Chapter 1

Methodology

1.1 Data Collection

The primary dataset used for model training was LibriCSS [3], 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 1.1). 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 1.1: The types of audio present in LibriCSS [3].

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' examples. To address this imbalance the Room Impulse Response and Noise (RIR&N) database [4] 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 by the author to further expand the zero-speaker class. These recordings were collected under varied conditions to capture a broader range of acoustic characteristics (Table 1.2), resulting in approximately 125 minutes of new data. In total the combined corpus amounted to roughly 13 hours of audio for pre-processing (Table 1.3).

Table 1.2: A description of each audio file added by the author. 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 1.3: 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

1.2 Data Pre-Processing

Each recording was downmixed to a single channel (mono) to emulate the conditions of a single-microphone recording. While this increases the difficulty of the learning task, it enhances the model's applicability to real-world scenarios where multi-microphone arrays are typically unavailable. The LibriCSS corpus is originally captured using a seven-channel setup [3], providing rich spatial information that facilitates separation and recognition; whilst this setup is state-of-the-art, 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 provides sufficient temporal context for distinguishing between speakers, while also maintaining a manageable input size for efficient training and inference.

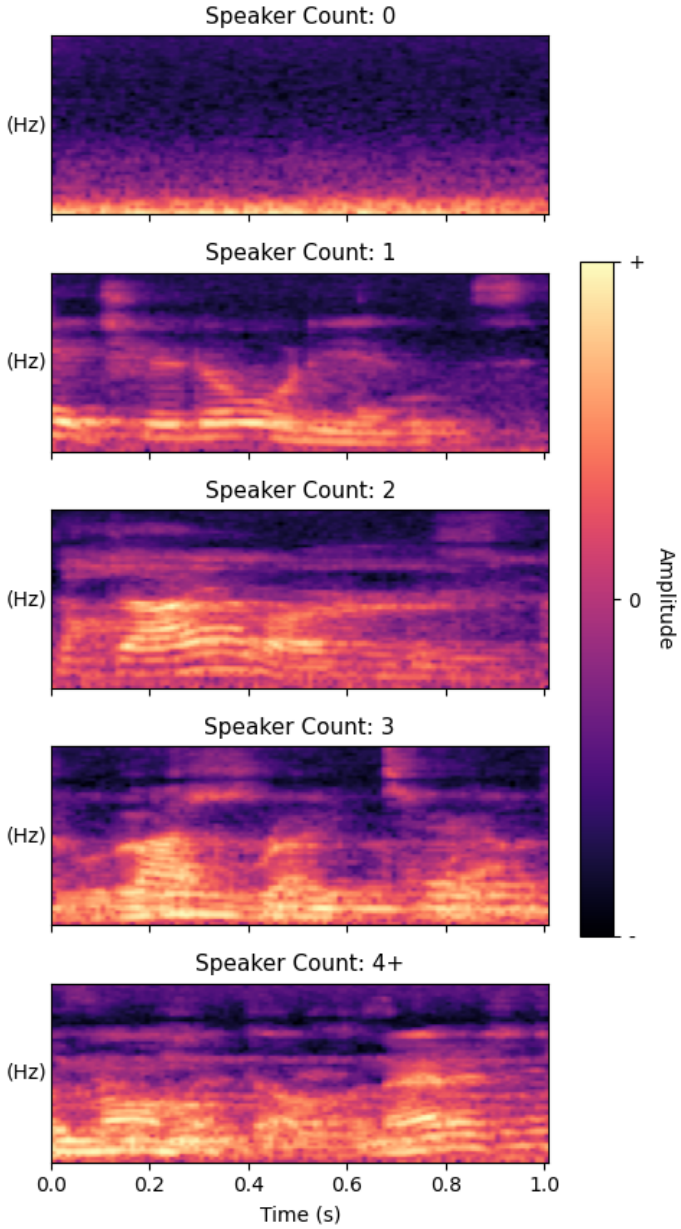

Figure 1.1: A series of spectrograms created with the `SpectrogramExtractor()` class on speech audio. Speaker count was found by manually cross-referencing the audio metadata. It's clear that activity increases with speaker count. With a low number of speakers ($n = 1, 2$) the spectral patterns of each 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 1.1).

After creating the spectrograms for each sample, the libricss metadata files (Appendix A) were algorithmically 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 also labelled with Speaker Count = 0. The resulting labelled dataset was formatted as a DataFrame (Table 1.4).

Table 1.4: An extract of the resulting dataset from using the LibriCSS metadata (Appendix A) 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']

Two primary limitations 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 generalizing to scenarios involving three or more speakers. Second, the dataset exhibited substantial class imbalance (Table 1.5), which could have biased model training toward the more frequently occurring classes.

Table 1.5: Number of samples per speaker count before augmentation. The distribution is highly imbalanced, which risks biasing the model towards majority classes.

Speaker Count	Samples
0	11719
1	29309
2	5251

1.2.1 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 pseduo-random; clips were seperated 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 1.6). 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 1.6: 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 1.7: 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

1.3 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.

1.3.1 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 1.2). 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 [6]. A detailed model architecture can be found in Appendix B.1.

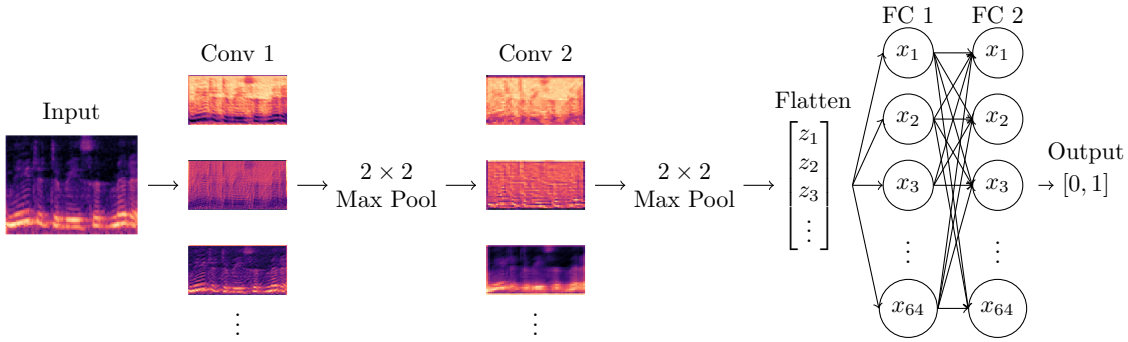

Figure 1.2: CNN architecture for the VAD model. An input spectrogram is fed through two convolutional layers with 3×3 kernels, applying 2×2 max pooling after each. This provides a series of 'feature maps' that represent the various features of the original image. These feature maps are flattened into 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.

1.3.2 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 1.3.2). 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 B.2.

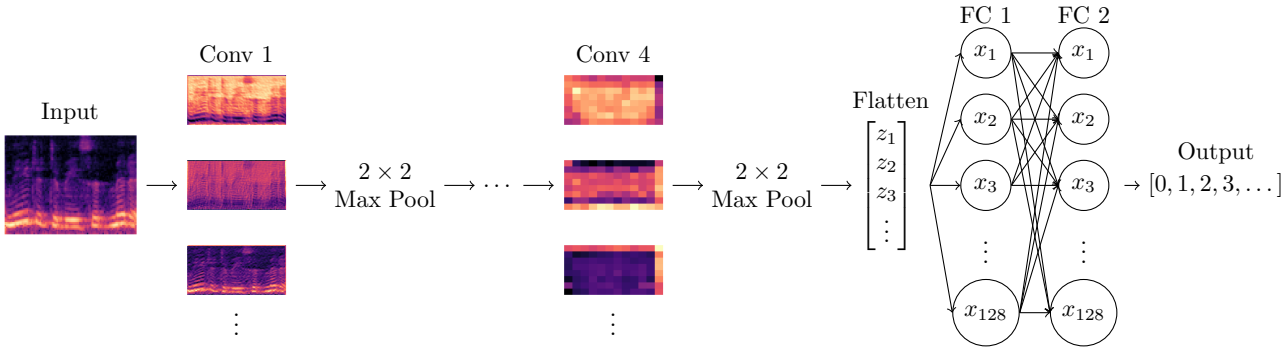

Figure 1.3: CNN architecture for the speaker count model. An input spectrogram is fed through four convolutional layers with 3×3 kernels, applying 2×2 max pooling after each. In comparison to Figure 1.2 the resulting feature maps are much smaller in spatial resolution but richer in representational detail. 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.

1.4 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 1.8). 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 1.8: 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 1.9) 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 (**CITE**). Cross-entropy was selected as the objective function because the tasks under study were classification problems: binary classification for VAD and multi-class classification for speaker count. Together, these choices provided stable convergence and compatibility with the Optuna optimisation framework.

Table 1.9: 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}

1.5 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 [5], 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 [2]. One-second clips were placed into a queue and processed continuously by a background thread, which applied preprocessing and then forwarded the results to the relevant model for inference. Predictions were stored in a rolling buffer, enabling both real-time display and short-term history tracking. This approach provided a practical way to demonstrate the responsiveness of the models under realistic streaming conditions. Because both applications shared the same architecture, only differing in the loaded model and output interpretation,

the methodology was consistent while remaining flexible. Streamlit was chosen to minimise development overhead and make the results accessible without requiring a separate deployment framework.

Chapter 2

Results

2.1 Hyperparameter study Results

best config that was applied to the model. Remember the model itself (the weights and config) is a result, not just its performance!

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

[3] 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.

[4] 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.

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

[6] 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.

Appendix A

LibriCSS Metadata

Table A.1: A metadata sample from LibriCSS [3]. 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 B

Model Architectures

B.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 B.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.

B.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 B.2: Summary of the Speaker Counting CNN. At 0.6M parameters the model is very compact relative to modern deep networks [6]. 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.