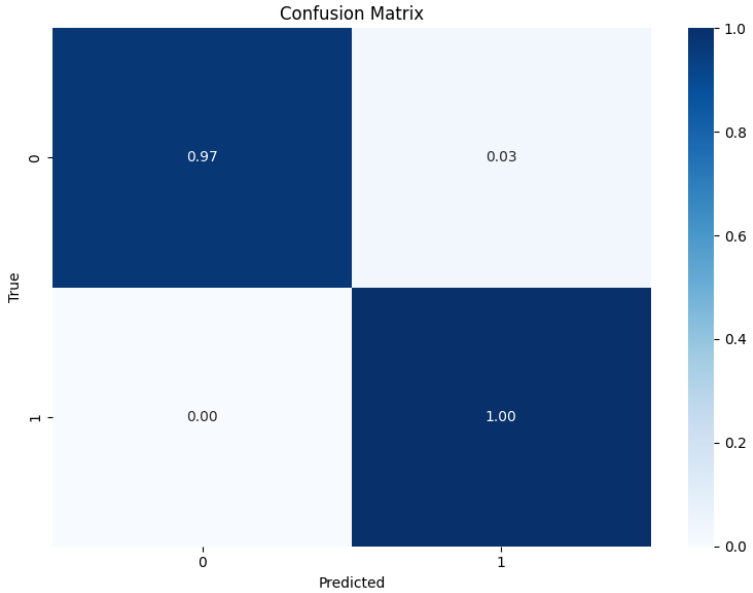
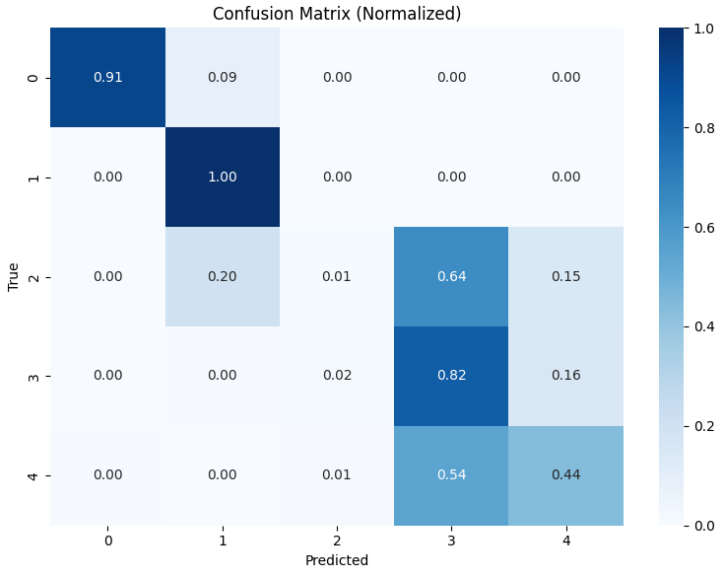
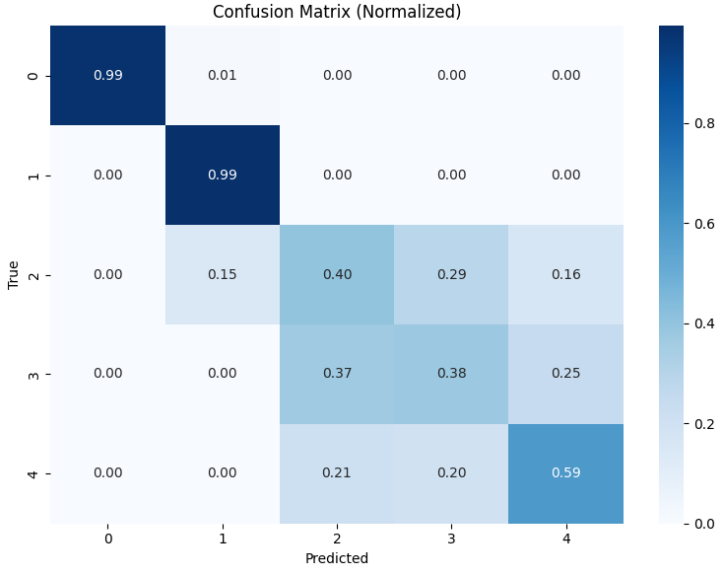


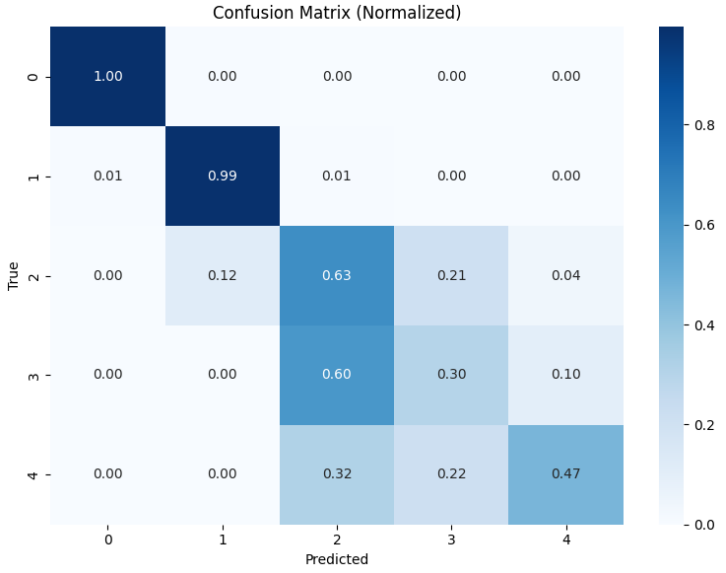
Figure 4.1: The confusion matrix for the Voice Activity Detection (VAD) model validation.



(a) 1% Subset



(b) 50% Subset



(c) Full dataset

Figure 4.2: Confusion matrices of the 5-class model's performance across different dataset stratifications.

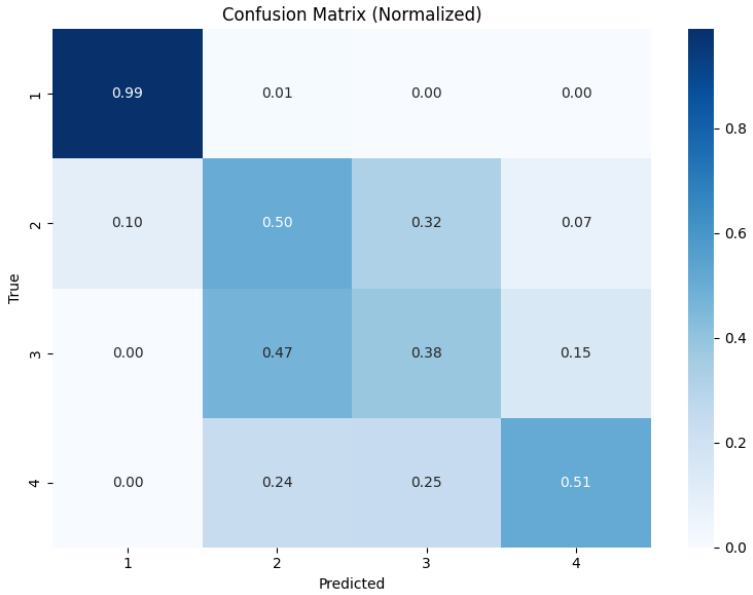


Figure 4.3: The confusion matrix result of validating the 4-class model.

4.3 VAD and Speaker Counting Real-Time Results

Both the VAD model and 5-Class Speaker Count model were evaluated in real-time. The VAD model performs at near SOTA level and is held back only by the front-end. The model almost always correctly identifies speech, and is able to ignore noise both in the foreground and background. The Speaker Counter model struggles, and often fails to correctly evaluate speech in real-time.

Figure 4.4: A screenshot of the resulting app that runs the Speaker Count model in real-time. The app was made entirely in python using Streamlit [39].

Chapter 5

Discussion

5.1 Dataset & Configuration Observations

5.1.1 Overall Effectiveness of the Dataset

The dataset, which consisted of audio from LibriCSS [12] and Room Impulse Response & Noise [23] as well as manually collected noise samples, was sufficient in providing spectral patterns that were learnable by a convolutional neural network (CNN). The classes were balanced, except for the zero class, which was still learned very well, despite having approximately half the samples of the other classes. A larger and more diverse dataset would have been better for the models to learn more types of speech, and more types of noise for the zero class. It's likely that the models would fail in a very busy environment, such as a factory, since they are only trained on data from an office atmosphere.

5.1.2 Effect of Dataset Size and Stratification

The impact of dataset size and stratification on the 5-class speaker counting model is clear (Figure 4.2); the three models (trained on 1%, 50% and 100% of the data, respectively) highlight that high accuracy is easily achieved for the lower classes (0 and 1) with minimal training, however even the 100% model struggles to distinguish a strong decision boundary on the higher classes (2-4+). The speaker count model may have benefited from a larger dataset, especially one which includes a clearer decision boundary between the higher classes. The dataset size was more than sufficient for the VAD model, which learned the decision boundary very well.

5.1.3 Hyperparameter Tuning Insights

The VAD model employs fully connected layers of 64 neurons, half that of the speaker counting models; this is understandable given that VAD is a simpler classification task than speaker counting. Additionally, the higher dropout probability in the speaker counting models mitigates overfitting given their deeper architectures. Interestingly, the Optuna [1] studies on the 4-class and the 5-class speaker counting models resulted in the same hyperparameter configuration, indicating that number of classes is insignificant when deciding a configuration, given that the task and training data are the same.

5.2 Model Performance

5.2.1 Voice Activity Detection (VAD) Model

The VAD model achieves near-perfect validation accuracy (99.43%), with insignificantly lower accuracy on the silence class (Class 0) than on the speech class (Class 1). The relatively small fully connected layers and shallow convolutional stack were sufficient for this binary task, reflecting the ability of CNNs to capture distinctive spectral patterns of human speech and to reliably distinguish speech from other sounds.

Risk of Overfitting

The extremely high validation accuracy of the VAD model raises the possibility of overfitting, particularly given the near-perfect separation achieved during training and validation. This was a concern because the model may have simply memorised training examples rather than learning generalisable representations of speech and silence. To investigate, the model was deployed in a real-time environment (Section 4.3) and evaluated using voices and noises not present in the training dataset. The model demonstrated strong generalisation by accurately detecting speech from previously unseen speakers and correctly classifying unseen background sounds such as finger snapping. These results suggest that the model effectively captures transferable spectral characteristics of speech and non-speech audio, rather than overfit the training data.

5.2.2 5-Class Speaker Counting Model Performance

The 5-class model achieved moderate overall accuracy (65.09%), with extreme variation in per-class performance (Table 4.2). Classes 0 and 1 achieved nearly perfect accuracy; these classes boosted the overall accuracy to a higher level. Classes 2-4 showed substantially lower performance and brought the overall accuracy down. This pattern is attributable to the difficulty of distinguishing overlapping speaker counts. The high per-class training losses for Classes 2-4 suggest that additional model complexity may be necessary to improve classification.

5.2.3 4-Class Post-VAD Model Performance

The 4-class model achieved slightly lower accuracy (61.43%) compared to the 5-class model, reaffirming that the accuracy score was being boosted mainly by the 0 and 1 classes. The remaining classes continue to dominate errors, highlighting the inherent difficulty of distinguishing mid-level speaker counts. The use of the same hyperparameters as the 5-class model may reflect limitations of the tuning process or insensitivity to the number of output classes.

5.2.4 Observations on Per-Class Performance

It's clear that the models encountered various levels of difficulty in learning the decision boundary for each class. Notably, the difficulty to learn the spectral patterns of speech increases exponentially as the number of speakers increase (Figure 5.2.4). This behaviour indicates an upper limit on the ability of CNNs to count speakers, however it also promises success on speaker counts 2-4+ given a more complex model. This trend is intuitive, as in real-world settings it's relatively easy to identify a small number of distinct voices in a room, whereas accurately separating several overlapping speakers, such as in a crowded environment, becomes exponentially difficult. This highlights the challenge of defining obscure decision boundaries in classification tasks.

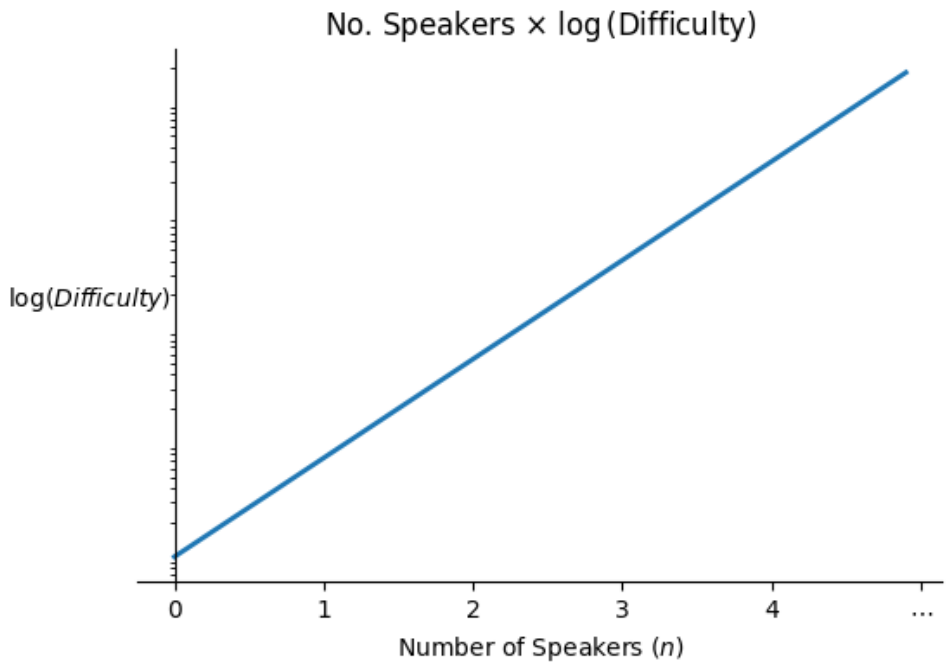


Figure 5.1: A log plot showing that difficulty increases exponentially as the number of speakers in audio increases. This is because as speaker count increases, the decision boundary between classes becomes exponentially more obscure.

5.2.5 Real-Time Evaluation

The success of the VAD model in real-time indicates that CNNs are a valid method for detecting speech in spectrograms in a production environment. The lightweight nature of the model means that it can be deployed as software on edge devices or mobile phones which can't handle the requirements of many SOTA solutions.

5.3 Limitations and Future Work

The VAD model showed success in real-time, however an application of this model in the real-world would require some changes; learning the noise of the environment and possibly dialects of speakers may be necessary for the model to reach its potential. This could be done by improving the training data or by fine-tuning in the production area.

The Speaker Count model isn't suitable for real-time application; a more complex model is required to better separate the decision boundaries. One way of achieving this is to use ensemble learning to create a pipeline (Figure 5.3).

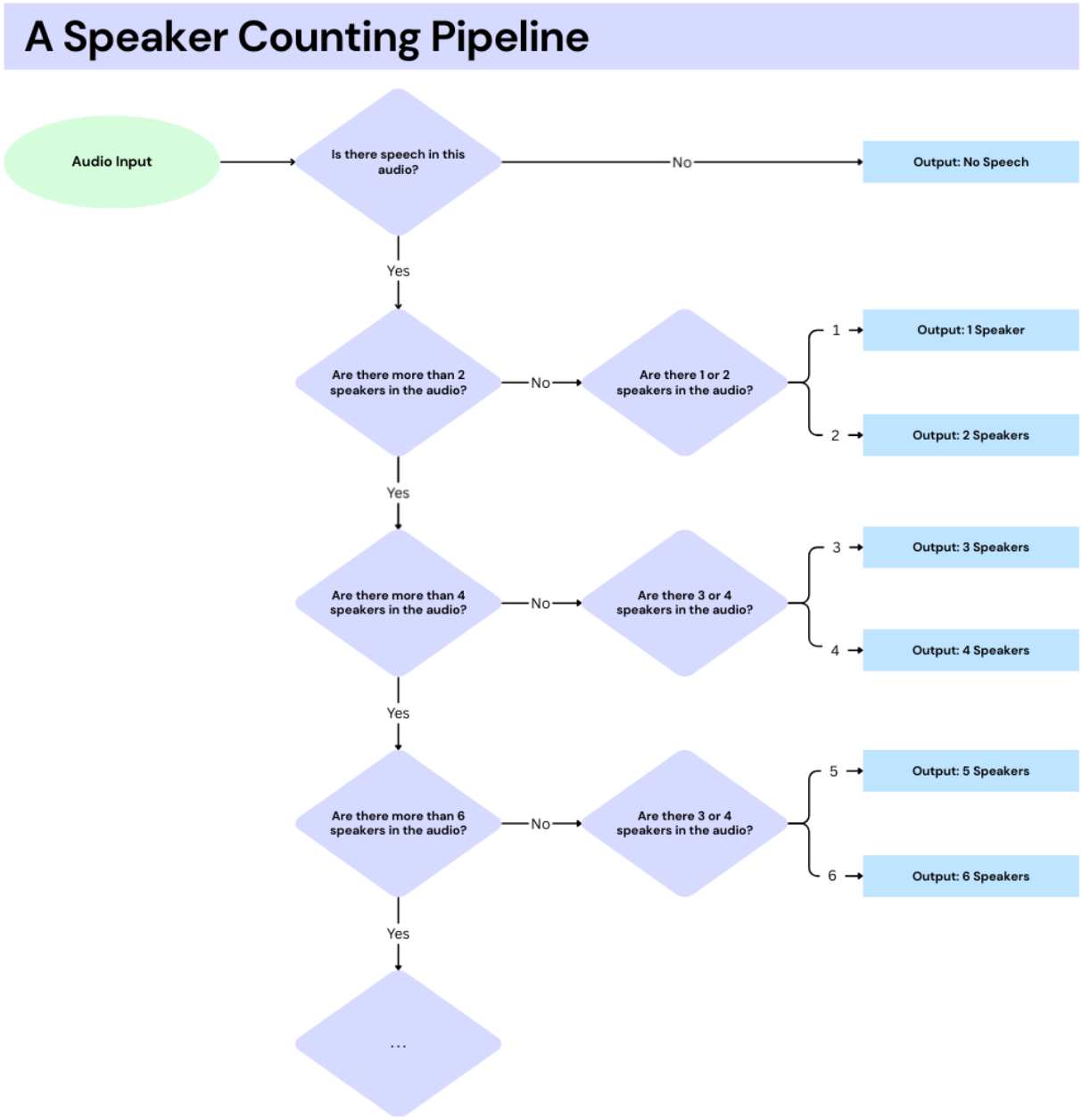


Figure 5.2: A theoretical pipeline for better learning of classification decision boundaries. It allows for more compute to be spent on the toughest learning, by training complex models only where necessary. The example begins with VAD segmentation, followed by a series of coarse speaker count classifiers that filters an image into the relevant category. Once a category is decided, a dedicated sub-model would distinguish between the specific counts within that range. This hierarchical approach allows the model to focus on narrower decision boundaries at each stage, potentially improving accuracy in cases where speaker overlap or acoustic similarity makes global classification difficult.

Its possible that even with a very complex CNN, the speaker counting task is not possible to solve completely. Additionally, the complexity required to solve the speaker counting task may be too high to also function in real-time. If either of these are the case, research should instead focus on improving the efficiency & sustainability of alternative models, as these are proven to work in a development environment but aren't currently suitable for real-world use.

Chapter 6

Conclusion

This project investigated the feasibility of applying convolutional neural networks (CNNs) to the tasks of voice activity detection (VAD) and multi-speaker counting. The motivation behind this was the increasing need for lightweight, real-time models that can operate on edge devices and mobile platforms, in contrast to the resource-intensive nature of many state-of-the-art speech processing methods. The study therefore asked a practical question: "can standalone CNNs provide sufficient performance to make deployment viable in real environments?"

The findings demonstrate a clear distinction between the two tasks. For VAD, the CNN achieved near-perfect performance, both in validation and in real-time deployment tests. Importantly, the model generalised effectively to unseen speakers and background noises, confirming that the spectral patterns of speech and silence can be reliably captured without resorting to large-scale or highly complex architectures. This result provides a strong case for using CNN-based VAD in resource-constrained scenarios. In short, CNNs are not only theoretically capable of distinguishing speech from non-speech, but they are also practically deployable in real-world environments.

In contrast, the speaker counting task proved substantially more challenging. While the models achieved high accuracy for low speaker counts (0 and 1), performance deteriorated sharply for higher counts (2-4+). This reflects the difficulty of distinguishing overlapping voices where decision boundaries are obscure - and the limitations of shallow CNNs when applied to complex classification spaces. Even with larger datasets and hyperparameter tuning, the models were unable to establish strong decision boundaries for higher classes. This points to a likely ceiling on the effectiveness of straightforward CNN-based approaches in this domain.

From these results, two broader implications emerge. First, task complexity matters; CNNs are well-suited to binary classification problems with distinct boundaries, such as VAD, but struggle when faced with tasks that require separating highly entangled features. Second, dataset design is critical. The models performed well where classes reflected clear boundaries but struggled where examples were inherently ambiguous (e.g. three versus four overlapping speakers). This highlights the importance of both quantity and quality of data in speech tasks. Taken together, these findings suggest that CNN-based VAD is ready for real-world deployment, while CNN-based speaker counting remains an open research challenge. The gap between the two tasks illustrates the need for either more sophisticated model architectures or entirely different approaches if accurate speaker counting is to become viable in real-time environments.

In summary, this project identified that the speech processing field hasn't yet matured beyond a development environment, and sought to address speech processing tasks in real-time. Research found that CNNs provide practical and lightweight solutions for real-time speech detection, but their ability to scale to more complex tasks is limited. This has meaning beyond the models themselves; progress in speech processing depends not only on algorithmic advances but also on aligning model design with the realities of the target environment. These findings therefore invite further research on identifying a middle ground where model complexity is sufficient for the task, yet efficient and robust enough to function outside controlled settings. Ultimately, current research for the field should not be to chase state-of-the-art performance in isolation, but to design solutions that translate reliably into the environments where they are most needed.

Bibliography

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha. Efficient machine learning for big data: A review. *Big Data Research*, 2(3):87–93, 2015.

[3] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8(1):53, 2021.

[4] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on audio, speech, and language processing*, 20(2):356–370, 2012.

[5] J. Barker, R. Marxer, E. Vincent, and S. Watanabe. The CHiME speech separation and recognition challenge. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 504–511. IEEE, 2015.

[6] K. Boakye, B. Trueba-Hornero, O. Vinyals, and G. Friedland. Overlapped speech detection for improved speaker diarization in multiparty meetings. In *2008 IEEE international conference on acoustics, speech and signal processing*, pages 4353–4356. IEEE, 2008.

[7] M. Brandl. sounddevice: Play and record sound with python. `https://python-sounddevice.readthedocs.io/`, 2021. Accessed: 2025-08-26.

[8] H. Bredin and A. Laurent. End-to-end speaker segmentation for overlap-aware resegmentation. In *Proc. Interspeech 2021*, Brno, Czech Republic, August 2021.

[9] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill. pyannote.audio: neural building blocks for speaker diarization. In *ICASSP 2020, IEEE International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain, May 2020.

[10] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, et al. The AMI meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 28–39. Springer, 2005.

[11] J. Chen and X. Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[12] Z. Chen, T. Yoshioka, L. Lu, T. Zhou, Z. Meng, Y. Luo, J. Wu, X. Xiao, and J. Li. Continuous Speech Separation: Dataset and Analysis. In *ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020. arXiv:2001.11482.

[13] S. R. Chetupalli and E. A. Habets. Speaker counting and separation from single-channel noisy mixtures. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1681–1692, 2023.

[14] J. S. Chung, A. Nagrani, and A. Zisserman. VoxCeleb2: Deep speaker recognition. In *Interspeech 2018*, pages 1086–1090. ISCA, 2018.

[15] J. Gu, G. Neubig, K. Cho, and V. O. Li. Learning to translate in real-time with neural machine translation. *arXiv preprint arXiv:1610.00388*, 2016.

[16] R. Gu, S.-X. Zhang, Y. Xu, L. Chen, Y. Zou, and D. Yu. Multi-modal multi-channel target speech separation. *IEEE Journal of Selected Topics in Signal Processing*, 14(3):530–541, 2020.

[17] R. High. The era of cognitive systems: An inside look at ibm watson and how it works. *IBM Corporation, Redbooks*, 1:16, 2012.

[18] Y. Huang. *Adaptive signal processing: applications to real-world problems.* Springer Science & Business Media, 2003.

[19] IBM. What are convolutional neural networks? `https://www.ibm.com/think/topics/convolutional-neural-networks`. Accessed: 2025-08-28.

[20] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub, and W. Pedrycz. A comprehensive survey on applications of transformers for deep learning tasks. *Expert Systems with Applications*, 241:122666, 2024.

[21] G. Jones. Echolocation calls of british bats. `https://www.garethjoneslab.com/british-bat-echolocation`, 2019. Accessed: 2025-08-16.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5220–5224, 2017.

[24] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, S. Kriman, S. Beliaev, V. Lavrukhin, J. Cook, et al. Nemo: a toolkit for building ai applications using neural modules. *arXiv preprint arXiv:1909.09577*, 2019.

[25] A. Mehrish, N. Majumder, R. Bharadwaj, R. Mihalcea, and S. Poria. A review of deep learning techniques for speech processing. *Information Fusion*, 99:101869, 2023.

[26] A. Mehrish, N. Majumder, R. Bharadwaj, R. Mihalcea, and S. Poria. A review of deep learning techniques for speech processing. *Information Fusion*, 99:101869, 2023.

[27] G. Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 55(12):1–37, 2023.

[28] A. Nagrani, J. S. Chung, and A. Zisserman. VoxCeleb: a large-scale speaker identification dataset. In *Interspeech 2017*, pages 2616–2620. ISCA, 2017.

[29] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. LibriSpeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[30] L. R. Rabiner and R. W. Schafer. Introduction to digital speech processing. *Foundations and Trends® in Signal Processing*, 1(1–2):1–194, 2007.

[31] A. Radford, J. W. Kim, T. Xu, G. Brockman, and C. McLeavey. Robust speech recognition via large-scale weak supervision, 2022. Accessed: 2025-08-18.

[32] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[33] J. Ramirez, J. M. Górriz, and J. C. Segura. Voice activity detection. fundamentals and speech recognition system robustness. *Robust speech recognition and understanding*, 6(9):1–22, 2007.

[34] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, et al. Speechbrain: A general-purpose speech toolkit. *arXiv preprint arXiv:2106.04624*, 2021.

[35] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[36] T. N. Sainath, R. J. Weiss, K. W. Wilson, B. Li, A. Narayanan, E. Variani, M. Bacchiani, I. Shafran, A. Senior, K. Chin, et al. Multichannel signal processing with deep neural networks for automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(5):965–979, 2017.

[37] M. Sharma, S. Joshi, T. Chatterjee, and R. Hamid. A comprehensive empirical review of modern voice activity detection approaches for movies and tv shows. *Neurocomputing*, 494:116–131, 2022.

[38] S. Shum, N. Dehak, R. Dehak, and J. R. Glass. Unsupervised speaker adaptation based on the cosine similarity for text-independent speaker verification. In *Odyssey*, volume 6, page 16, 2010.

[39] Streamlit. Streamlit: A faster way to build and share data apps. `https://streamlit.io/`. Accessed: 2025-08-18.

[40] Z.-Q. Wang and D. Wang. Count and separate: Incorporating speaker counting for continuous speaker separation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11–15. IEEE, 2021.

[41] G. Wichern, J. Antognini, M. Flynn, L. McQuinn, D. Crow, E. Manilow, and J. Le Roux. Wham!: Extending speech separation to noisy environments. In *Interspeech 2019*, pages 1368–1372. ISCA, 2019.

[42] B. Widrow, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, J. E. Dong, and R. C. Goodlin. Adaptive noise cancelling: Principles and applications. *Proceedings of the IEEE*, 63(12):1692–1716, 1975.

[43] L. Wyse. Audio spectrogram representations for processing with convolutional neural networks. *arXiv preprint arXiv:1706.09559*, 2017.

[44] D. Yu and L. Deng. *Automatic speech recognition*, volume 1. Springer, 2016.

[45] Y. Yu, X. Si, C. Hu, and J. Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[46] Y. Zheng, H. Huang, and J. Chen. Comparative analysis of various models for image classification on cifar-100 dataset. *Journal of Physics: Conference Series*, 2711:012015, 02 2024.

[47] J. Zhu, R. A. Yeh, and M. Hasegawa-Johnson. Multi-decoder dprnn: High accuracy source counting and separation. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3420–3424, 2021.

Appendix A

Spectrogram Example

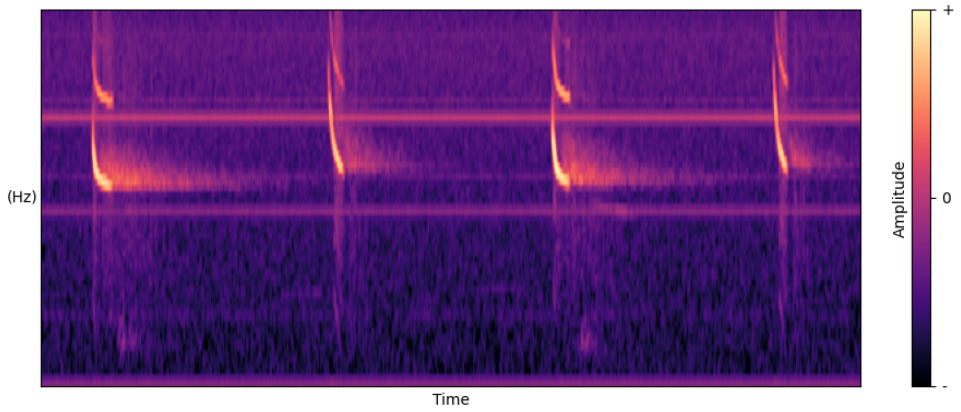


Figure A.1: A spectrogram of a bat's echolocation [21]. Spectrograms encode temporal and frequency information simultaneously, making them well-suited for CNN-based feature extraction.

Appendix B

LibriCSS Metadata

Table B.1: A metadata sample from LibriCSS [12]. By scraping the start and end time of each utterance it's possible to know how many speakers are present in each clip. For example it's clear that at $t = 8$ there are two speakers, because the first and second speakers are speaking in overlap.

Start Time	**End Time**	**Speaker**	**Utterance ID**	**Transcription**
3.000	10.260	3570	3570-5695-0002	BUT AS WE DESCEND THE SOCIAL SCALE THE POINT IS PRESENTLY REACHED WHERE THE DUTIES OF VICARIOUS LEISURE AND CONSUMPTION DEVOLVE UPON THE WIFE ALONE
6.416	14.676	3575	3575-170457-0013	THE MORE SHE IS ENGAGED IN HER PROPER DUTIES THE LESS LEISURE WILL SHE HAVE FOR IT EVEN AS AN ACCOMPLISHMENT AND A RECREATION
15.650	28.690	1188	1188-133604-0021	IT WILL BE WHOLLY IMPOSSIBLE FOR YOU TO RETAIN THE TRANQUILLITY OF TEMPER AND FELICITY OF FAITH NECESSARY FOR NOBLE PURIST PAINTING UNLESS YOU ARE ACTIVELY ENGAGED IN PROMOTING THE FELICITY AND PEACE OF PRACTICAL LIFE
28.797	39.387	1188	1188-133604-0018	IN ALL EARLY GOTHIC ART INDEED YOU WILL FIND FAILURE OF THIS KIND ESPECIALLY DISTORTION AND RIGIDITY WHICH ARE IN MANY RESPECTS PAINFULLY TO BE COMPARED WITH THE SPLENDID REPOSE OF CLASSIC ART
39.389	53.669	1188	1188-133604-0011	THEY ARE BEYOND ALL OTHER WORKS THAT I KNOW EXISTING DEPENDENT FOR THEIR EFFECT ON LOW SUBDUED TONES THEIR FAVORITE CHOICE IN TIME OF DAY BEING EITHER DAWN OR TWILIGHT AND EVEN THEIR BRIGHTEST SUNSETS PRODUCED CHIEFLY OUT OF GRAY PAPER
48.241	58.941	7021	7021-85628-0001	HE MADE A BOW SO DEEP THAT HIS BACK CAME NEAR BREAKING AND HE WAS DUMBFOUNDED I CAN TELL YOU WHEN HE SAW IT WAS NOBODY BUT ANDERS

Appendix C

Model Architectures

C.1 Voice Activity Detection (VAD) Architecture

#	Layer Type	Output Shape	Param #
1	Conv2d	[-1, 32, 64, 101]	320
2	BatchNorm2d	[-1, 32, 64, 101]	64
3	MaxPool2d	[-1, 32, 32, 50]	0
4	Conv2d	[-1, 64, 32, 50]	18,496
5	BatchNorm2d	[-1, 64, 32, 50]	128
6	MaxPool2d	[-1, 64, 16, 25]	0
7	Dropout	[-1, 25600]	0
8	Linear	[-1, 64]	1,638,464
9	Linear	[-1, 2]	130
Total params			1,657,602
Input size (MB)			0.02
Forward/backward pass size (MB)			5.50
Params size (MB)			6.32
Estimated Total Size (MB)			11.85

Table C.1: Summary of the Voice Activity Detection model. Most of the computation is done in the fully-connected (linear) layers, as the VAD task doesn't require much hierarchical learning.

C.2 Speaker Counter Architecture

#	Layer Type	Output Shape	Param #
1	Conv2d	[-1, 8, 64, 101]	80
2	BatchNorm2d	[-1, 8, 64, 101]	16
3	MaxPool2d	[-1, 8, 32, 50]	0
4	Conv2d	[-1, 32, 32, 50]	2,336
5	BatchNorm2d	[-1, 32, 32, 50]	64
6	MaxPool2d	[-1, 32, 16, 25]	0
7	Conv2d	[-1, 128, 16, 25]	36,992
8	BatchNorm2d	[-1, 128, 16, 25]	256
9	MaxPool2d	[-1, 128, 8, 12]	0
10	Conv2d	[-1, 128, 8, 12]	147,584
11	BatchNorm2d	[-1, 128, 8, 12]	256
12	MaxPool2d	[-1, 128, 4, 6]	0
13	Dropout	[-1, 3072]	0
14	Linear	[-1, 128]	393,344
15	Linear	[-1, 5*]	645
Total params			581,573
Input size (MB)			0.02
Forward/backward pass size (MB)			2.88
Params size (MB)			2.22
Estimated Total Size (MB)			5.12

Table C.2: Summary of the Speaker Counting CNN. At 0.6M parameters the model is very compact relative to modern deep networks [46]. Despite being more architecturally complex, this model is smaller than the VAD model; the four convolutional layers provide a much higher compression rate. Most of the computation is done in the convolutional layers to maximise the extraction of hierarchical features. *The main model contains 5 classes, [0,1,2,3,4+]; an additional 'post-VAD' model was made with 4 classes, [1,2,3,4+]. The post-VAD model assumes speech in the audio. The post-VAD model wasn't significantly different to this model.

Appendix D

GitLab Structure

```
cxb1114/
|-- data/
|   |-- data.wav
|   `-- ...
|
|-- notebooks/
|   |-- DataWrangling.ipynb
|   |-- HPTuning.ipynb
|   |-- ModelTraining1_5Class.ipynb
|   |-- ModelTraining2_4Class.ipynb
|   |-- ModelTraining3_VAD.ipynb
|   `-- Visuals.ipynb
|
|-- src/
|   |-- app.py
|   |-- app_VAD.py
|   |-- data.py
|   |-- model.py
|   |-- predict.py
|   |
|   |-- utils/
|   |   |-- helpers.py
|   |   `-- SpectrogramExtractor.py
|   |
|   `-- model/
|       `-- models.pt
|       `-- ...
|
|-- thesis/
|   `-- dissertation.pdf
|
|-- demo.pptx
`-- requirements.txt
```

Figure D.1: The project repository structure.

Some notes on the repository contents: '**data/**' is empty because it's too large to store in a repository. '**notebooks/**' contains all results from practical work, while '**src/**' contains helper functions, model classes, configs and the real-time apps. '**src/model/**' contains the model weights.
A full description and set-up can be found in the readme.

D.1 Running the app

Assuming a venv is set up, use the following to run the app:

```
./.venv/Scripts/activate
streamlit run src/app_VAD.py
```