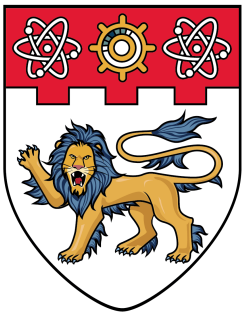

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SCSE23-0748
ROBUST VOICE ACTIVITY DETECTION USING DNN APPROACHES

Parashar Kshitij

Submitted in Partial Fulfilment of the Requirements for the Degree of Bachelor of Engineering (Computer Science)

Supervisor: Assoc. Prof. Chng Eng Siong

Examiner: Dr. Josephine Chong Leng Leng

School of Computer Science and Engineering

2023/2024

Abstract

Voice activity detection (VAD) is a pivotal component in various speech processing applications, playing a crucial role in tasks such as speech recognition, speaker diarization, and noise suppression. Recognizing its significance, this thesis delves into the exploration of advancements in single-channel VAD systems, leveraging the power of deep learning techniques. Through meticulous experimentation and analysis, we undertake comprehensive evaluations of three prominent VAD models: Pyannote, Silero, and MarbleNet, across a spectrum of conditions and scenarios. Our investigations encompass a nuanced examination of varying parameters such as chunk sizes, strides, and prediction thresholds, aiming to discern their nuanced impacts on model performance. From our findings, we discern Pyannote as the standout performer exhibiting superior accuracy compared to Silero by approximately 16.87% and MarbleNet by approximately 25.97% on the DIHARD III dataset. Consequently, we pivot our focus towards enhancing Pyannote's capabilities. In the process of enhancing, we looked into how different parameters affect the performance of Pyannote and trained models on varying chunk sizes and stride to deduce the same. With this, we were able to conclude that models trained on small chunk size and strides do not necessarily perform well during inference with small chunks and strides. Additionally, we delve into the realm of scalability and production readiness, exploring strategies facilitated by the Open Neural Network Exchange (ONNX) framework. These efforts provide important insights that can enhance the field of VAD, leading to the development of more robust and efficient voice activity detection systems capable of meeting the needs of modern speech processing applications.

Keywords: Deep Learning, Voice activity detection (VAD), Single-channel, Pyannote, Silero, MarbleNet, Chunk sizes, Strides, Prediction thresholds, Scalability, Robustness, Open Neural Network Exchange (ONNX).

Acknowledgement

I would like to express my heartfelt appreciation to my supervisor, Prof. Chng Eng Siong, for their invaluable guidance and unwavering support throughout the progression of my project. Working under their mentorship has been a privilege, and I am truly grateful for their continuous encouragement and constructive feedback, which have been instrumental in shaping my research journey. Their expertise and dedication have inspired me to strive for excellence, and I am fortunate to have had the opportunity to learn from them.

Furthermore, I am grateful for the opportunity to contribute to the project, as it has allowed me to delve deeper into the fascinating realm of speech processing and deep neural networks. The experience gained has been invaluable, and I am thankful for the chance to expand my knowledge and skills in this field.

I would also like to extend my sincere gratitude to all the members of the Speech Lab at Nanyang Technological University, Singapore, for their valuable contributions and insights. Collaborating with such a talented and dedicated group of researchers has been an enriching experience, and I am appreciative of the support and camaraderie extended to me throughout this journey. Their expertise and enthusiasm have significantly contributed to the success of the project, and I am grateful for the opportunity to work alongside them.

Table of Contents

Nomenclature

BLSTM	Bidirectional Long Short Term Memory
CNN	Convolutional Neural Networks
DNN	Deep Neural Network
FAR	False Alarm Rate
MDR	Missed Detection Rate
MLP	Multi Layer Perceptron
RNN	Recurrent Neural Network
RTF	Real Time Factor
RTTM	Rich Text Time Marked
VAD	Voice Activity Detection

Lists of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

Speech, a fundamental form of human communication, is a complex and dynamic interplay of linguistic, physical, and psychological elements. In its raw form, speech is a continuous acoustic signal with variations in frequency, amplitude, and timing. In an era characterized by increasing use of voice-driven technologies, the accurate detection of human speech within the ambient soundscape takes on pivotal significance. According to Statista, the number of digital voice assistants is projected to grow to 8.4 billion by 2024 [1]. Voice Activity Detection (VAD) is a fundamental task in speech processing, with applications in various fields such as real-time speech translation, speech recognition, voice-controlled devices, and deceptive speech detection. It is a binary classifier that detects the presence of human speech in audio. Speaker diarization is the process of splitting up an input audio stream that often includes several speakers into homogeneous segments according to the speaker identity. This helps to identify different speakers' turns in a conversation. Figure 1-1 describes the key components in a typical speaker diarization pipeline. Figure 1-2 shows how an audio signal is broken down into speech and non-speech regions after being passed through a VAD.

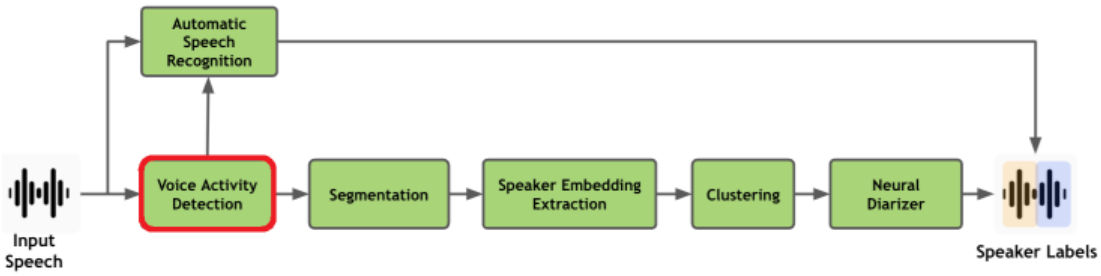

Figure 1-1: VAD in a typical speaker diarization pipeline

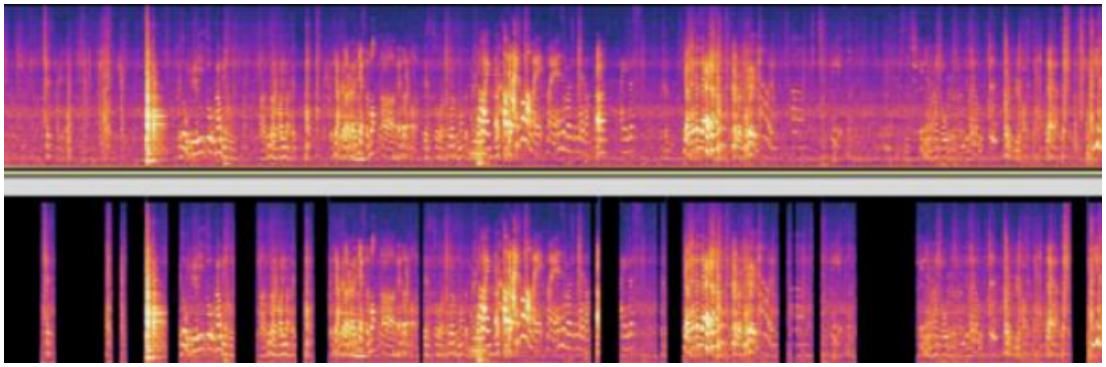

Figure 1-2: Audio signal after being passed through a VAD

1.2 Motivation

Despite its significance, voice activity detection faces several significant challenges that impact its effectiveness and efficiency. These challenges occur from the inherent complexity of audio signals and the variability of human speech [2]. Understanding these challenges is crucial for advancing the field and developing more robust VAD systems. Some of the primary challenges include:

1. **Background Noise and Acoustic Variability:** Background noise and varying acoustic conditions remain significant challenges for single-speaker VAD systems. Background noise, along with echoes or distortions, can obscure speech signals, making it difficult for VAD systems to distinguish between speech and noise accurately.

2. **Segmentation Errors:** Dividing the audio stream into segments containing speech or silence accurately is crucial for VAD systems. Errors in segmentation, such as incorrectly classifying periods of silence as speech or vice versa, can lead to inaccuracies in the detection of speech presence, impacting the overall performance of the system.

3. **Scalability and Computational Efficiency:** Ensuring that VAD systems are scalable and computationally efficient is essential, even in single-speaker scenarios. As the duration of audio recordings increases, the computational resources required for processing also escalate. Therefore, developing efficient algorithms capable of handling longer audio streams is critical.

4. **Domain Adaptation:** Single-speaker VAD systems may struggle to maintain high performance when applied to audio data from different domains or

recording conditions. Adapting the system to different environments, recording qualities, or speaker characteristics poses a significant challenge, necessitating robust domain adaptation techniques.

We aim to address these challenges to enhance the accuracy and efficiency of single-speaker VAD systems, making them more reliable for applications requiring the detection of speech presence or absence in various real-world scenarios.

1.3 Objectives

The primary objective of this research project is to investigate advancements in single-channel voice activity detection (VAD) systems leveraging deep learning techniques. Specifically, the project aims to achieve the following objectives:

1. **Comprehensive Evaluation of VAD Models:** Conduct a thorough evaluation of three prominent VAD models, namely Pyannote, Silero, and MarbleNet, under various conditions and scenarios. This includes assessing their performance metrics, such as accuracy, false alarm rate, and receiver operating characteristic (ROC) curve analysis.
2. **Exploration of Model Parameters:** Investigate the impact of different model parameters, such as chunk sizes, strides, and prediction thresholds, on the performance of VAD models. This involves varying these parameters systematically to understand their influence on model accuracy and robustness.
3. **Identification of Top-Performing Model:** Identify the top-performing VAD model based on experimental results and analysis. This includes determining which model exhibits the highest accuracy and efficiency across different experimental conditions.
4. **Enhancement of Model Capabilities:** Explore strategies for enhancing the capabilities of the top-performing VAD model. This may involve optimizing model architecture, fine-tuning parameters, or integrating additional features.
5. **Evaluation of Scalability and Production Readiness:** Assess the scalability and production readiness of the selected VAD model. This includes exploring methods for optimizing inference times, resource utilization, and integration into real-world applications.

By achieving these objectives, this research project aims to contribute valuable insights to the field of voice activity detection, paving the way for the development of more robust and efficient VAD systems.

1.4 Scope

The focus of this thesis is directed towards single-channel far-field voice activity detection tasks. This allows for a deep exploration of the nuances and intricacies involved in detecting voice activity specifically in far-field audio recordings, which are prevalent in various real-world scenarios. By narrowing the scope to this particular domain, the thesis aims to provide a comprehensive understanding of the challenges and solutions unique to single-channel far-field voice activity detection.

1.5 Organisation of Report

This report is divided into 5 chapters and the overview of each chapter is provided below:

- **Chapter 1:** Provides an overview of the research topic, outlines objectives, and sets the stage for subsequent chapters.
- **Chapter 2:** Conducts a literature review, identifies gaps, and lays the groundwork for further investigations.
- **Chapter 3:** Describes the methodology employed for evaluation, ensuring transparency and reproducibility.
- **Chapter 4:** Presents experimental findings, analyzing the performance of VAD models under various conditions.
- **Chapter 5:** Summarizes key conclusions, highlights contributions, and suggests future research directions.

Chapter 2 Literature Review and Relevant Theory

This chapter provides a comprehensive overview of Voice Activity Detection (VAD) systems, encompassing foundational concepts, deep neural network (DNN) architectures, diverse VAD approaches, evaluation metrics, and the Open Neural Network Exchange (ONNX) framework. It explores fundamental audio concepts, delves into DNN architectures such as MLP, CNN, BLSTM, and analyzes various VAD methodologies. Additionally, it discusses evaluation metrics like RTF, Accuracy, MDR, FAR, and explores the role of ONNX in enhancing scalability and production readiness of VAD systems.

2.1 Audio Concepts

This section explains the various audio signal related concepts and describes the methods with which audio files are processed before using it as an input to a neural-network based model.

2.1.1 Far-field audio

Far-field audio [3] refers to sound captured from a distance away from the audio source or the microphone. In far-field scenarios, the microphone is typically placed several feet away from the speaker or sound source, such as in conference rooms, living rooms, or public spaces. Far-field audio capture presents challenges due to factors like background noise, reverberation, and signal degradation over distance.

2.1.2 Near-field audio

Near-field audio [4], on the other hand, refers to sound captured from a close distance to the audio source or the microphone. In near-field scenarios, the microphone is placed relatively close to the speaker or sound source, such as in headset microphones, smartphone microphones, or studio recording setups. Near-field audio capture often results in clearer and more direct sound recordings, with fewer environmental interferences compared to far-field audio.

2.1.3 Single-channel audio

Single-channel audio [3] refers to audio recordings or signals captured using a single microphone or audio input channel. In single-channel audio setups, all audio information is captured and processed through a single audio stream. Single-channel audio recordings are common in various applications, including voice calls, speech recognition, and audio recording devices with a single microphone.

2.1.4 Multi-channel audio

Multi-channel audio [3], on the other hand, refers to audio recordings or signals captured using multiple microphones or audio input channels. In multi-channel audio setups, audio information is captured and processed through multiple audio streams simultaneously. Multi-channel audio recordings are commonly used in applications such as surround sound systems, spatial audio recording, and noise cancellation systems, where capturing audio from multiple directions or sources is necessary for immersive or enhanced audio experiences.

2.1.5 File formats

The audio file formats used for tasks like voice activity detection are WAV and FLAC. Table 2-1 provides a brief overview of the features provided by these formats.

Feature	WAV	FLAC
File Size	Generally larger due to uncompressed format.	Smaller due to lossless compression.
Compression	Uncompressed, resulting in high-quality audio but larger file sizes.	Lossless compression, which maintains audio quality while reducing file size.
Quality	High-quality, as it preserves the original audio data without any loss.	High-quality, as it's a lossless compression format.
Editing	Suitable for editing as it preserves all audio data.	Suitable for editing as it retains the original audio quality.

Table 2-1: Comparison of WAV and FLAC audio file formats

2.1.6 Frame

A frame is a single sample of an audio file. The length of a frame is typically measured in samples, and it is related to the sampling rate of the audio signal. For example, if the sampling rate is 16000 Hz (samples per second) and the frame length is set to 512 samples, then the duration of each frame would be 512/16000 = 0.032 seconds or 32 milliseconds. Figure 2-1 describes how frames are sampled from an audio signal.

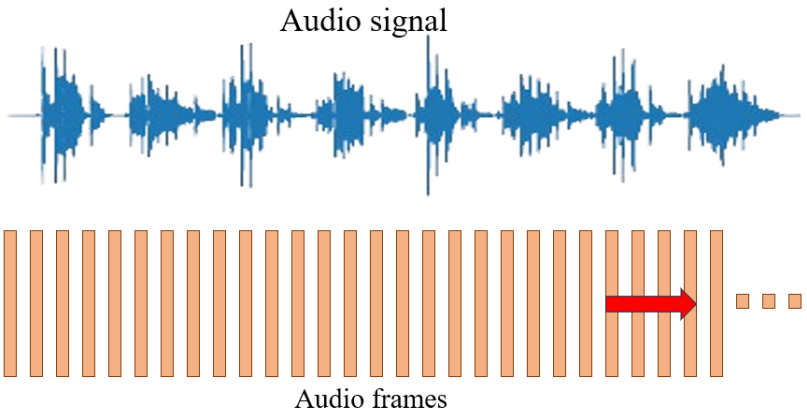

Figure 2-1: Frames sampled from an audio signal

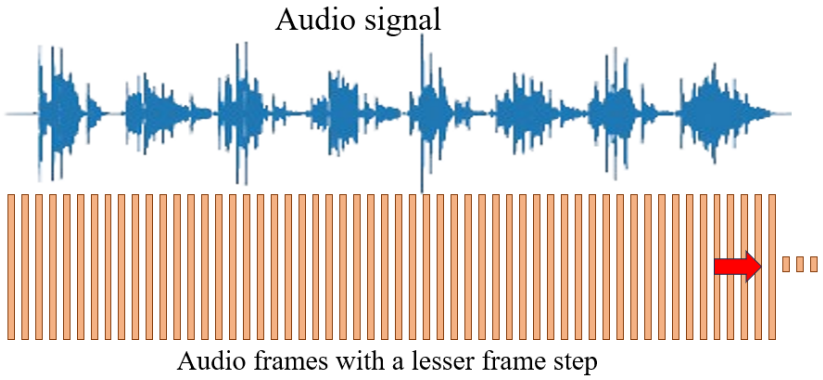

Figure 2-2: Frames sampled from an audio signal with smaller frame step

2.1.7 Frame Step

The frame step, also known as the frame shift or frame increment, refers to the number of samples by which the analysis window (or frame) is shifted forward in time to sample the next frame of the audio signal. In other words, it determines the overlap or spacing between consecutive frames during analysis. Figure 2-2 depicts an example of how frames are sampled with a smaller frame step as compared to Figure 2-1.

2.1.8 Chunk

A chunk, also known as an analysis window, is a larger segment of audio data that encompasses multiple consecutive frames. It represents a longer time duration over which various audio processing operations or analyses can be performed. For example, if the sampling rate is 16000 Hz (samples per second) and the chunk size is set to 1 second, then a chunk would contain 16000 frames, assuming each frame consists of a single sample. However, in practice, frames often contain multiple samples (e.g., 512 samples per frame), and the number of frames within a chunk would be adjusted accordingly. Figure 2-3 depicts an example of a chunk.

2.1.9 Stride

It refers to the step size or the number of samples by which the analysis window (or chunk) is advanced or shifted for processing the next segment of the audio signal. It determines the overlap or spacing between consecutive analysis windows, similar to the stride concept in deep learning for convolutional operations. The choice of stride

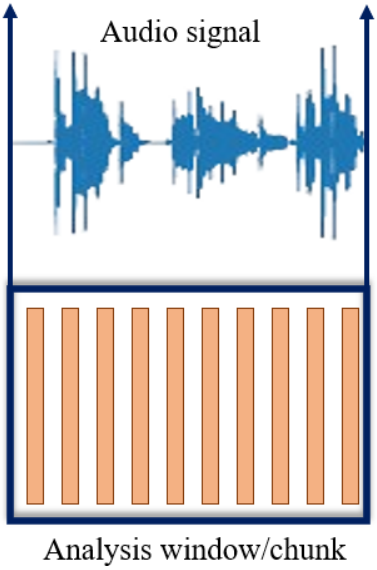

Figure 2-3: An example of a chunk sampled from an audio file

value depends on the specific audio processing task, the desired trade-off between time and frequency resolution, and the computational resources available. It is common to experiment with different stride values and analyze their impact on the audio processing results to find the optimal configuration. Figure 2-4 depicts how subsequent chunks are created using stride as the incremental value.

2.2 DNN Concepts

Deep Neural Network (DNN) frameworks have shown promise in addressing speech processing challenges, leveraging their ability to learn complex patterns [5]. DNNs, including Convolutional Neural Networks (CNN), Multi-Layer Perceptrons (MLP), and Long Short-Term Memory (LSTM) networks, are commonly employed for Voice Activity Detection (VAD) systems [6]. Within the supervised model-based approach, CNN-based VAD systems have achieved over 99% accuracy on datasets like CENSREC-1-C [7]. Moreover, joint VAD and speaker localization systems, utilizing DNN models, have significantly reduced detection errors [8].

However, the absence of standardized testing methodologies poses a significant challenge, hindering fair comparisons across implementations [9]. Generally, the performance of VAD models are assessed in terms of metrics like Receiver Operating Characteristic (ROC) curves, missed detection and false alarm rate. The ROC curve directly depends on the data and does not require any assumptions. To express this

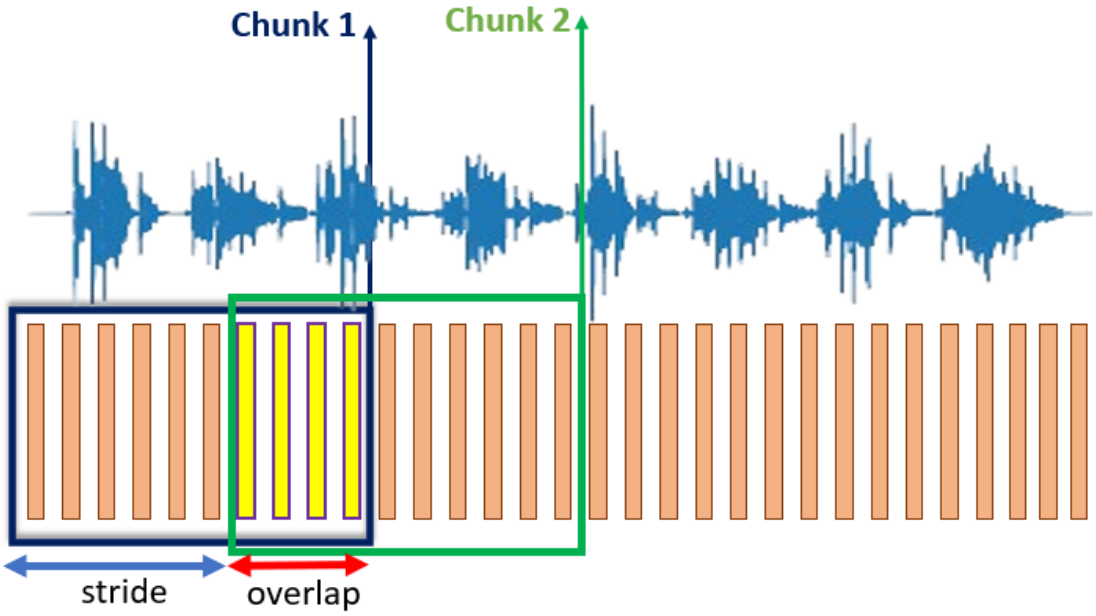

Figure 2-4: Chunks with stride as increments

metric as a single value, Area Under the (ROC) Curve (AUC) is calculated. As ROC and AUC are based on values which are averaged over time, they cannot reflect performance with respect to specific time intervals. Freeman et al. [10] introduced time-selective evaluation methods to address this limitation, distinguishing between different types of errors. While accuracy remains vital, the efficiency and speed of VAD systems are equally crucial for real-world applications. To address this concern, this report is focused on testing and improving the performance of various open source VAD models to better predict the presence of voice on speech datasets like Alimeeting and DIHARD III.

The integration of LSTM architectures in speech processing and CNN architectures in Computer Vision has reshaped research in speaker diarization [2]. This transformation, starting around 2017, witnessed a shift towards compact, neural network-driven approaches. Neural networks, including MLPs, CNNs, and Bidirectional LSTMs (BLSTMs), play a pivotal role in VAD development.

2.2.1 Multi Layer Perceptron (MLP)

MLP is one of the earliest and most well-known kind of artificial neural network. In a MLP network, each node uses a tanh activation function over the weighted sum of its inputs. The nodes are then arranged in layers with feed forward connections from one

layer to the next. Furthermore, stochastic gradient descent with error back-propagation algorithm is used for the supervised learning of this network [11].

2.2.2 Convolutional Neural Network (CNN)

CNN is a feed-forward neural network [12] generally comprising of three types of layers: neuron layers and convolutional layers and pooling layers. The convolutional layer is for performing a mathematical operation known as convolution between a multi-dimensional input and a fixed-size kernel. Subsequently, a non-linear operation is applied elementwise. Typically, the kernels are relatively small compared to the input, enabling convolutional neural networks (CNNs) to handle sizable inputs with a minimal number of learnable parameters. Lastly, a pooling layer is commonly employed to decrease the dimensions of the feature map. Figure 2-5 provides an example of a CNN-based VAD model with details about the convolutional layers [2].

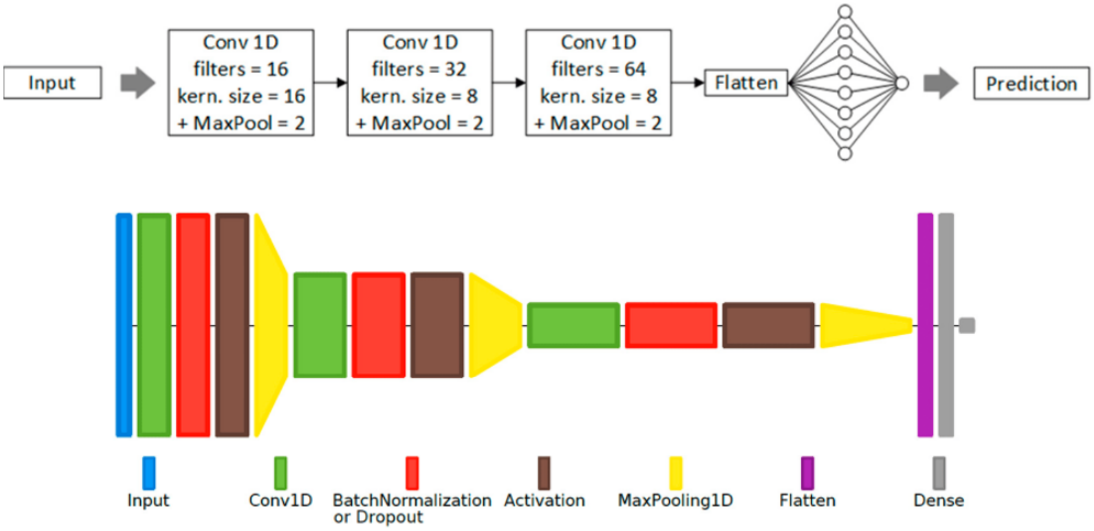

Figure 2-5: Example of CNN-based VAD model [2]

2.2.3 Bidirectional Long Short Term Memory (BLSTM)

The BLSTM is a special type of RNN in which the hidden units are replaced by long short term memory (LSTM) units [13]. Each LSTM unit consists of a memory cell and three gates: forget gate, input gate and output gate. The gates help control duration for which the information is stored in the memory cell. In this way, the network can remember long-range temporal information. Furthermore, it can be made

bidirectional to access context from both temporal directions [11]. Figure 2-6 describes the architecture of a VAD model using BLSTM.

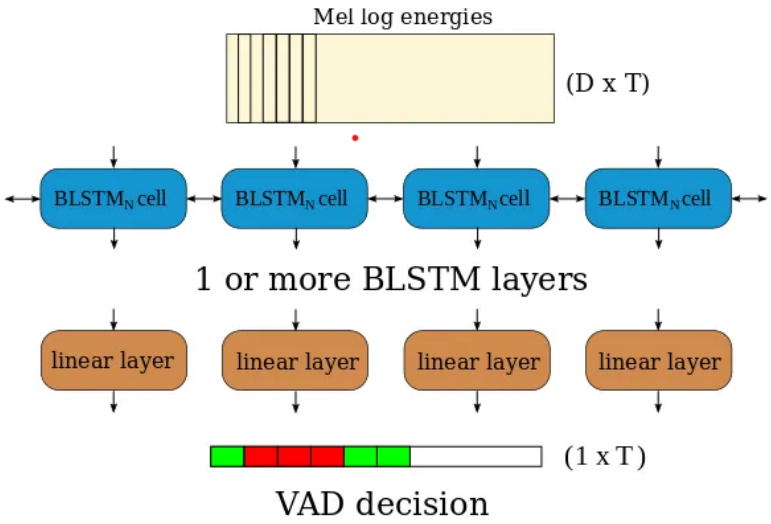

Figure 2-6: BLSTM-based architecture of ViVoVAD [14]

2.2.4 Binary Cross Entropy (BCE) Loss Function

The Binary Cross Entropy (BCE) loss function is commonly used for binary classification tasks. It measures the dissimilarity between the true binary labels and the predicted probabilities output by the model.

For a single sample, the BCE loss is calculated as follows:

$$\text{BCE}(p,y) = -\left(y\log(p) + (1-y)\log(1-p)\right)$$

where p is the predicted probability, y is the true label (1 for positive, 0 for negative) and log denotes the natural logarithm.

Similarly, for N samples, the BCE loss can be represented as follows:

$$BCE(p,q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i)\cdot\log(1-p(y_i))$$

In this formula, $BCE(p,q))$ represents the BCE loss, y_i is the true binary label (0 or 1) and $p(y_i)$ is the predicted probability.

The BCE loss is averaged over all samples in the batch to compute the overall loss for that batch. The logarithmic terms ensure that the loss is higher when predictions are far from the actual labels.The implementation for this in Python code is shown below:

```
loss = torch.nn.BCELoss(reduction='mean')(preds, batch["label"])
```

- **torch.nn.BCELoss:** This is a class provided by *PyTorch* from the *torch.nn* module, which represents the Binary Cross Entropy loss function. It's commonly used for binary classification tasks.
- **reduction='mean':** This parameter specifies how the losses for individual elements of the batch are aggregated to calculate the overall loss. In this case, 'mean' indicates that the losses will be averaged across all elements of the batch.
- **(preds, batch["label"]):** This part of the code computes the BCE loss by comparing the model predictions (preds) with the ground truth labels (batch["label"]). The BCE loss is calculated element-wise for each prediction-label pair in the batch.

2.3 VAD Approaches

At its core, VAD is a signal processing technique aimed at identifying segments of audio that contain human speech, distinguishing them from periods of silence or background noise. The objective is to efficiently partition an audio stream into regions of speech (active) and non-speech (inactive) based on acoustic characteristics of the audio. The process of VAD typically involves analyzing short frames of the audio signal, often spanning 10 to 30 milliseconds. For each frame, certain acoustic features are computed, commonly including Mel Frequency Cepstral Coefficients (MFCCs), energy, zero-crossing rate, and spectral characteristics. These features serve as input to machine learning models, signal processing algorithms, or a combination of both. There are mainly 2 types of approaches in the development of VADs:

2.3.1 Online VAD

Often referred to as streaming VAD, it is an approach that real-time analysis and segmentation of an ongoing audio stream. It operates by dynamically analyzing the audio stream in small frame typically spanning 10s of milliseconds and instantly categorizes each frame has speech or non-speech on specific acoustic characteristics. The key advantage of online VADs lies in its ability to provide immediate results making it highly responsive and suitable for applications that require quick speech detection and segmentation.

Online VAD has a significant advantage in its real-time processing capability. The system can promptly classify each frame, providing instantaneous results, which is crucial for applications like live streaming, telephony, voice-controlled systems, or real-time transcription services. This low latency ensures timely decision-making based on the detected speech segments. Moreover, online VAD is dynamically adaptable to changing acoustic conditions. It can adjust to variations in the environment, making it versatile for use in varying and noisy settings where speech characteristics may rapidly change.

However, real-time processing characteristic demands significant computational resources, which can limit the scalability of the system and may require efficient algorithms and robust hardware for optimal performance. Additionally, online VAD may be susceptible to false positives, especially in the presence of background noise. The immediate analysis of small audio frames can lead to false identifications, affecting the accuracy of speech detection and potentially generating false alarms for non-speech segments.

2.3.2 Offline VAD

Offline Voice Activity Detection (VAD) is an alternative approach that involves batch processing of pre-recorded audio data, where the entire audio recording is available for analysis. Instead of analyzing the audio in real-time frames, offline VAD processes the audio in larger chunks, allowing for a more comprehensive view of the entire recording. This approach can provide a global view of the audio, aiding in optimized algorithm selection and a better understanding of the overall acoustic characteristics [15].

Offline Voice Activity Detection (VAD) has its advantages. Analyzing the complete audio recording enables a comprehensive understanding of the acoustic properties and characteristics present throughout the audio, leading to a more accurate classification of speech and non-speech segments. Moreover, since offline VAD does not require real-time processing, it proves to be more resource-efficient and can be implemented on less powerful hardware.

However, offline VAD has its own set of challenges. One significant drawback is the increased latency it introduces. The processing is done after the audio recording is complete, resulting in higher processing time, making it unsuitable for applications that require real-time analysis or immediate response.

2.4 Silero

Silero VAD [16] is a highly efficient speech processing system designed to identify voice activity within an audio stream. One of its notable properties is the flexibility in accommodating various audio input characteristics. Silero VAD can operate on audio streams with different sampling rates, making it adaptable to a wide range of

audio sources. It can effectively process audio streams sampled at 8kHz and 16kHz, showcasing its versatility and applicability across diverse platforms and devices.

In terms of chunk size, Silero VAD processes audio data in segments or chunks, where each chunk typically consists of 30 ms of audio. The small chunk size allows for efficient processing and analysis of the audio stream in manageable parts, enabling real-time voice activity detection. The utilization of appropriate chunk sizes is crucial for balancing computational efficiency and the accuracy of VAD. Silero VAD employs a neural network-based architecture to classify these audio chunks into two categories: speech or non-speech. The neural network is trained to recognize acoustic patterns and features that are indicative of speech presence. This approach enables Silero VAD to make precise predictions for each chunk, determining whether it contains speech or not.

The adaptability to different sampling rates and the appropriate chunk size selection are vital properties that enhance Silero VAD's utility and effectiveness in diverse applications. Whether processing lower or higher sampling rate audio streams, and irrespective of the chunk size, Silero VAD can deliver reliable voice activity detection, making it a valuable tool in speech-related technologies and applications.

2.5 Pyannote

Pyannote VAD [17] is a state-of-the-art VAD algorithm based on DNNs. The architecture is designed to extract features from the audio that are relevant to predicting the presence of speech. Figure 2-7 describes the detailed architecture of the model. The Pyannote VAD consists of SincNet layer [18] which is a CNN layer that encourages the first convolutional layer to find more meaningful filters . It is based on parameterized sinc functions, which implements band-pass filters.

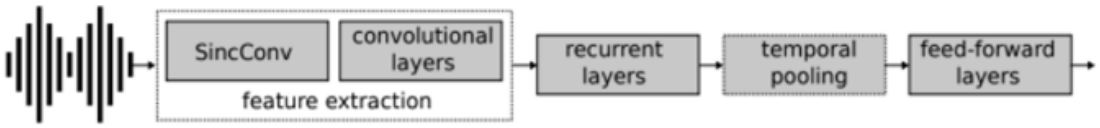

Figure 2-7: Architecture of Pyannote VAD

After feature extraction, the convolutional layers are followed by one or more LSTM layers. Long Short-Term Memory (LSTM) layers are a type of recurrent neural network (RNN) layer that are well-suited for modeling temporal sequences, such as audio signals. The LSTM layers learn to extract long-range dependencies from the audio signal, which is important for accurate voice activity detection. The final layer in the Pyannote VAD is a fully connected layer that predicts the probability of speech for each audio frame. The Pyannote VAD DNN is trained using a supervised learning approach. The training data consists of audio recordings that are labeled as speech or non-speech. The DNN is trained to minimize the cross-entropy loss between the predicted probabilities and the true labels. Once the DNN is trained, it can be used to predict the probability of speech for new audio recordings. The DNN is applied to the audio recording frame-by-frame, and the predicted probabilities are used to generate a sequence of speech flags. A speech flag is a binary value indicating whether the corresponding audio frame contains speech. The results of Pyannote VAD are often used as a baseline for evaluating other models. Table 2-2 [17] shows the results of Pyannote VAD on Dihard III Eval set.

Dataset	**Missed Detection**	**False Alarm**
DIHARD III	3.3	3.9

Table 2-2: Benchmark Pyannote VAD results [17]

2.6 MarbleNet

MarbleNet VAD [19] is a voice activity detection model developed by NVIDIA NeMo. It is a DNN model trained on a large corpus of labelled speech and non-speech data. MarbleNet VAD is designed to be accurate, robust, and efficient, making it suitable for a variety of applications, such as speech recognition, speaker identification, and audio compression. Figure 2-8 describes the architecture of MarbleNet VAD [19]. It is based on a 1D time-channel separable convolutional neural network (CNN) architecture. The CNN consists of a stack of convolutional layers, followed by a fully connected layer. The convolutional layers extract features from the audio signal, while the fully connected layer predicts the probability of speech. The 1D time-channel separable CNN architecture is a key design feature of MarbleNet VAD. It allows the model to learn features from the audio signal in both the time and frequency domains. Additionally, the 1D time-channel separable CNN architecture is more efficient than traditional CNN architectures, making MarbleNet VAD suitable for real-time applications. The VAD is trained using a supervised learning approach. The training data consists of audio recordings that are labeled as speech or non-speech. The DNN model is trained to minimize the cross-entropy loss between the predicted probabilities and the true labels.

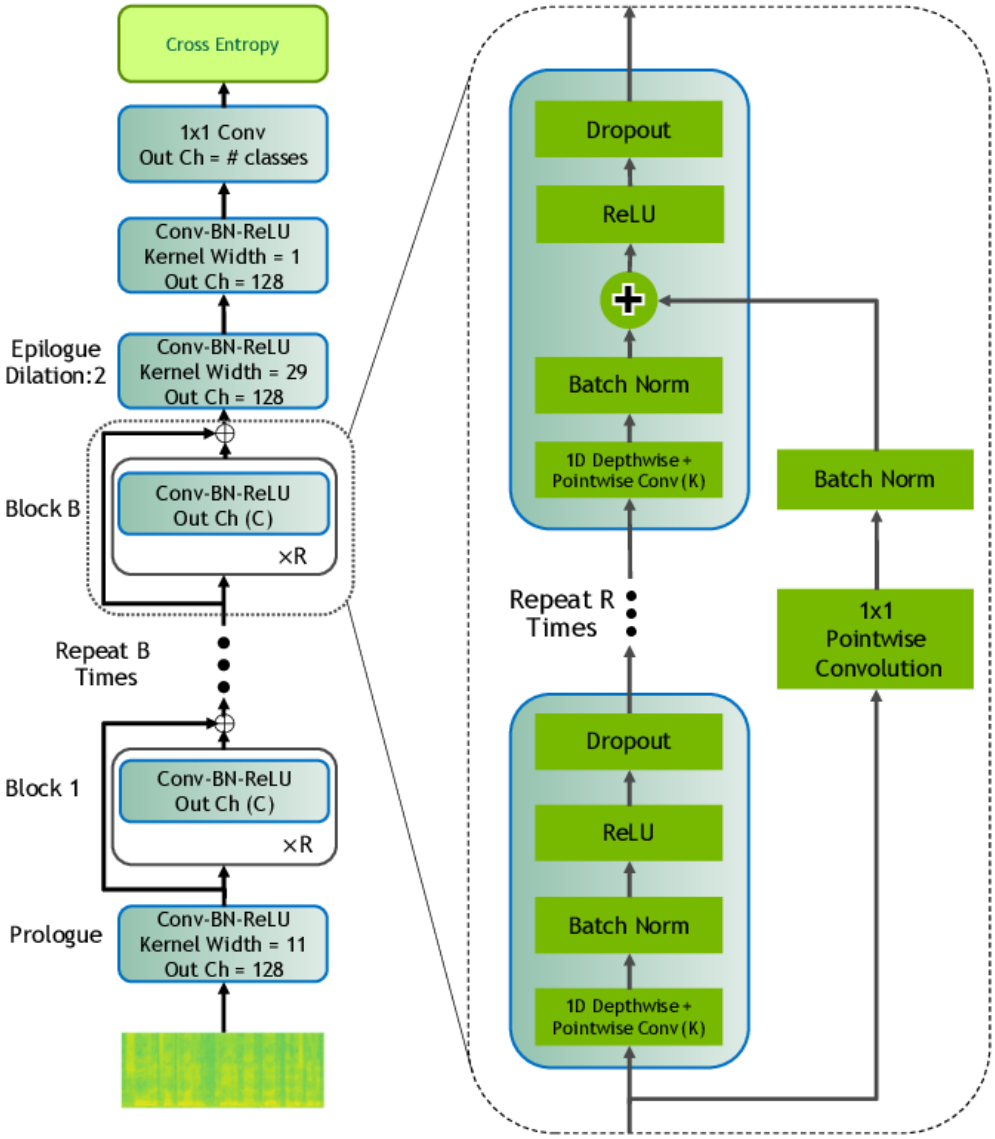

Figure 2-8: Architecture of MarbleNet VAD [19]

2.7 Evaluation Metrics

Yadav et al. contend that the absence of standardized testing methodologies for VAD models poses a significant challenge, hindering the ability to conduct equitable comparisons across diverse implementations utilizing varying sets of testing data [9]. Taking these points into consideration, the experiments described in this thesis utilize all the evaluation metrics mentioned below:

2.7.1 Real Time Factor (RTF)

The Real Time Factor (RTF) in Voice Activity Detection (VAD) refers to the speed and efficiency with which the VAD system can process and analyze an ongoing audio stream, providing instantaneous results for speech detection. In real-time applications, such as live speech processing, telephony, or voice-controlled systems, it is crucial for the VAD system to quickly identify speech segments as they occur. The real-time factor is measured in relation to the actual duration of the audio being processed. A VAD system with a real-time factor of 1 processes the audio in the same amount of time as its duration (e.g., processing a 10-second audio recording in 10 seconds). A lower real-time factor implies faster processing, which is advantageous in time-sensitive applications. Achieving a real-time VAD system is essential for ensuring smooth and seamless integration into various applications where prompt and accurate speech detection is paramount. It can be expressed by the formula:

$$RTF = \frac{Time\ taken\ to\ process\ audio\ file}{Duration\ of\ audio\ file}$$

2.7.2 Accuracy

Accuracy, in the context of Voice Activity Detection (VAD), is a fundamental evaluation metric representing the overall correctness of the VAD system's classifications. It calculates the ratio of correctly classified frames (both speech and non-speech) to the total number of frames in the audio recording. A higher accuracy signifies a VAD system's proficiency in accurately identifying speech and non-speech segments. However, it's important to note that accuracy can be misleading, especially in imbalanced datasets where one class (e.g., non-speech) dominates the other (e.g., speech). In such scenarios, a high accuracy might be achieved by the model correctly identifying the dominant class but failing to accurately detect the minority

class. Hence, while accuracy provides a general overview of system performance, it should be interpreted alongside other metrics to better understand the VAD model's effectiveness, particularly in scenarios where the classes are imbalanced. It can be expressed by the formula:

$$Accuracy = \frac{No.\ of\ correct\ predictions}{Total\ no.\ of\ predictions} = \frac{TP + TN}{TP + FP + FN + TN}$$

where TP = True Positives, TN = True Negatives, FP = False Positives and FN = False Negatives.

2.7.3 Missed Detection Rate (MDR)

Missed Detection, also known as False Negative (FN), is a significant evaluation metric in Voice Activity Detection (VAD) that gauges the system's capability to accurately identify speech segments. It represents instances where the VAD system fails to detect actual speech and incorrectly labels them as non-speech or silence. The ratio of false negatives to the total actual speech frames constitutes the missed detection rate. A higher missed detection rate indicates a higher tendency of the VAD system to overlook genuine speech, which can be problematic, especially in applications where all speech segments need to be accurately captured. Lowering the missed detection rate is essential to ensure that speech is not erroneously classified as non-speech, enabling the VAD system to effectively serve applications like automatic speech recognition (ASR) or speaker diarization where precise speech segmentation is vital for accurate downstream processing. Balancing missed detection with other evaluation metrics provides a comprehensive assessment of the VAD system's performance and aids in its refinement for optimal accuracy.

2.7.4 False Alarm Rate (FAR)

The False Alarm Rate (FAR) is a crucial evaluation metric in Voice Activity Detection (VAD) that measures the rate at which the VAD system incorrectly identifies non-speech segments as speech. It is computed by taking the ratio of false positive frames (frames incorrectly classified as speech) to the total actual non-speech frames. A lower FAR indicates a VAD system's capability to minimize false alarms, which is essential for applications requiring precise speech detection in the presence of various background noises. A high FAR signifies a higher tendency of the VAD system to erroneously trigger on non-speech regions, which can be particularly problematic in scenarios where accurate differentiation between speech and silence is critical. Balancing a low FAR with other metrics like recall and precision is crucial for optimizing the VAD system to achieve an accurate and reliable speech segmentation, avoiding unnecessary triggers on non-speech regions. If TN = True Negatives and FP = False Positives, FAR can be expressed by the formula:

$$FAR = False\ Positive\ Rate = \frac{FP}{FP + TN}$$

2.7.5 ROC-AUC

The Receiver Operating Characteristic (ROC) curve, along with its associated Area Under the Curve (AUC), constitutes an essential evaluation tool for assessing the performance of Voice Activity Detection (VAD). The ROC curve visually displays the trade-off between the true positive rate (sensitivity) and false positive rate (1-specificity) as the VAD system's discrimination threshold varies. It provides valuable insights into the system's ability to distinguish between speech and non-speech at different threshold settings. The AUC quantifies the overall performance of the VAD system by representing the area under the ROC curve. A higher AUC indicates a

more effective VAD system, proficient in distinguishing between speech and non-speech across various operating points. Utilizing ROC-AUC analysis allows for a comprehensive understanding of the VAD system's capabilities, enabling fine-tuning and optimization to achieve the desired balance between true positives and false positives in speech detection applications. In this context, a higher ROC-AUC value is generally regarded as positive, signifying a stronger ability of the VAD system to accurately discriminate between speech and non-speech, thus highlighting its reliability and efficiency in speech segmentation tasks.

2.8 Open Neural Network Exchange (ONNX)

The Open Neural Network Exchange (ONNX) [20] is an open-source format for representing deep learning models, developed by a collaboration among major tech companies such as Microsoft, Amazon, and Facebook. ONNX aims to provide an interoperable way of representing models across different frameworks, enabling model portability and reusability. It defines a common set of operators, data types, and conventions for representing deep learning models as computation graphs, where nodes represent operators (e.g., convolutions, activations), and edges represent tensors (input/output data). Figure 2-9 describes the flexibility provided by ONNX to run AI models on different platforms

ONNX enables efficient model deployment and inference by allowing models to be converted to optimized formats or even compiled to machine code for specific hardware targets. This conversion process can leverage hardware-specific optimizations, resulting in faster inference times and reduced resource consumption.

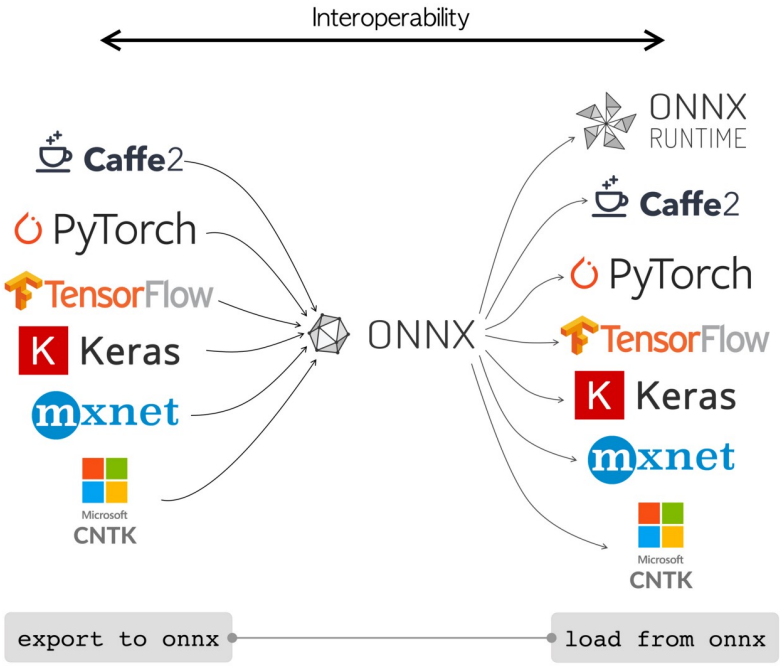

Figure 2-9: ONNX Interoperability [21]

While ONNX aims to provide a common representation, some complex or framework-specific operations may not have direct equivalents in ONNX, requiring approximations or workarounds. Model conversion between frameworks and ONNX may introduce minor differences or precision issues, depending on the operator implementations and numerical precision used. Ongoing development and community support are necessary to ensure ONNX's compatibility with the latest deep learning frameworks and hardware platforms.

Chapter 3 Methodology

This chapter presents the methodology employed in scrutinizing various aspects of Voice Activity Detection (VAD) systems. It encompasses the operational environment, programming languages, and frameworks, alongside discussions on dataset selection and preparation, training procedures, and evaluation methodologies. The sequence of experiments unfolds as follows: Experiment 1 establishes baseline performance by conducting initial inferences with Pyannote, MarbleNet, and Silero. Experiment 2 explores optimal configurations through variations in chunk sizes and strides to gauge their impact on model performance. Experiment 3 involves domain-wise analysis using the DIHARD III dataset to assess performance across different domains. Experiment 4 intricately details Pyannote training, covering hyperparameters and training procedures. Additionally, Experiment 5 evaluates the efficiency of ONNX for inference, providing insights into its comparative performance against traditional inference methods.

3.1 Operating Environment

All the experiments were done on Aspire 2A GPU cluster provided by National Super Computing Centre (NSCC), Singapore. For CPU-based computations, HPE Cray EX 2x AMD EPYC Millan 7713 was used while for GPU-based computations, GPU nodes comprising of 4xNVIDIA A100-40G SXM per node were used.

3.2 Programming Languages and Frameworks

The programming languages used in this thesis for conducting experiments are Python 3.8 and C++ (compiled with GCC 11.4.0).

The main libraries and frameworks used for training and inference of models are PyTorch, Librosa, NumPy and SoundFile. PyTorch models were transformed into C++ using the Open Neural Network Exchange (ONNX) framework [20].

3.3 Datasets and Data Preparation

This section describes the features of each dataset used in training and testing of VAD models for this thesis.

3.3.1 AliMeeting

The AliMeeting Mandarin corpus [22] was developed for the ICASSP 2022 Multi-channel Multi-party Meeting Transcription Challenge (M2MeT). It comprises recordings from actual meetings, capturing both distant speech via an 8-channel microphone array and close-range speech using participants' headset microphones. The dataset contains a total of 118.75 hours of speech data, split into 104.75 hours for training (Train), 4 hours for evaluation (Eval), and 10 hours for testing (Test). The Train, Eval, and Test portions include 212, 8, and 20 meeting sessions respectively, with each session involving discussions lasting 15 to 30 minutes and featuring 2-4 participants. This dataset represents various real-world meeting scenarios, including different meeting settings, participant numbers, and speaker overlap levels. The dataset includes high-quality transcriptions and supports tasks such as meeting transcription

enhancement, speaker identification or counting, and multi-speaker automatic speech recognition. Audio files in WAV format are provided for training, testing, and evaluation, alongside ground truth transcriptions in TextGrid format. More detailed information about the AliMeeting dataset can be found in Table 3-1 [22].

	Train	**Eval**
Duration (in hours)	104.75	4
# Sessions	212	8
# Rooms	12	5
# Participants	456	25
# Males	246	12
# Females	210	13
# Overlap Ratio (Avg.)	42.27%	24.20%

Table 3-1: Details of AliMeeting Train and Eval dataset [22]

3.3.2 DIHARD III

The DIHARD III dataset was originally designed for the third DIHARD Speech Diarization Challenge [23]. The challenge provides a dataset of approximately 34 hours of English and Chinese speech data along with corresponding annotations used in support of the challenge. The development and evaluation sets are drawn from a diverse sampling of sources including monologues, map task dialogues, broadcast interviews, sociolinguistic interviews, meeting speech, speech in restaurants, clinical recordings, and YouTube videos. The development and evaluation datasets have audio files in '.wav' format and the ground truth is in '.rttm' format. Table 3-2 provides an insight into Dihard-3 dataset [23].

Set	Partition	# recordings	# hours	% speech	% overlap
Dev	Core	181	23.94	78.43	10.04
	Full	254	34.15	79.81	10.70
Eval	Core	184	22.73	77.35	8.75
	Full	259	33.01	79.11	9.35

Table 3-2: Overview of DIHARD III datasets [23]

3.3.3 Kaldi Toolkit

Kaldi [24] is an open-source toolkit designed for dealing with speech data. It is widely used in voice-related applications, primarily for speech recognition, but also for other tasks such as speaker recognition and speaker diarization. This toolkit was used to prepare the audio files and their corresponding ground truths before being assigned to the data loaders for training.

3.4 Training Procedure

In this section, we provide an overview of the training procedure for a VAD model. The training procedure encompasses several key steps, including data preparation, model architecture selection, training optimization, and evaluation. Each step plays a vital role in the overall success of the training process, ultimately leading to a robust and accurate VAD model.

3.4.1 Dataloaders

In the training procedure, dataloaders play a crucial role in efficiently loading and preprocessing the data for training the voice activity detection (VAD) model. The dataloaders are responsible for managing the input data, which includes both the raw audio samples and their corresponding labels indicating voice activity segments.

The dataloader is implemented using PyTorch's DataLoader module, which facilitates parallel data loading and preprocessing. The dataset is custom-designed to handle the specific requirements of the VAD task, including:

- **Chunking:** The audio data is segmented into chunks of fixed size to facilitate training. Each chunk is considered as an individual training sample.
- **Data Augmentation:** The dataloader incorporates data augmentation techniques such as adding random noise to the audio samples to enhance the model's robustness to real-world conditions.
- **Batching:** The dataloader batches multiple chunks together to form mini-batches, which are fed into the model during training. Batching improves training efficiency by processing multiple samples in parallel.
- **Label Generation:** Voice activity masks are generated from RTTM (Rich Text Time Marked) files associated with the audio data. These masks indicate the presence or absence of speech in each time frame of the audio signal.

The *Dataloader* class is implemented in Python and utilizes the *torch.DataLoader* module for efficient data handling. It encapsulates the logic for loading and preprocessing the data, providing a seamless interface for integrating the data into the training pipeline.

3.4.2 Pyannote Hyperparameters

The Pyannote model architecture consists of SincNet (CNN layers), LSTM, and optional feed-forward (linear) layers. SincNet extracts low-level features from the input audio waveform, LSTM layers capture temporal dependencies, and feed-forward layers further process the extracted features. Table 3-3 describes the various hyperparameters required for training the model.

Layer	**Parameter**	**Description**
SincNet	stride	Stride parameter for SincNet block
LSTM	hidden_size	Number of units in the LSTM hidden layer
	num_layers	Number of LSTM layers
	bidirectional	Whether the LSTM is bidirectional
	monolithic	Whether to use a single monolithic LSTM layer
	dropout	Dropout probability for LSTM layers
Linear	hidden_size	Number of units in the linear layers
	num_layers	Number of linear layers

Table 3-3: Pyannote hyperparameters

3.4.3 Training

The training loop iterates over the training dataset to optimize the model parameters using gradient descent. Algorithm 1 describes the steps involved:

Algorithm 1 Training Procedure

```
Set the model to training mode
for each epoch do
    for each batch in training dataset do
        Transfer batch data to appropriate device ▷ e.g., GPU
        Perform forward pass through model to get predictions
        Calculate loss between predictions and ground truth labels ▷ e.g., binary cross-entropy loss
        Compute gradients of loss w.r.t. model parameters
        Update model parameters using optimizer ▷ e.g., stochastic gradient descent
        Log training loss and metrics
    end for
end for
```

3.4.4 Evaluation

The evaluation loop assesses the model's performance on a separate evaluation dataset without updating its parameters. Algorithm 2 describes the steps involved:

Algorithm 2 Evaluation Procedure

```
Set model to evaluation mode
Initialize evaluation loss to 0
for each epoch do
    for each batch in evaluation dataset do
        Transfer batch data to appropriate device
        Perform forward pass through model to obtain predictions
        Calculate loss between predictions and ground truth labels
        Log evaluation loss and other metrics
        Update evaluation loss by adding batch loss
    end for
    Compute average evaluation loss over all batches
end for
```

These training and evaluation loops allow us to train the model on the training dataset and assess its performance on the evaluation dataset separately, enabling us to monitor its progress and evaluate its effectiveness in solving the task.

3.5 Experiment 1: Initial Inference

The project was started by understanding deep learning concepts and their implementations in PyTorch. After this, the literature review process commenced on the existing state-of-the-art methods for VAD models and replicating the results and procedures for Pyannote, Silero and MarbleNet VAD. Pyannote was obtained by the replication of training procedure as explained in Section 3.4. Silero was obtained from the official Github repository made by the publishers [16]. MarbleNet was obtained through a collaboration project with AI Singapore.

In this experiment, a self-trained model of Pyannote, as well as pretrained models of Silero and Marblenet, were used to perform inference on the DIHARD III Full Eval set and the Alimeeting Eval set using different values of chunk sizes (analysis window) and strides. It is important to note that each of the 3 models predict the presence of speech at the frame level. Chunks essentially are the number of frames being passed to a model at a time for prediction. Stride determines the number of chunks created for passing the model. With this concept, we will now delve into the implementation of inference procedures of the three models.

3.5.1 Pyannote

Algorithm 3 provides a brief overview of the inference procedure used for Pyannote. Parameters like chunk size, overlap percentage, sample rate, etc., can be provided as arguments for performing the inference.

Algorithm 3 Inference with Pyannote

```
1:  procedure INFERENCE(args)
2:      Initialize evaluation metrics
3:      if source is a directory then
4:          for all audio file in source do
5:              Load pretrained VAD model
6:              Perform VAD on audio file
7:              Compute evaluation metrics
8:              Update aggregate metrics and counters
9:              Clear model from memory
10:         end for
11:     else
12:         Load pretrained VAD model
13:         Perform VAD on source audio file
14:         Compute evaluation metrics
15:         Update aggregate metrics
16:     end if
17:     Print summary of aggregated evaluation metrics
18: end procedure
```

3.5.2 MarbleNet

The initial inference methodology for MarbleNet VAD can be described by Figure 3-1. In this procedure, chunks of size 63ms were created from all the audio files and saved in another directory. The path of the saved chunked files was then passed to a method to handle the retrieval of each audio chunk and provide it to the model as input. Multiple other methods then worked on processing the input, reading the ground truth from the RTTM files and then providing the output for that particular chunk. Once inference was completed, the function would retrieve the next chunk and follow the same procedure for inference.

This method of inference made it impossible to work with real-time audio as the entire data was not available to perform chunking in the beginning. Therefore, the inference file was modified to perform chunking in 100ms windows and predict the presence of voice in the current chunk before moving on to the next chunk. Furthermore, stride was also added to the input parameter for inference. The provides

the user with greater flexibility to perform inference using overlapping analysis windows. The modified version of MarbleNet inference file is described by Figure 3-2

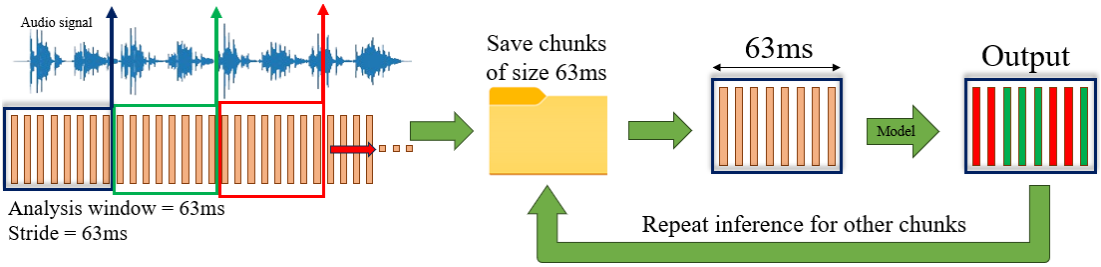

Figure 3-1: Initial MarbleNet Inference Methodology

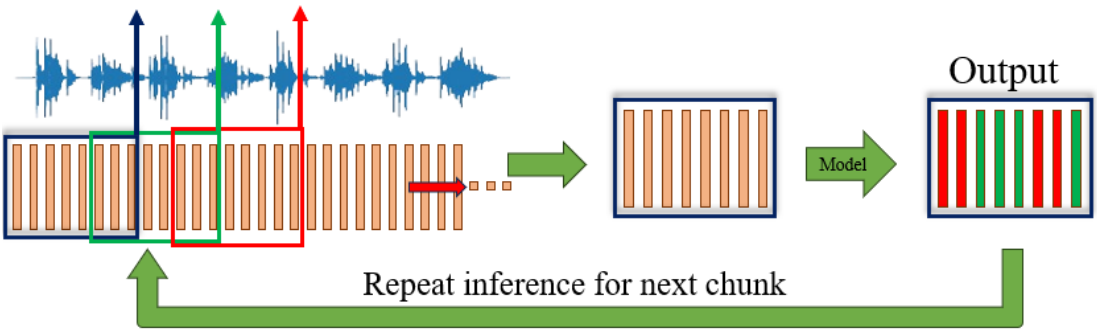

Figure 3-2: Final MarbleNet Inference Methodology

3.5.3 Silero

Silero was designed to work as an online streaming VAD due to which it provides low latency inference [16]. While low latency (low RTF) is desirable, it also leads to a decrease in accuracy and other evaluation metrics. Algorithm 4 briefly explains the inference procedure with Silero. The inference code for silero has been adapted from the official Github repository for Silero [16].

Algorithm 4 Inference with Silero

```
1:  procedure INFERENCESINGLEFILE(audio_path, out_rttm_path)
2:      Set chunk_size, stride, duration, sampling_rate
3:      Initialize empty annotation (hyp_annot)
4:      for all time_st in range(0, duration, stride) do
5:          Load audio chunk from audio_path
6:          Perform voice activity detection on audio chunk
7:          Update annotation with detected speech segments
8:      end for
9:      Write annotation to RTTM file at out_rttm_path
10: end procedure
11:
12: procedure MAIN
13:     Parse command-line arguments
14:     Initialize total metrics variables
15:     Initialize Silero VAD
16:     for all audio files in input_wav directory do
17:         Perform VAD and generate predicted RTTM file
18:         Load audio and ground truth masks
19:         Compute evaluation metrics
20:         Update total metrics
21:     end for
22:     Compute average metrics
23:     Print summary of average metrics
24: end procedure
```

3.6 Experiment 2: Effect of Varying Chunk Sizes and Strides

From experiment 1, we concluded that threshold fine-tuning is required only towards the end as it just helps balance the values of missed detection rate and false alarm rate. In this experiment, we investigate the impact of varying chunk sizes and strides on the performance of three different speech activity detection models: Pyannote, Silero, and MarbleNet. The experiments are conducted on the DIHARD III Eval set, aiming to explore how different configurations of chunk sizes and strides influence the models' evaluation metrics.

For Pyannote, Silero and MarbleNet, we analyze the performance across various configurations, including chunk sizes ranging from 5 seconds to 0.5 seconds and strides ranging from 5 seconds to 0.1 seconds to assess their effects on the model's detection capabilities.

3.7 Experiment 3: DIHARD III Domain-wise Analysis

From the results of experiment 1 and 2, we concluded that Pyannote will be our model of interest. In this experiment, we try to understand the performance of Pyannote for different domains of DIHARD III. This dataset has audio from different domains like audiobooks, meeting speech, clinical interviews, web videos and conversational telephone speech [23].

For performing this analysis, The DIHARD III Eval set needs to be split domain-wise. This can be achieved reading the UEM file of the corresponding domain provided in the dataset directory and separating the FLAC (audio) and RTTM (ground truth/labels) files accordingly. Once the data is prepared, inference can be performed as described in Experiment 1. The threshold used for this experiment is 0.5. For the

domain analysis, specific domains like restaurant, meeting and webvideo are chosen because VAD models are known to perform poorly on domains with characteristics like high background noise, high speech overlap, etc. For relative comparison with a well performing domain, the same analysis was done with audiobooks domain. This domain gives good result because this domain has files with a single speaker in a noise-free environment. VAD models perform very well in such situations.

3.8 Experiment 4: Training Pyannote

Now, we will try to train the Pyannote model for different values of chunk sizes and strides to see how this affects the performance. The implementation procedure for training of Pyannote is followed from Section 3.4. The goal of this experiment is to ascertain whether models trained on small chunk size and strides perform better than our initially trained model i.e., the one trained on chunk size as 5 seconds and stride as 4 seconds.

3.8.1 Hyperparameters

Different hyperparameters control different aspects of the model, such as its capacity, regularization, optimization algorithm, and training dynamics.In this context, we provide the values of the hyperparameters used in our model. Table 3-4 provides the hyperparameter values for each layer of the Pyannote model. Table 3-5 provides the hyperparameter values related to training and size of input to the model.

Layer	Hyperparameter	Value
SincNet	stride	10
LSTM	hidden_size	128
	num_layers	2
	bidirectional	True
	monolithic	True
	dropout	0
Linear	hidden_size	128
	num_layers	2

Table 3-4: Pyannote layer hyperparameters

Hyperparameter	Value
Max. number of epochs	50
Batch size	16
Chunk size (s)	5
Stride (s)	4
Frame size (s)	0.025
Frame shift (s)	0.01
Optimiser	Adam
Learning rate	10^{-4}
Max. Gradient Norm	5
Number of channels	1
Sample rate (Hz)	16000

Table 3-5: Additional Pyannote hyperparameters

3.8.2 Training procedure

Using the hyperparameter values provided in Subsection 3.8.1 we train Pyannote model using the procedure described in Section 3.4. After successfully replicating the results of the baseline model (Pyannote with chunk size 5s and stride 5s), we retrain the model for varying values of chunk sizes with the goal of observing whether models trained on small chunk size and stride perform well during inferencing with small chunks and strides. The reason for using small chunks and strides is to ensure that the model can perform well in real-time streaming conditions.

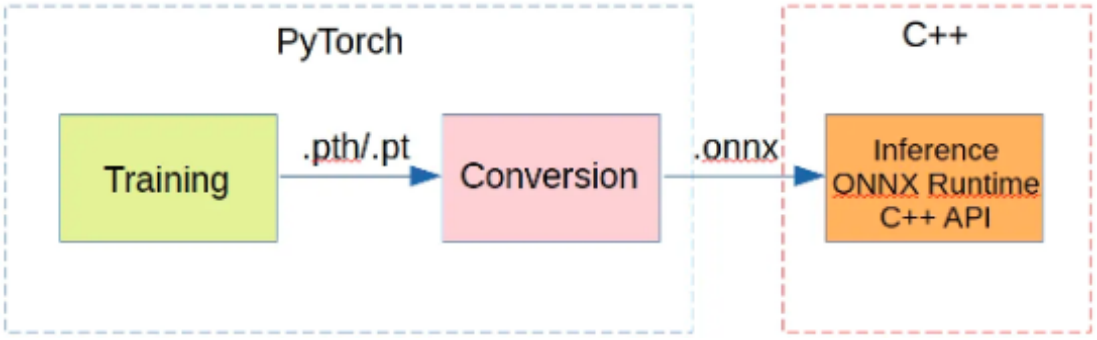

Figure 3-3: ONNX Pipeline

3.9 Experiment 5: ONNX for Inference

In this experiment, we aim to use ONNX to perform quicker inference in C++ with the best available model, ie., Pyannote. The pipeline for creating the ONNX version of the model is described by Figure 3-3.

3.9.1 Inference with ONNX model

Similar to the inference mechanism in Python, the C++ code also implements the procedure in which chunks containing multiple frames are passed to the model and the model predicts the presence of speech for each frame. Then, the predictions are compared to the ground truth i.e., an RTTM file and the inference times are compared. The same procedure is described by Figure 3-4.

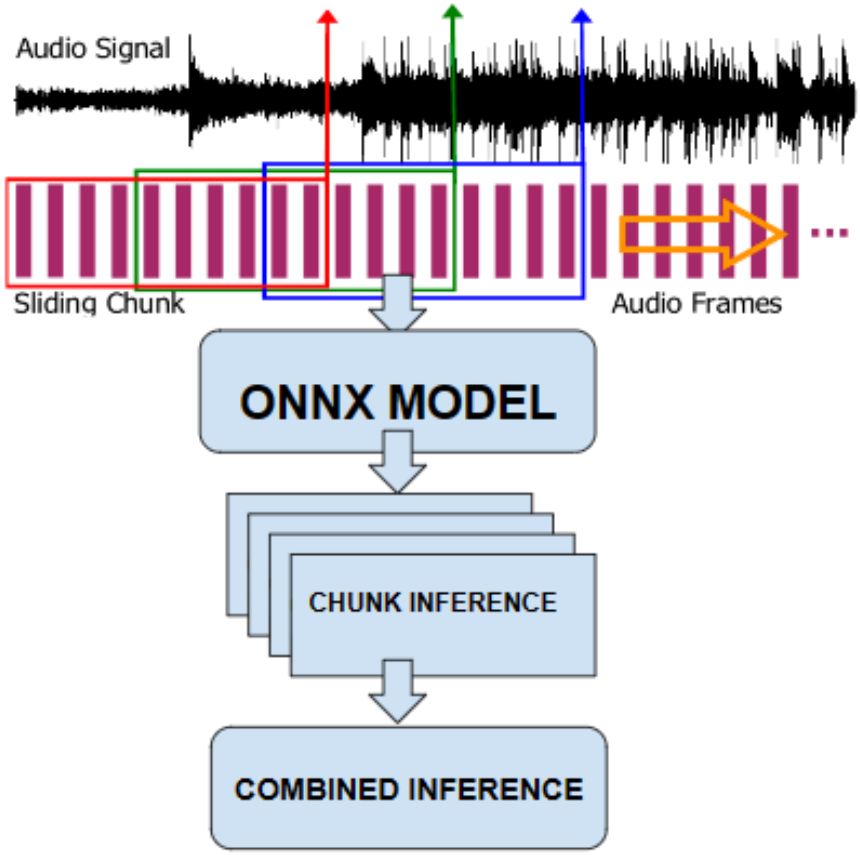

Figure 3-4: Inference with .ONNX Pyannote model

Chapter 4 Results and Discussion

This chapter delves into the findings and insights derived from the conducted experiments. Experiment 1 focuses on initial inference, where the baseline performance of Pyannote, MarbleNet, and Silero is evaluated. Within this experiment, specific modifications in MarbleNet inference are also explored. Experiment 2 investigates the effects of varying chunk sizes and strides on model performance. Experiment 3 conducts a domain-wise analysis using the DIHARD III dataset to assess performance across different domains. Experiment 4 elaborates on the training of Pyannote, detailing the procedures and outcomes. Finally, Experiment 5 evaluates the utilization of ONNX for inference, comparing its performance against traditional inference with Python. Through these experiments, the chapter presents a comprehensive discussion of the results, offering insights into the efficacy and implications of each approach in the context of VAD systems.

4.1 Experiment 1: Initial Inference

The inference results with both chunk size and stride as 5s on DIHARD III evaluation set and Alimeeting test set are provided in Table 4-1 and Table 4-2 respectively:

Model	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
Pyannote	5s	5s	0.97	1.00	2.00	0.941	0.095
Silero			0.83	11.1	6.1	0.795	0.022
MarbleNet			0.77	1.6	21.6	0.536	0.011

Table 4-1: DIHARD III test results for various VAD models

Model	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
Pyannote	5s	5s	0.95	3.0	1.7	0.777	0.011
Silero			0.74	21.9	4.0	0.624	0.038
MarbleNet			0.92	8.7	4.8	0.732	0.047

Table 4-2: Alimeeting test results for various VAD models

4.1.1 Modifications in MarbleNet Inference

While Pyannote and Silero already had the compatibility to work as real-time streaming VADs, MarbleNet did not because its methodology for inference was different as explained in Subsection 3.5.2. In this Subsection, we explore the performance of the modified MarbleNet inference methodology for 100ms chunk size and stride with different thresholds to replicate the functioning of a real-time VAD. Experiments with varying chunk sizes and strides are performed in Experiment 2.

Threshold	Chunk size	Stride	RTF	Accuracy	MDR	FAR	ROC-AUC
0.4	100ms	100ms	0.0260	0.787	3.1	18.2	0.583
0.5			0.0260	0.791	3.5	17.4	0.594
0.6			0.0258	0.795	3.9	16.6	0.605
0.7			0.0259	0.800	4.3	15.7	0.616
0.8			0.0259	0.804	4.9	14.7	0.629

Table 4-3: MarbleNet test values for different thresholds using DIHARD III

Threshold	Chunk size	Stride	RTF	Accuracy	MDR	FAR	ROC-AUC
0.4	100ms	100ms	0.0290	0.925	3.2	3.7	0.655
0.5			0.0287	0.921	3.8	4.1	0.695
0.6			0.0260	0.920	4.6	3.4	0.728
0.7			0.0260	0.918	5.4	2.9	0.755
0.8			0.0249	0.914	6.2	2.4	0.776

Table 4-4: MarbleNet test values for different thresholds using AliMeeting

Discussion for Experiment 1

From Table 4-1 and Table 4-2, it is evident that Pyannote outperforms both Silero and MarbleNet when the chunk size and stride are both 5 seconds.

From Table 4-3, we notice that across all thresholds, there's a slight variation in accuracy from 0.787 to 0.804. The change in accuracy is minimal despite varying the threshold, indicating that the model's performance remains relatively stable. The RTF remains consistent at around 0.0260 for all thresholds, suggesting consistent computational efficiency regardless of the threshold. As the threshold increases, MDR tends to decrease while FAR tends to increase. This indicates a trade-off between miss detections and false alarms.

From Table 4-4, we notice that there's minimal variation in accuracy across different thresholds, ranging from 0.914 to 0.921. This suggests that the model's performance is relatively stable across thresholds. Similar to the DIHARD III dataset, the RTF remains consistent at around 0.0260 for all thresholds in the AliMeeting dataset. As the threshold increases, MDR generally decreases while FAR tends to increase.

Across both datasets, there is a trade-off between MDR and FAR. Adjusting the

threshold allows for tuning this trade-off. Higher thresholds tend to reduce MDR but may increase FAR, and vice versa. Finding an optimal threshold involves balancing these factors based on the specific requirements of the application. This is because with higher thresholds, the model evaluates each prediction more strictly resulting in lower false alarms and higher missed detections.

Next, we analyse the performance of Pyannote, Silero and MarbleNet for different values of chunk sizes and strides.

4.2 Experiment 2: Effects of Varying Chunk Sizes and Strides

In this experiment, we investigate the impact of varying chunk sizes and strides on the performance of three different speech activity detection models: Pyannote, Silero, and MarbleNet.

Model	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
Pyannote	5s	5s	0.970	1.0	2.0	0.941	0.001
	1s	1s	0.933	2.3	4.5	0.879	0.003
	1s	0.2s	0.903	0.2	9.5	0.775	0.013
	0.5s	0.5s	0.901	3.0	6.9	0.819	0.005
	0.5s	0.2s	0.885	0.7	10.8	0.743	0.011
	0.5s	0.1s	0.865	0.3	13.3	0.692	0.021

Table 4-5: Pyannote inference results with varying chunk sizes and strides on the DIHARD III Eval set

Model	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
Silero	5s	5s	0.828	11.1	6.1	0.795	0.039
	1s	1s	0.714	24.7	3.9	0.759	0.096
	1s	0.2s	0.851	7.2	7.8	0.787	0.452
	0.5s	0.5s	0.681	26.8	5.1	0.716	0.175
	0.5s	0.2s	0.839	7.2	9.0	0.761	0.432
	0.5s	0.1s	0.846	4.8	10.6	0.738	0.844

Table 4-6: Silero inference results with varying chunk sizes and strides on the DIHARD III Eval set

Model	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
MarbleNet	5s	5s	0.77	1.6	21.6	0.536	1.359
	1s	1s	0.79	1.5	19.0	0.574	0.267
	1s	0.2s	0.78	0.3	21.8	0.534	0.089
	0.5s	0.5s	0.79	1.3	19.2	0.574	1.228
	0.5s	0.2s	0.78	0.3	21.5	0.540	0.053
	0.5s	0.1s	0.78	0.1	22.4	0.526	0.020

Table 4-7: MarbleNet inference results with varying chunk sizes and strides on the DIHARD III Eval set

Discussion for Experiment 2

For Pyannote, smaller chunk sizes and strides correlate with decreased accuracy, higher missed detection rates (MDR) and false alarm rates (FAR), and reduced ROC-AUC values. Conversely, larger chunk sizes and strides tend to maintain stability in performance metrics. Silero exhibits mixed trends, with some configurations showing decreased accuracy and increased MDR and FAR with smaller chunk sizes and strides. MarbleNet demonstrates relatively stable but low accuracy across different

configurations, with minor fluctuations in other performance metrics. Overall, while smaller chunk sizes and strides may offer finer granularity, they often come at the cost of decreased accuracy and increased computational costs. On comparing all of the 3 models, Pyannote exhibits the best performance with higher average evaluation metrics. Therefore, for the next few experiments, Pyannote will be our model of interest.

4.3 Experiment 3: DIHARD III Domain-wise Analysis

In this experiment, we will be looking into the performance of Pyannote model (trained on chunk size 5s and stride 4s) for different values of chunks and strides across different domains of DIHARD III dataset. Restaurant, Webvideo and Meeting domains have been chosen because they are amongst the hardest domains with a lot of background noise. Inference has also been done on Audiobooks domain to provide a comparison of performance between the harder and easier domains.

Domain	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
Restaurant	5s	5s	0.960	1.7	2.4	0.888	0.078
	1s	1s	0.886	7.4	4.0	0.776	0.076
	1s	0.4s	0.911	1.9	7.0	0.680	0.035
	0.5s	0.5s	0.840	10.8	5.3	0.704	0.070
	0.5s	0.3s	0.871	6.0	6.9	0.662	0.049
	0.5s	0.2s	0.887	3.1	8.1	0.628	0.032

Table 4-8: Performance metrics for Pyannote in the Restaurant domain (DIHARD III Eval set)

Domain	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
Meeting	5s	5s	0.966	1.3	2.1	0.929	0.088
	1s	1s	0.941	2.3	3.7	0.875	0.074
	1s	0.4s	0.931	0.6	6.2	0.809	0.034
	0.5s	0.5s	0.913	3.5	5.2	0.821	0.070
	0.5s	0.3s	0.913	1.6	7.1	0.778	0.048
	0.5s	0.2s	0.909	0.7	8.4	0.747	0.032

Table 4-9: Performance metrics for Pyannote in Meeting domain (DIHARD III Eval set)

Domain	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
Webvideo	5s	5s	0.947	1.7	3.6	0.917	0.080
	1s	1s	0.871	3.8	9.1	0.812	0.087
	1s	0.4s	0.848	1.1	14.1	0.747	0.030
	0.5s	0.5s	0.833	4.7	12.0	0.759	0.070
	0.5s	0.3s	0.827	2.3	15.0	0.721	0.048
	0.5s	0.2s	0.816	1.2	17.2	0.693	0.031

Table 4-10: Performance metrics for Pyannote in Webvideo domain (DIHARD III Eval set)

Domain	Chunk size	Stride	Accuracy	MDR	FAR	ROC-AUC	RTF
Audiobooks	5s	5s	0.985	0.4	1.1	0.973	0.091
	1s	1s	0.962	0.8	3.0	0.929	0.075
	1s	0.4s	0.946	0.1	5.3	0.883	0.029
	0.5s	0.5s	0.919	1.4	6.7	0.843	0.070
	0.5s	0.3s	0.905	0.5	9.0	0.797	0.049
	0.5s	0.2s	0.889	0.2	10.9	0.756	0.032

Table 4-11: Performance metrics for Pyannote in Audiobooks domain (DIHARD III Eval set)

Discussion for Experiment 3

In general, larger chunk sizes (e.g., 5s) tend to result in higher accuracy compared to smaller chunk sizes (e.g., 0.5s) across all domains. Smaller strides (e.g., 0.2s) often lead to lower accuracy compared to larger strides (e.g., 1s) for a given chunk size. The highest accuracy is typically achieved with a combination of a large chunk size (5s) and a large stride (5s) across all domains.

Smaller chunk sizes and smaller strides generally result in higher MDR and FAR values, indicating more missed detections and false alarms. Larger chunk sizes (e.g., 5s) and larger strides (e.g., 5s) tend to have lower MDR and FAR values, suggesting better precision and recall. There appears to be a trade-off between MDR and FAR, where configurations that reduce one metric often increase the other. This trade-off can be further balanced by performing threshold optimisation as suggested by Discussion for Experiment 1.

The ROC-AUC (Receiver Operating Characteristic Area Under the Curve) metric follows a similar pattern to accuracy, where larger chunk sizes and larger strides generally result in higher ROC-AUC values. The highest ROC-AUC values are

typically achieved with a combination of a large chunk size (5s) and a large stride (5s) across all domains. ROC-AUC provides an overall assessment of the model's performance, and higher values indicate better discrimination between positive and negative instances.

The Real-Time Factor (RTF) remains relatively consistent across different configurations within each domain, suggesting that computational efficiency is not significantly affected by chunk size and stride. However, smaller chunk sizes and smaller strides tend to have slightly lower RTF values compared to larger chunk sizes and larger strides because low values of chunk size and stride make more segments for each audio file as compared to large chunk sizes and strides.

The Audiobooks domain generally exhibits higher accuracy, lower MDR and FAR, and higher ROC-AUC values compared to the other domains. The Webvideo, Meeting and Restaurant domains tend to have lower accuracy, higher MDR and FAR, and lower ROC-AUC values. This is because Audiobooks domain has data from a single speaker reciting a book in a noise-free environment whereas the other domains contain data from noisy environments with multiple speakers.

Another interesting observation is that for all the small chunk sizes (¡=1s), Pyannote performs reasonably on chunk size of 1s and stride of 0.4s. Therefore, we will be using this value to retrain the model and compare the performance with the baseline model i.e., Pyannote trained with chunk size 5s and stride 4s.

4.4 Experiment 4: Training Pyannote

Building on the conclusions provided by Discussion for Experiment 3, we trained a model on chunk size 1s and stride 0.4s.

For the purpose of convenience, different Pyannote models will be referred in this format: **Py (x, y)**

where,

'**Py**' refers to the model name i.e., Pyannote,

'**x**' refers to the chunk size (in seconds) **Py** was trained on,

'**y**' refers to the stride (in seconds) **Py** was trained on.

Table 4-12 compares the inference results on chunk size 1s and stride 0.4s for these 2 models:

1. Pyannote trained on chunk size 5s and stride 4s

2. Pyannote trained on chunk size 1s and stride 0.4s

Domain	Model	Accuracy	MDR	FAR	ROC-AUC	RTF
Restaurant	Py (5, 4)	0.911	1.9	7.0	0.680	0.035
	Py (1, 0.4)	0.891	3.0	7.9	0.642	0.034
Webvideo	Py (5, 4)	0.848	1.1	14.1	0.747	0.030
	Py (1, 0.4)	0.831	3.9	13.0	0.761	0.035
Meeting	Py (5,4)	0.931	0.6	6.2	0.809	0.034
	Py (1, 0.4)	0.848	0.9	14.2	0.596	0.034
Audiobooks	Py (5,4)	0.946	0.1	5.3	0.883	0.029
	Py (1, 0.4)	0.976	0.7	1.7	0.956	0.035
DIHARD III	Py (5, 4)	0.92	0.6	7.4	0.821	0.029
Eval set	Py (1, 0.4)	0.92	1.8	6.2	0.846	0.034

Table 4-12: Performance Comparison between Py (5, 4) and Py (1, 0.4) across DIHARD III Eval set domains on chunk size 1s and stride 0.4s

Additionally, we tried training models on other small chunk sizes and strides as well. It is important to note that the data provided in the below tables were obtained using a **prediction threshold value of 0.9**.

Chunk size	Stride	Domain	Accuracy	MDR	FAR	ROC-AUC	RTF
3s	3s	Restaurant	0.723	24.8	2.9	0.731	0.035
		Webvideo	0.775	17.3	5.2	0.808	0.036
		Meeting	0.834	15.9	0.8	0.874	0.034
		DIHARD III Eval set	0.897	8.7	1.6	0.908	0.036

Table 4-13: Performance of Py (3, 2) across DIHARD III Eval set domains on chunk size 3s and stride 3s

Chunk size	Stride	Domain	Accuracy	MDR	FAR	ROC-AUC	RTF
1.5s	1.5s	Restaurant	0.692	29.1	1.7	0.751	0.035
		Webvideo	0.767	19.1	4.2	0.807	0.036
		Meeting	0.831	16.3	0.6	0.878	0.034
		DIHARD III Eval set	0.882	10.5	1.2	0.903	0.036

Table 4-14: Performance of Py (1.5, 1) across DIHARD III Eval set domains on chunk size 1.5s and stride 1.5s

Chunk size	Stride	Domain	Accuracy	MDR	FAR	ROC-AUC	RTF
0.75s	0.75s	Restaurant	0.645	34.3	1.1	0.0.742	0.035
		Webvideo	0.776	17.8	4.6	0.810	0.036
		Meeting	0.834	15.8	0.7	0.880	0.034
		DIHARD III Eval set	0.887	9.8	1.5	0.902	0.036

Table 4-15: Performance of Py (0.75, 0.5) across DIHARD III Eval set domains on chunk size 0.75s and stride 0.75s

For the model trained on smallest chunk size and stride, ie., Py (0.1875, 0.125), we provide inference results on chunk size 0.1875 and stride 0.1875 for Py (1, 0.4) as well for a better overall comparison.

Domain	Model	Accuracy	MDR	FAR	ROC-AUC	RTF
Restaurant	Py (5, 4)	0.727	21.9	5.4	0.0.640	0.035
	Py (0.1875, 0.125)	0.622	36.1	1.8	0.0.714	0.035
Webvideo	Py (5, 4)	0.772	9.8	13.0	0.702	0.030
	Py (0.1875, 0.125)	0.755	18.7	5.8	0.784	0.035
Meeting	Py (5, 4)	0.809	13.7	5.4	0.755	0.034
	Py (0.1875, 0.125)	0.801	18.8	1.2	0.848	0.034
DIHARD III Eval set	Py (5, 4)	0.842	7.3	8.5	0.754	0.029
	Py (0.1875, 0.125)	0.869	11.2	1.9	0.885	0.034

Table 4-16: Performance Comparison between Py (5, 4) and Py (0.1875, 0.125) across DIHARD III Eval set domains on chunk size 0.1875s and stride 0.1875s

Discussion for Experiment 4

In Table 4-12, for the Restaurant domain, both models exhibit relatively high accuracies, with Py (5, 4) achieving 0.911 and Py (1, 0.4) achieving 0.891. However, Py (5, 4) outperforms Py (1, 0.4) in terms of minimizing the Miss Detection Rate (MDR) and False Alarm Rate (FAR), indicating better performance in event detection. Similar trends are observed in the Webvideo and Meeting domains. Py (5, 4) consistently achieves higher accuracies compared to Py (1, 0.4), while maintaining lower MDR and FAR values. This suggests that the configuration with a larger chunk size and stride (Py (5, 4)) generally performs better in these domains. Interestingly, in the Audiobooks domain, both configurations perform exceptionally well, with accuracies exceeding 0.94. However, Py (1, 0.4) achieves a slightly higher accuracy of 0.976 compared to Py (5, 4) with an accuracy of 0.946. This suggests that Py (1, 0.4) is better at generalising data from easier domains, i.e., Py (1, 0.4) is not as robust as Py (5, 4).

In Table 4-13, the model finds a good balance between missing detections (MDR) and false alarms (FAR) on the ”Meeting” domain with high Accuracy and ROC-AUC. ”Restaurant” and ”Webvideo” domains show a decent trade-off, with MDR being higher than ”Meeting” but still acceptable. The DIHARD III Eval set achieves the best balance with low MDR (8.7%) and FAR (1.6%), reflected in the comparatively high Accuracy and ROC-AUC.

Similar to Table 4-13, in Table 4-14 Py (1.5, 1) achieves a good balance on the ”Meeting” domain. There’s a slight increase in MDR and FAR compared to Table 4-13 for all domains, but the impact on overall performance seems minimal. The DIHARD III Eval set maintains a good balance with moderate MDR (10.5%) and low FAR (1.2%).

In Table 4-15, despite lower Accuracy across all domains, Py (0.75, 0.5) shows a similar balance of MDR and FAR based on the DIHARD III Eval set (high Accuracy and ROC-AUC). This is because Py (0.75, 0.5) might've generalised the easier domains like Audiobooks, Clinical, Maptask, etc., similar to Py (1, 0.4).

The general observation from all the tables in this experiment point to the fact that models trained on small chunk size and strides do not necessarily perform well during inference with small chunks and strides. Py (5,4) is more generalised and robust to all chunk size and stride. This probably has something to do with the inferencing mechanism of Pyannote. Audio data is broken down into chunks for passing as input to the model but the predictions are done at the frame level. As mentioned in Section 2.1, a chunk comprises of multiple consecutive frames and the model predicts the presence of speech at the frame level. Building upon the fact that the architecture is BLSTM-based, larger chunk sizes help provide more context to the model for a more accurate prediction. Due to this reason, VAD models trained on large chunk sizes continue to perform well for tasks which require a small chunk size as input.

4.5 Experiment 5: ONNX for Inference

To improve the Real Time Factor values of VAD models to enhance their suitability for use in real-time VAD systems, we can further use Open Neural Network Exchange (ONNX) [20] to perform inference in C++. Inference times were recorded for Pyannote model in both Python and C++ with various lengths of audio files were recorded and compared. Table 4-17 presents a comparison between the inference times of the Pyannote model in Python and C++ using Open Neural Network Exchange (ONNX) across different lengths of audio files. The model configuration includes a chunk size of 5 seconds, a stride of 5 seconds, and a threshold of 0.5.

S. No.	Length of Audio	Python	C++
1	616.803 sec	33.08 sec	2 sec
2	49 sec	1.81 sec	<1 sec
3	599.952 sec	26.07 sec	1.8 sec
4	368.43 sec	15.29 sec	1.2 sec
5	33 hrs 0 min 28 sec	58 min 20 sec	12 min 3 sec

Table 4-17: Comparison between Inference times in Python and C++ for Pyannote with chunk size as 5s, stride as 5s and threshold as 0.5

Discussion for Experiment 5

The most notable observation is the significant reduction in inference time when using the C++ implementation compared to Python. For instance, for a 616.803-second audio file (approximately 10 minutes), the inference time reduces from 33.08 seconds in Python to only 2 seconds in C++. This reduction is drastic and demonstrates the efficiency of the C++ implementation for real-time processing.

The improvement in inference time is consistent across different lengths of audio files. Whether it's a shorter 49-second audio file or a longer 33-hour audio file, the C++ implementation consistently outperforms Python in terms of inference time. This consistency indicates that the efficiency gains from using C++ are not limited to specific scenarios but apply across various lengths of audio files.

The scalability of the C++ implementation is evident, especially for larger audio files. For instance, for a 33-hour and 28-second audio file, the inference time reduces from 58 minutes and 20 seconds in Python to only 12 minutes and 3 seconds in C++. This scalability is crucial for real-time VAD systems that may need to process long audio recordings efficiently.

Chapter 5 Conclusion

In this chapter, we outline challenges in single-channel voice activity detection (VAD), including speech variability, background noise, segmentation errors, scalability concerns, and domain adaptation challenges. We then discuss our research contributions, which involved evaluating Pyannote, Silero, and MarbleNet VAD models under various conditions and exploring different chunk sizes and strides. Our experiments identified Pyannote as the top-performing model, prompting further optimization efforts. Finally, we highlight potential future research directions in VAD, such as exploring ensemble learning, deep reinforcement learning, transfer learning, and expanding single-channel VAD models for speaker recognition and diarization applications.

5.1 Summary of Challenges

In the realm of single-channel voice activity detection (VAD), significant challenges impact both effectiveness and efficiency. One prominent hurdle is the inherent variability in speech characteristics, where nuances like accent, pitch, and tone fluctuate even within a single speaker's discourse. Emotional states or health conditions can further complicate accurate speech detection.

Background noise and acoustic variability present another formidable challenge. These environmental factors often obscure speech signals, making it difficult for VAD systems to distinguish between speech and noise accurately. Segmentation

errors add to the complexity, as accurately delineating speech and silence segments is crucial. Misclassifying periods of silence as speech or vice versa undermines system performance.

Ensuring scalability and computational efficiency is crucial as audio recordings lengthen. Developing efficient algorithms capable of processing longer audio streams becomes imperative. Domain adaptation is also challenging, as single-speaker VAD systems may struggle to maintain high performance across diverse domains or recording conditions.

Addressing these challenges is critical for advancing single-speaker VAD systems. While the current focus is on single-channel detection, future efforts aim to expand to multi-channel detection, broadening the scope of research in this field.

5.2 Contribution

In this thesis, our primary objective was to conduct a comprehensive evaluation of three leading voice activity detection (VAD) models: Pyannote [17], Silero [16], and MarbleNet [19]. Our approach was distinct in that we aimed to provide a thorough analysis of these models under various conditions, employing diverse evaluation metrics and leveraging standard speech datasets. Additionally, we sought to explore novel avenues for enhancing the existing state-of-the-art model by experimenting with different values of chunk sizes and strides, a facet largely unexplored in the existing literature.

Our research journey began with an initial evaluation of the models using large chunk sizes and strides, focusing on understanding their baseline performance. Furthermore, we undertook the task of adapting MarbleNet to function as a real-time streaming VAD model, a critical step in enhancing its practical utility. Subsequently,

Experiment 1 delved into the effects of varying prediction threshold values, shedding light on their impact on VAD performance.

Experiment 2 represented a significant step forward as we investigated the influence of chunk size and stride variations on model performance. Leveraging the DIHARD III Eval set, we meticulously assessed the models across different domains to gain insights into their robustness and adaptability in diverse scenarios [23]. The outcomes of Experiments 1 and 2 led us to identify Pyannote as the top-performing model, showing a superiority of approximately 16.87% over Silero and approximately 25.97% over MarbleNet on the DIHARD III dataset. This recognition prompted us to focus subsequent experiments on enhancing Pyannote's capabilities.

Experiment 3 involved conducting a detailed domain-wise analysis of Pyannote using the extensive DIHARD III Eval dataset, encompassing a wide array of domains such as audiobooks, meetings, clinical interviews, web videos, and conversational telephone speech. This comprehensive analysis allowed us to gain deeper insights into the model's performance across diverse real-world scenarios.

Expanding on the insights gleaned from the preliminary investigations, Experiment 4 was devised to delve deeper into the retraining of the Pyannote model, incorporating variations in chunk sizes and strides. This experiment also encompassed the training of the state-of-the-art model. Through meticulous comparison of the performance exhibited by these retrained models, we could unravel the nuanced interplay between various parameters and their consequential effects on evaluation metrics. Notably, our analysis sheds light on the intriguing observation that models trained on smaller chunk sizes and strides do not consistently exhibit superior performance during inference with correspondingly small chunk sizes and strides. This finding underscores the need for a nuanced understanding of the intricate dynamics governing model training and inference, thereby guiding future optimization efforts.

Lastly, Experiment 5 explored strategies for enhancing the scalability and production readiness of the trained models through the Open Neural Network Exchange (ONNX) framework [20]. Leveraging this framework, we achieved a significant reduction in inference times, thereby enhancing the models' scalability and efficiency.

In essence, our research endeavors aimed to address the inherent challenges in creating robust and scalable VAD systems using deep neural network (DNN) approaches, contributing valuable insights and advancements to the field of speech processing.

5.3 Limitations of Current Work

While our research contributes significant insights into the field of voice activity detection (VAD) and deep learning-based approaches, it is essential to acknowledge certain limitations in our study:

- **Dataset Selection:** Our experiments were conducted using standard speech datasets such as DIHARD III and Alimeeting, which may not fully represent the diversity of real-world audio environments. Future research could benefit from evaluating VAD models on a wider range of datasets, including those with more varied acoustic conditions and speaker demographics.
- **Hyperparameter Tuning:** While we explored different combinations of chunk sizes and strides, our hyperparameter tuning process may not have fully optimized model performance. Fine-tuning additional hyperparameters or employing more advanced optimization techniques could further enhance model effectiveness.

- **Domain-Specific Challenges:** The performance of VAD models can vary significantly across different domains, such as audiobooks, meetings, or web videos. Our study provides insights into general trends but may not capture domain-specific challenges comprehensively.

5.4 Future Work

After successfully evaluating three prominent VAD models and identifying areas for improvement, future research could focus on several promising avenues. One potential direction is to investigate the effectiveness of ensemble learning techniques in enhancing VAD performance by combining multiple models. Additionally, exploring deep reinforcement learning approaches for adaptive VAD systems could lead to more robust and adaptable models capable of dynamically adjusting to changing environments. Moreover, incorporating transfer learning techniques to leverage pre-trained models from related tasks could expedite model training and improve performance, especially in domains with limited annotated data. Furthermore, expanding the current single-channel VAD model from a binary classification problem to a multi-class classification problem, by determining which speaker spoke during the speech segment, could open new avenues for research and applications in speaker recognition and diarization.

References

[1] F. Laricchia, *Number of voice assistants in use worldwide 2019-2024*, Mar. 2022. [Online]. Available: https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/.

[2] S. Mihalache, I.-A. Ivanov, and D. Burileanu, "Deep neural networks for voice activity detection," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, 2021, pp. 191–194. DOI: 10.1109/TSP52935.2021.9522670.

[3] A. Chhetri, P. Hilmes, T. Kristjansson, *et al.*, "Multichannel audio front-end for far-field automatic speech recognition," in *2018 26th European Signal Processing Conference (EUSIPCO)*, IEEE, Sep. 2018. DOI: 10.23919/eusipco.2018.8553149. [Online]. Available: http://dx.doi.org/10.23919/EUSIPCO.2018.8553149.

[4] S. MacDonald, *Sound fields: free versus diffuse field, near versus far field*. [Online]. Available: https://community.sw.siemens.com/s/article/sound-fields-free-versus-diffuse-field-near-versus-far-field.

[5] G. Hinton, L. Deng, D. Yu, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012. DOI: 10.1109/MSP.2012.2205597.

[6] H. Dinkel, S. Wang, X. Xu, M. Wu, and K. Yu, "Voice activity detection in the wild: A data-driven approach using teacher-student training,"

IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 1542–1555, 2021. DOI: 10.1109/TASLP.2021.3073596.

[7] M. Wang, Q. Huang, J. Zhang, *et al.*, "Deep learning approaches for voice activity detection," in *Cyber Security Intelligence and Analytics*, Z. Xu, K.-K. R. Choo, A. Dehghantanha, R. Parizi, and M. Hammoudeh, Eds., Cham: Springer International Publishing, 2020, pp. 816–826, ISBN: 978-3-030-15235-2.

[8] P. Vecchiotti, E. Principi, S. Squartini, and F. Piazza, "Deep neural networks for joint voice activity detection and speaker localization," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 1567–1571. DOI: 10.23919/EUSIPCO.2018.8553461.

[9] S. Yadav, P. A. D. Legaspi, M. S. O. Alink, A. B. J. Kokkeler, and B. Nauta, "Hardware implementations for voice activity detection: Trends, challenges and outlook," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 3, pp. 1083–1096, 2023. DOI: 10.1109/TCSI.2022.3225717.

[10] D. Freeman, G. Cosier, C. Southcott, and I. Boyd, "The voice activity detector for the pan-european digital cellular mobile telephone service," in *International Conference on Acoustics, Speech, and Signal Processing,*, 1989, 369–372 vol.1. DOI: 10.1109/ICASSP.1989.266442.

[11] F. Vesperini, P. Vecchiotti, E. Principi, S. Squartini, and F. Piazza, "Deep neural networks for multi-room voice activity detection: Advancements and comparative evaluation," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 3391–3398. DOI: 10.1109/IJCNN.2016.7727633.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[14] P. G. Jordán, I. V. Bailo, A. O. Giménez, A. M. Artiaga, and E. L. Solano, "Vivovad: A voice activity detection tool based on recurrent neural networks," *Jornada de Jóvenes Investigadores del I3A*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:196176652.

[15] I. J. Tashev, "Offline voice activity detector using speech supergaussianity," in *2015 Information Theory and Applications Workshop (ITA)*, 2015, pp. 214–219. DOI: 10.1109/ITA.2015.7308991.

[16] S. Team, *Silero VAD: Pre-trained enterprise-grade voice activity detector (VAD), number detector and language classifier*, https://github.com/snakers4/silero-vad, 2021.

[17] H. Bredin and A. Laurent, *End-to-end speaker segmentation for overlap-aware resegmentation*, 2021. arXiv: 2104.04045 [eess.AS].

[18] M. Ravanelli and Y. Bengio, *Speaker recognition from raw waveform with sincnet*, 2019. arXiv: 1808.00158 [eess.AS].

[19] F. Jia, S. Majumdar, and B. Ginsburg, *Marblenet: Deep 1d time-channel separable convolutional neural network for voice activity detection*, 2021. arXiv: 2010.13886 [eess.AS].

[20] J. Bai, F. Lu, K. Zhang, *et al.*, *ONNX: Open neural network exchange*, https://github.com/onnx/onnx, 2019.

[21] S. Gyanchandani, *ONNX, ONNX RunTime, and TensortRT - Auriga IT*, Jan. 2023. [Online]. Available: https://aurigait.com/blog/onnx-onnx-runtime-and-tensortrt/.

[22] F. Yu, S. Zhang, Y. Fu, *et al.*, "M2met: The ICASSP 2022 multi-channel multi-party meeting transcription challenge," *CoRR*, vol. abs/2110.07393, 2021. arXiv: 2110.07393. [Online]. Available: https://arxiv.org/abs/2110.07393.

[23] N. Ryant, P. Singh, V. Krishnamohan, *et al.*, *The third DIHARD diarization challenge*, 2021. arXiv: 2012.01477 [eess.AS].

[24] *Kaldi: A toolkit for speech recognition written in c++ and licensed under the apache license v2.0*, http://kaldi-asr.org/doc/about.html, Accessed on 18/03/2024.