COMPARISON METRICS AND PERFORMANCE ESTIMATIONS FOR DEEP BEAMFORMING DEEP NEURAL NETWORK BASED AUTOMATIC SPEECH RECOGNITION SYSTEMS USING MICROPHONE-ARRAYS

by

Aviad Ben-Haim

A thesis submitted to Johns Hopkins University

in conformity with the requirements for the degree of

Master of Science

Baltimore, Maryland

October 2022

Abstract

Automatic Speech Recognition (ASR) functionality, the automatic translation of speech into text, is on the rise today and is required for various use-cases, scenarios, and applications. An ASR engine by itself faces difficulties when encountering live input of audio data, regardless of how sophisticated and advanced it may be. That is especially true, under the circumstances such as a noisy ambient environment, multiple speakers, or faulty microphones. These kinds of challenges characterize a realistic scenario for an ASR system.

ASR functionality continues to evolve toward more comprehensive End-to-End (E2E) solutions. E2E solution development focuses on three significant characteristics. The solution has to be robust enough to show endurance against external interferences. Also, it has to maintain flexibility, so it can easily extend in expectation of adapting to new scenarios or in order to achieve better performance. Lastly, we expect the solution to be modular enough to fit into new applications conveniently. Such an E2E ASR solution may include several additional micro-modules of speech enhancements besides the ASR engine, which is very complicated by itself. Adding these micro-modules can enhance the robustness and improve the overall system's performance. Examples of such possible micro-modules include noise cancellation and speech separation, multi-microphone arrays, and adaptive beamformer(s). Being a comprehensive solution built of numerous micro-modules is technologically

challenging to implement and challenging to integrate into resource-limited mobile systems. By offloading the complex computations to a server on the cloud, the system can fit more easily in less capable computing devices. Nevertheless, that compute offloading comes with the cost of giving up on real-time analysis, and increasing the overall system bandwidth. In addition, offloading to a server must have connectivity to the cloud over the internet.

To find the optimal trade-offs between performance, Hardware (HW) and Software (SW) requirements or limitations, maximal computation time allowed for real-time analysis, and the detection accuracy, one should first define the different metrics used for the evaluation of such an E2E ASR system. Secondly, one needs to determine the extent of correlation between those metrics, plus the ability to forecast the impact each variation has on the others.

This research presents novel progress in optimally designing a robust E2E-ASR system targeted for mobile, resource-limited devices. First, we describe evaluation metrics for each domain of interest, spread over vast engineering subjects. Here, we emphasize any bindings between the metrics across domains and the degree of impact derived from a change in the system's specifications or constraints. Second, we present the effectiveness of applying machine learning techniques that can generalize and provide results of improved overall performance and robustness. Third, we present an approach of substituting architectures, changing algorithms, and approximating complex computations by utilizing a custom dedicated hardware acceleration in order to replace the traditional state-of-the-art SW-based solutions, thus providing real-time analysis capabilities to resource-limited systems.

Thesis Committee

Primary Readers

Cleon Davis, Ph.D. (Co-Advisor)
Johns Hopkins University Whiting School of Engineering
Engineering for Professionals

Jack Carmody, Ph.D. (Co-Advisor)
Johns Hopkins University Whiting School of Engineering
Engineering for Professionals

Clinton Edwards, Ph.D. (Academic Advisor)
Johns Hopkins University Whiting School of Engineering
Engineering for Professionals

Acknowledgments

Thanks to my family, my wife and daughters, and to all my friends for their support and encouragement throughout this research process.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Speech recognition systems started several decades ago. They began with simple decoding capabilities through keyword spotting and progressed to the modern Automatic Speech Recognition (ASR) systems that are common nowadays [11].

Recent advances in the field of speech recognition and acoustic modeling are due to the introduction of Deep Neural Networks (DNN) based alternatives vice the traditional Hidden-Markov Models—Gaussian Mixtures Models (HMM-GMM) approaches [8, 17]. With the introduction of DNN algorithms, ASR systems performance significantly improved compared to the conventional HMM-GMM based speech recognition approach [8, 17]. In addition, integration with deep beamformers, which are also DNN-based, improves the front-end (FE) performance. That performance improvement paves the way to lower error rates in results. Different neural network (NN) architectures can be used to implement ASR engine building blocks. Usually, the NN used to train the ASR engine is the same NN used in the implementation of the front-end components, microphones, channel management, and beamformers.

In recent years, the research interest has been in End-to-End (E2E) ASR systems [2]. The E2E approach consists of the "complete" pipeline process, starting with the audio domain physical signal reception up to the output of the recognition engine.

Figure 1.1 shows a general skeleton of an E2E ASR system. This general form describes the internal partitioning and the pipeline process. However, most importantly, it emphasizes the distinction between the two domains of interest, "audio-related" and "Speech-related". These two domains are sometimes referenced as "Audio-Enhancement" and "Speech-Enhancement", respectively. Because there can be an overlap between the two regions, "audio-related" and "Speech-related" are probably the more precise terms.

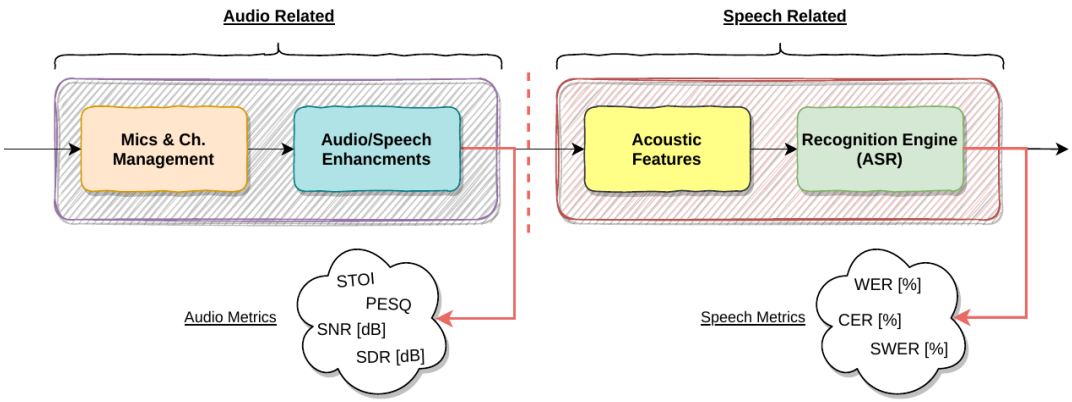


Figure 1.1: General E2E ASR System Blocks Diagram

The distinction between the two is characterized by different metrics. Typically, speech-recognition performance is evaluated using the Word Error Rate (WER) and Character Error Rate (CER) metrics. On the other hand, audio-related performance, measured on the enhanced version of the input signal, whether it be speech separated or noise suppressed, is evaluated using different metrics. If such audio enhancement processing is applied, metrics

like Clarity [9], Definition [15], Reverberation-Time [15], and other standard signal processing metrics such as Signal to Noise Ratio (SNR) and Signal to Distortion Ratio (SDR) are used.

In turn, the first building block of the “audio-related” part that is called "Microphones & Channels Management" as shown in Figure 1.1, can be evaluated with different sets of performance metrics such as Directivity Factor (DF), Noise Gain (NG) and more.

Compared to a reference and a specific performance metric, a change in any of the building blocks shown in Figure 1.1 may improve or degrade performance. On the other hand, performance increase in one metric can cause a degradation in other metrics. Hence, there is a great need to perform cross-correlation and trade-off estimations over the given metric constraints before and during the design stage of an E2E ASR system. The outcomes can help in advancing a better approach which may be a promising solution for a given use case. Furthermore, other performance metrics can be examined. For example, metrics like computation time and resources utilization can assist in having a more comprehensive overview of how the system operates given a static pipeline with different algorithms. Thus, application developers can benefit from such metrics estimations, especially when dealing with strict application requirements or resource-limited platforms.

1.1 Literature Review

References [14, 17] are two pioneering research papers focusing on studying neural networks usages for beamforming in the front-end, before the acoustic features extraction and recognition engine stages in E2E ASR systems. In both papers, the researchers built their architecture upon the principles of an E2E ASR pipeline. Those architectures share a similar baseline but have different DNN-type beamformers, filters, and ASR engines. Experiments in [14, 17] focus on WER performance evaluations to express the system's capabilities under different test-cases and inputs. Few of the test-cases, for example, are a single microphone with single channel input, a single microphone with multi-channel inputs, and a microphones array.

1.1.1 Recent Work

Recently, new techniques have been introduced [7, 10, 12, 13, 16], such as different NN architectures that serve as the basis for the ASR engines.

1.1.1.1 ASR Engine Alternatives

Performance comparisons between different recognition engines are described in [12]. The different ASR engines include Recurrent neural network-transducer (RNN-T), Recurrent neural network-attention encoder decoder (RNN-AED), and Transformer-AED. These architectures belong to the right side of the extended E2E ASR structure shown in Figure 1.1, also referred to as the Back-end (BE). BE engines are complex systems by themselves and thus can be split into multiple smaller blocks to ease integration. Despite being very comprehensive,

Reference [12] concentrates on the “Speech-related” domain, such that the primary metric used for evaluation is WER, without considering the front-end effect and its correlation to performance. In addition, the recognition engines in this paper do not make use of the Listen, Attend, and Spell (LAS) [3] nor the Connectionist Temporal Classification (CTC) [6], which are now considered as integral components in modern ASR engines after proving to enhance recognition rates drastically.

1.1.1.2 Biasing

Reference [16] explores the effect of biasing on a complete E2E ASR system pipeline. First, masking operations were applied in the frequency domain. Then, biasing information was fed into the system in combination with the masked frequency output. The added biasing information together with the T-F masking and the beamformer in the front-end showed a substantial reduction in error rate detections.

1.1.1.3 Low Latency Beamformers

Low latency beamformers were studied in [13]. The research results show "audio-related" performance comparisons as a function of the beamformer type and the number of microphones in the array. Moreover, each setup’s latency was measured to determine its time to process. The authors of this paper used two different datasets for their experiments. The TIMIT dataset [5] for the generations of noisy reverberant inputs to the microphone array and the CHiME 3 [1] dataset for ASR evaluations.

1.1.1.4 Masking Operations

Reference paper [7] demonstrates experiments on estimations of spectral masks effects with neural networks based beamformers. Two different beamformers, Generalized Eigenvector (GEV) and Minimum Variance Distortionless Response (MVDR) were tested with and without Bidirectional Long Short-Term Memory (BLSTM) spectral masks concerning the Power Spectral Distribution (PSD) and SNR. However, this paper does not include an E2E pipeline nor the recognition engine. In other words, this research focuses on the "audio-related" domain. Indeed, SNR belongs to the "audio-related' metric set rather than the "Speech-related" metric set, as shown in Figure 1.1.

Another comprehensive research that studies masking operations is described in [10]. In this paper, the architecture does not include the ASR engine. Instead, it mainly deals with the enhanced output signal, signifying that it is also in the "audio-related" domain. However, this research is unique in that it utilized both the WER and SDR metrics measurements for different engines that were plugged in at the BE stage.

1.2 Project Outline

Applications have requirements and limitations that are dictated by their platform or available hardware. A way to estimate Hardware (HW) requirements for speech applications based on speech-related or audio-related metric sets can be helpful to developers of such platforms. Therefore, optimizations of a given E2E ASR architecture by careful trade-off selections can lead to more

robust and rapidly developed applications or setups. That is a significant advance towards fast setup constructions for different HW platforms. As such, in this thesis, the effects of changing different building blocks and applying various enhancement techniques in a given E2E ASR system will be evaluated. The effects of such replacements and fine-tunes will be presented with respect to different performance metrics. Based on the results, one will be able to deduce the trade-offs that can be selected to optimize the implementation process for a specific application, platform, requirement, feasibility, and other specifications.

Evaluated performance metrics are shown in Figure 1.2

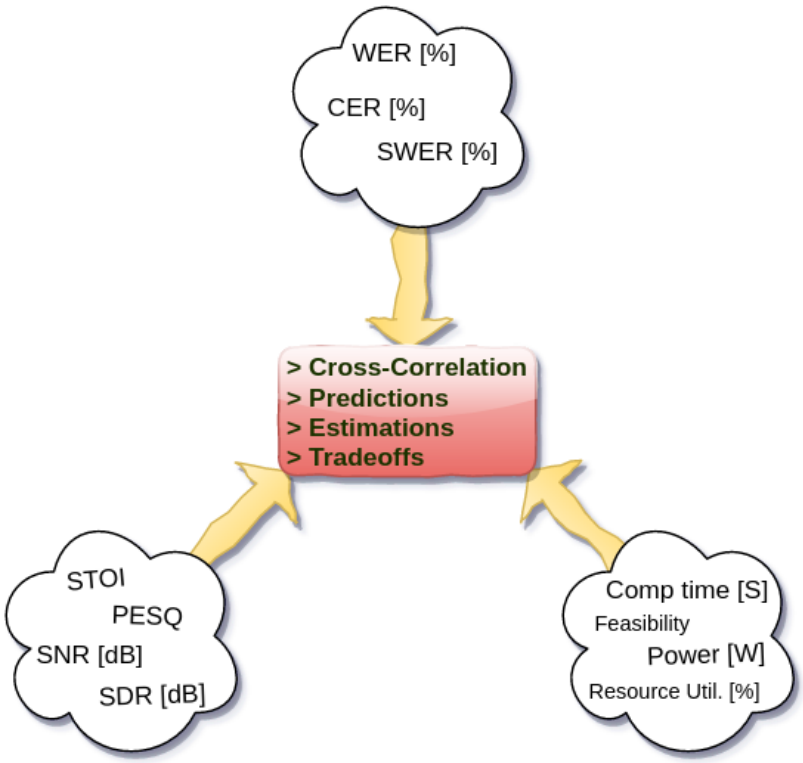


Figure 1.2: Project Outline Summary Diagram

A comprehensive performance metrics table that contains the Audio, Speech, and HW characteristics of an E2E ASR system, will be composed. Such tables make it easier to detect cross-correlations between the metric sets. In consequence, deduction of each metric's projection on others can be estimated.

Our approach is to setup an architecture that follows the entire E2E ASR pipeline including the beamforming FE as presented in Figure 1.3.

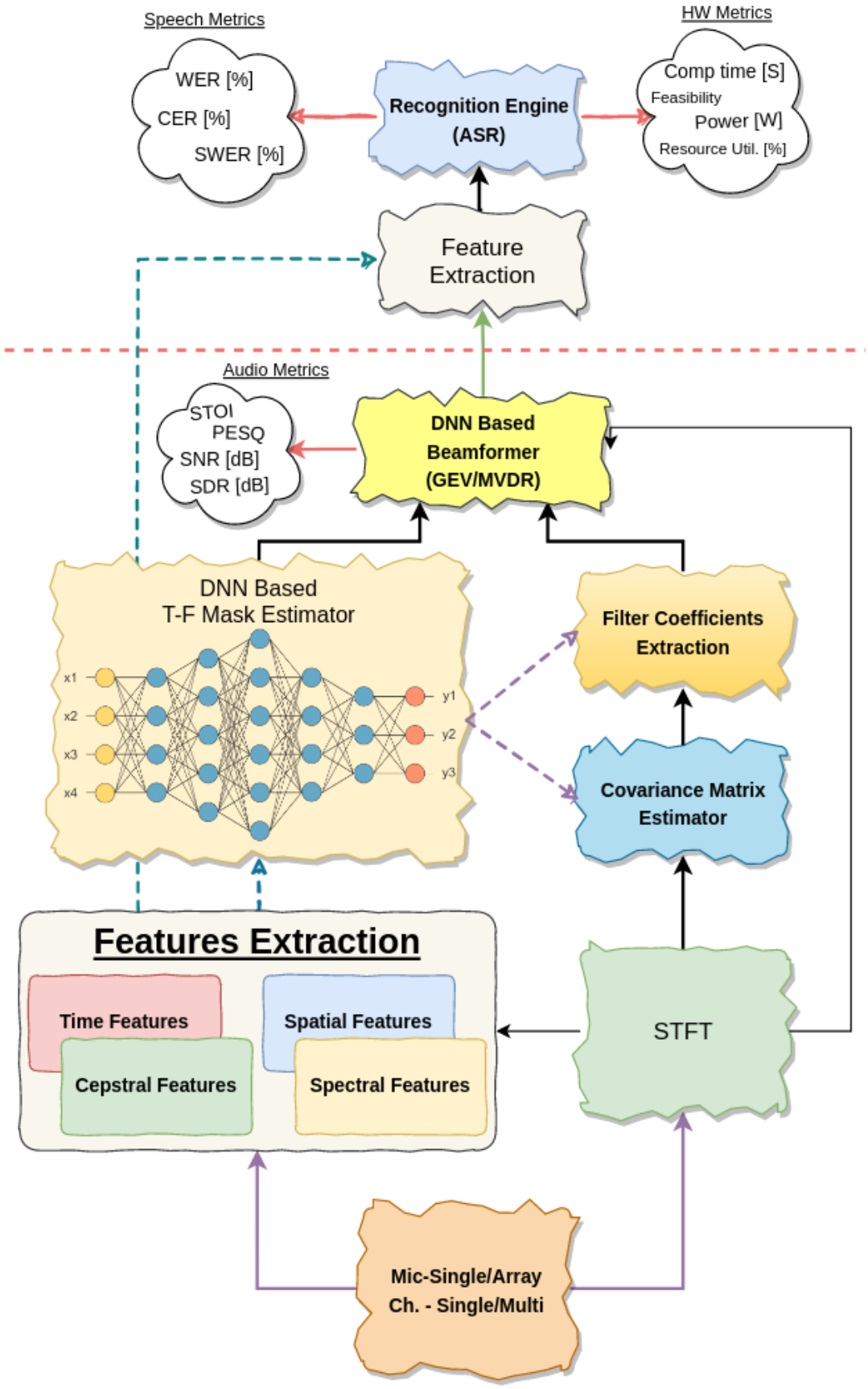


Figure 1.3: Project Architecture

This paper's structure follows the architecture presented in Figure 1.3. In Chapter 2, we introduce the selected performance evaluation metrics of interest and how they relate to these domains. A detailed explanation of how each performance metric is extracted and calculated is provided as well for a better understanding of the motivation behind these metrics selections.

Chapter 3 describes three different scaling methods that are commonly used in audio processing and analysis. We also present detailed performance comparisons in this chapter based on the evaluation metrics that we described in Chapter 2.

The understanding of audio frequency scaling and the differences between scaling methods are essential for Chapter 4.

Chapter 4 presents key aspects of feature analysis along with the considerations and importance of every feature. We conclude that chapter indicating that a wise feature selection is essential for the sake of accurate speech classification and audio processing.

Chapter 5 surrounds Time-Frequency (TF) masking techniques, which is a preliminary process taken prior to beamforming. In this chapter, we cover four dominant masking techniques, starting with background theory, through implementation on to measured performance evaluations. The T-F masking outputs are dynamically estimated by a DNN subsystem and serve as the input weights to a beamformer that is connected next.

Beamforming concepts and common beamforming architectures for speech are described in Chapter 6. This chapter follows the T-F masking chapter since the beamformer actually works on the outputs that are produced by the T-F

masking DNN system.

General background about ASR systems is given in Chapter 7. We then go through the history of the development of ASR systems to the E2E ASR systems. We emphasize the benefits of using E2E solutions and what other advancements were done in this field of research. Here, we present our general approach of an ASR engine implementation, followed by multiple variants of suggested ASR engines and their measurement evaluations.

Chapter 8 describes the datasets that we use for training and evaluations of our various DNN systems. We make use of the multi-microphone CHiME datasets [1] for the T-F masking and beamforming parts, and the Common-Voice V7 dataset [4] for the ASR engine module.

In Chapter 9, we state our conclusions for this study and our plan for future work.

1.3 References

[1] Jon Barker et al. *CHiME3 Speech Corpus*. 2017. URL: https://catalog.ldc.upenn.edu/LDC2017S24 (cit. on pp. 5, 11).

[2] Christoph Boeddeker et al. "Exploring Practical Aspects of Neural Mask-Based Beamforming for Far-Field Speech Recognition". In: *ICASSP 2018*. IEEE, 2018. URL: https://www.microsoft.com/en-us/research/publication/exploring-practical-aspects-neural-mask-based-beamforming-far-field-speech-recognition/ (cit. on p. 2).

[3] W. Chan et al. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 4960–4964. DOI: 10.1109/ICASSP.2016.7472621 (cit. on p. 5).

[4] Mozilla Corporation. *Mozilla CommonVoice Dataset*. 2022. URL: https://commonvoice.mozilla.org/en (cit. on p. 11).

[5] John S. Garofolo et al. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. 1993. URL: https://catalog.ldc.upenn.edu/LDC93s1 (cit. on p. 5).

[6] Awni Hannun. "Sequence Modeling with CTC". In: *Distill* (2017). DOI: 10.23915/distill.00008 (cit. on p. 5).

[7] J. Heymann, L. Drude, and R. Haeb-Umbach. "Neural network based spectral mask estimation for acoustic beamforming". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 196–200. DOI: 10.1109/ICASSP.2016.7471664 (cit. on pp. 4, 6).

[8] G. Hinton et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97. DOI: 10.1109/MSP.2012.2205597 (cit. on p. 1).

[9] Nicholas Dobinson Jack Harvie-Clark. "The practical application of G and C50 in classrooms". In: *Internoise* (2013). DOI: 10.23915/distill.00008 (cit. on p. 3).

[10] W. Jiang, F. Wen, and P. Liu. "Robust Beamforming for Speech Recognition Using DNN-Based Time-Frequency Masks Estimation". In: *IEEE Access* 6 (2018), pp. 52385–52392. DOI: 10.1109/ACCESS.2018.2870758 (cit. on pp. 4, 6).

[11] B. Juang and Lawrence Rabiner. "Automatic Speech Recognition - A Brief History of the Technology Development". In: (2005) (cit. on p. 1).

[12] Jinyu Li et al. *On the Comparison of Popular End to End Models for Large Scale Speech Recognition*. 2020. arXiv: 2011.03110 [eess.AS] (cit. on pp. 4, 5).

[13] Y. Luo et al. "FaSNet: Low-Latency Adaptive Beamforming for Multi-Microphone Audio Processing". In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2019, pp. 260–267. DOI: 10.1109/ASRU46091.2019.9003849 (cit. on pp. 4, 5).

[14] Z. Meng et al. "Deep long short-term memory adaptive beamforming networks for multichannel robust speech recognition". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 271–275. DOI: 10.1109/ICASSP.2017.7952160 (cit. on p. 4).

[15] Psarras Sotirios et al. "Measurements and Analysis of the Epidaurus Ancient Theatre Acoustics". In: (2013) (cit. on p. 3).

[16] Xiaofei Wang et al. *Exploring End-to-End Multi-channel ASR with Bias Information for Meeting Transcription*. 2020. arXiv: 2011.03110 [eess.AS] (cit. on pp. 4, 5).

[17] X. Xiao et al. "Deep beamforming networks for multi-channel speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 5745–5749. DOI: 10.1109/ICASSP.2016.7472778 (cit. on pp. 1, 4).

Chapter 2

E2E Evaluation Metrics

2.1 Audio Metrics

2.1.1 SNR – Signal to Noise Ratio

The signal-to-noise ratio (SNR) metric evaluates how distinct the desired signal is out of the overall noise.

Let $y(t)$ denote a time-domain signal consisting of the desired speech signal $x(t)$, and some interferences, referred to as noise $n(t)$. That signal is given by:

$$y(t) = x(t) + n(t) \tag{2.1}$$

Ideal speech separation of a noisy mixture signal is characterized by a perfect match between the predicted speech signal, $\widehat{x}(t)$, and the original (reference) speech signal $x(t)$.

Properly modeling the problem, we can optimize it using the MSE (Mean Square Error) loss function, also noted as the L2 function. The L2 loss function

is given in Equation 2.2:

$$\ell(\widehat{x}, x) = \sum_{t=0}^{T-1} [\widehat{x}(t) - x(t)]^2 \tag{2.2}$$

$$= \sum_{t=0}^{T-1} |r(t)|^2 \tag{2.3}$$

The term $\sum_t |r(t)|^2$ is the total energy of the residual error between the predicted signal and the desired target speech, which is related to the additive noise.

First, let's break $\widehat{x}(t)$ to its fundamental components [9].

$$\widehat{x}(t) = x_s + e_{noise} + e_{interf} + e_{artif} \tag{2.4}$$

Where x_s stands for the part of $\widehat{x}(t)$ coming from the wanted source(s), and e_{noise} represents the part of $\widehat{x}(t)$ coming from the sensor's noise. The sensor can be the microphone itself or one of its counterparts. e_{interf} denotes the unwanted sources presented in $\widehat{x}(t)$, and the e_{artif} represents any other artifacts that cause distortions in the prediction of x_s.

According to Parseval's theorem, the total residual energy in time equals the sum of the spectral power in the frequency domain resulting from the squared difference between the magnitudes of the predicted and target speech [4].

Since the residual energy over time is referred to as the noise power, minimizing the residual, which is minimizing the MSE loss function, translates

into an increase in SNR.

$$\sum_{t} |r(t)|^2 = \sum_{\tau=0}^{T-1} \sum_{f=0}^{T-1} \left[\widehat{X}(\tau, f) - X(\tau, f) \right]^2 \tag{2.5}$$

The SNR is therefore given by:

$$\begin{aligned} SNR &= 10 \log_{10} \left(\frac{\|x_s\|^2}{\|\widehat{x} - x_s\|^2} \right) \\ &= 10 \log_{10} \left(\frac{\|x_s\|^2}{\|r\|^2} \right) \end{aligned} \tag{2.6}$$

2.1.2 SI-SNR – Scale Invariant SNR

To ensure that the SNR is amenable to scale invariance [7], both the target and estimated signals are normalized to zero-mean.

$$\begin{aligned} SI - SNR &= 10 \log_{10} \left(\frac{\|x_s - \mathbf{E}[x_s]\|^2}{\|(\widehat{x} - \mathbf{E}[\widehat{x}]) - (x_s - \mathbf{E}[x_s])\|^2} \right) \\ &= 10 \log_{10} \left(\frac{\|x_{AC}\|^2}{\|\widehat{x}_{AC} - x_{AC}\|^2} \right) \\ &= 10 \log_{10} \left(\frac{\|x_{AC}\|^2}{\|r_{AC}\|^2} \right) \end{aligned} \tag{2.7}$$

2.1.3 Segmental SNR

An SNR evaluation is basically the ratio between the overall energies of the signal and those in the noise. However, some portions of the signal are almost pure noise, especially in the case of speech signals, where there are gaps between phonemes, articulation stops, and air aspiration breaks. As a result,

the SNR calculation may be impacted, and it depends on the length of the empty sections with respect to the length of the other sections where speech is present.

With Segmental SNR [5], instead of taking the entire signal, the signal is segmented to relatively small chunks (segments), each in length usually set to a frame of (typically) 25 *ms* long with the option of setting an overlap between segments. Per segment, the SNR is calculated, and then averaged across all the segments. If the energy of the speech reference in a segment is below some threshold or duration, that segment is negligible and is excluded, thus limiting the evaluation only to sections where significant speech is present.

Equation 2.6 can be rewritten as:

$$SEG - SNR = \frac{1}{M} \sum_{m=1}^{M} 10 \log_{10} \left(\frac{\|x_s\|^2_{(m)}}{\|r\|^2_{(m)}} \right) \tag{2.8}$$

Where M denotes the number of segments the signal is divided by.

Despite being more accurate for speech signals, Segmental SNR suffers from a limitation that can affect the actual results severely. In speech enhancement evaluations, the signal's predicted (enhanced) version is compared to a clean reference signal concerning the noisy mixture.

Unfortunately, speech analysis for the extraction of the Segmental SNR causes misalignment in time due to a lack of common reference time between the reconstructed signal and the clean reference. Moreover, the reconstructed signal is not aligned with the noisy mixture either. These misalignments are a side effect of the time-domain to the frequency-domain transformation, the processing manipulations on the transformed signal, and the reconstruction

of the signal in the time-domain using the inverse-transform technique. Therefore, without any alignments, extraction of the Segmental-SNR is meaningless and most probably inaccurate. Due to that limitation, an alignment process should be applied prior to taking the Segmental SNR calculation. These alignments usually have a small marginal error that spans over a few sampling points.

2.1.4 STOI – Short-Time Objective Intelligibility

STOI [8] is a metric that is used to evaluate the intelligibility of a speech signal. The intelligibility is measured by taking the correlation coefficient between the temporal envelopes of the clean and degraded speech. In our case, the term degraded might be confusing since the degraded speech input is actually the outcome of the beamformer following the T-F masking at the front-end. However, relative to the clean speech, the beamformer's output is indeed degraded, although it is considered an enhanced version of the noisy mixture.

The naming convention *Short-Time* comes from the time frame length of the overlapping segments, which is $384ms$.

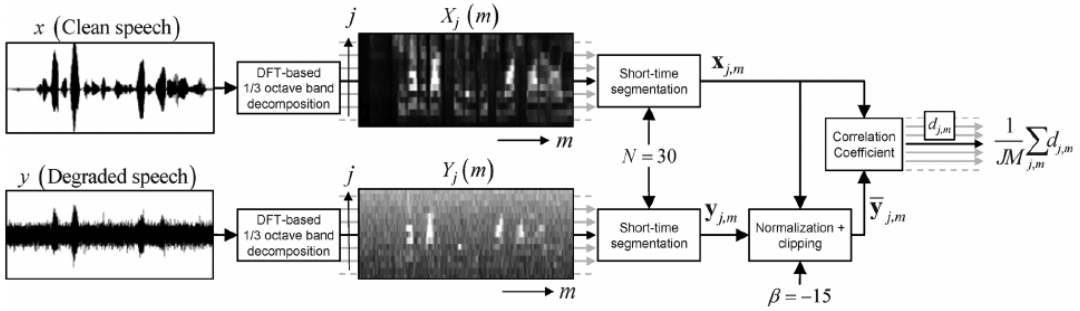


Figure 2.1: STOI flow diagram

Source: Adapted from [8]

The STOI algorithm structure is demonstrated in the blocks diagram shown in Figure 2.1.

The short-time temporal envelop of the degraded speech $Y_{j,m}$ is clipped and normalized before the extraction of the correlation coefficient with the short-time temporal envelop of the clean speech $X_{j,m}$. This clipped normalized version then be:

$$\mathcal{Y}[n] = \min \left\{ \frac{||X_{j,m}||}{||Y_{j,m}||} Y_{j,m}[n],\ (1 + 10^{-\beta/20}) X_{j,m}[n] \right\} \tag{2.9}$$

Thus, the correlation coefficient can be expressed as the distance given in Equation 2.10.

$$d_{j,m} = \frac{(X_{j,m} - \bar{X}_{j,m})^{tr} \cdot (\mathcal{Y}_{j,m} - \bar{\mathcal{Y}}_{j,m})}{||X_{j,m} - \bar{X}_{j,m}|| \cdot ||\mathcal{Y}_{j,m} - \bar{\mathcal{Y}}_{j,m}||} \tag{2.10}$$

Also, defining the intermediate intelligibility measure, Equation 2.10, it stands for the m^{th} time frame. Extending it to form a definition for the entire signal, we can take the average of $d_{j,m}$ as in Equation 2.11.

$$d = \frac{1}{JM} \sum_{j,m} d_{j,m} \tag{2.11}$$

Where J presents the total number of one-third octave bands, and the averaging overlaps M number of time frames.

2.1.5 PESQ – Perceptual Evaluation of Speech Quality

PESQ [6] is a measuring method adopted by the ITU (International Telecommunication Union, ITU-T P.862) to test the speech quality of telephony and

mobile stations.

This measuring metric evolved from different previous measuring techniques such as Bark Spectral Distortion (BSD), Perceptual Analysis Measurement System (PAMS), and Perceptual Speech Quality Measure (PSQM).

The motivation behind the development of the PESQ metric was the need to assess the speech quality in an E2E communication channel that considers the entire link rather than particular parts.

The evaluation of a speech signal quality by PESQ follows the MOS (Mean Opinion Scores) scoring model, where the actual speech quality is ranked between 1 to 5 by a group of listeners. The MOS is a subjective measure, while the PESQ is an objective measure.

Figure 2.2 shows the data flow of the PESQ computation for a predicted signal, with respect to the clean reference.

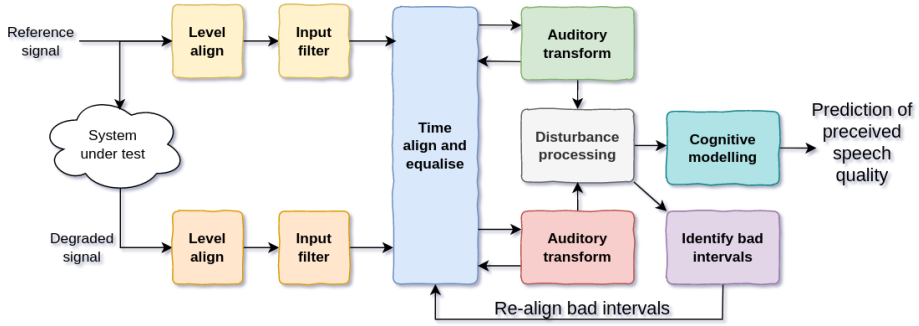


Figure 2.2: PESQ Algorithm Blocks Diagram

Source: Adapted from PESQ paper [6] and redesigned

2.2 ASR Metrics

2.2.1 WER – Word Error Rate

WER [3] metric is probably the most used evaluation technique for speech recognition systems.

Evaluation of this metric occurs at the ASR engine's output, where the predicted text is segmented into sentences. Each word in the predicted text is then matched with its counterpart in the annotated reference transcript. The sum of mismatches between a predicted sentence and the reference, divided by the total counted words in the reference, indicates the WER.

However, in some cases, the predicted sentences differ in size compared to the reference. Therefore, special care for *Insertions* and *Deletions* should be carried out as well, without neglecting the detected *Substitutions*.

The WER is described by:

$$WER = \frac{S + D + I}{N} \tag{2.12}$$

Where N is the total count of words in the reference, and S, D, I are the number of *Substitutions* (wrong word detection), *Deletions* (Omitting words), and *Insertions* (Wrong words insertions).

2.2.2 CER – Character Error Rate

CER is another metric with some similarities to the WER evaluation metric but with a narrower resolution. The change in resolution is due to comparing characters instead of words. The same rules of *Substitutions*, *Deletions*, and

Insertions apply, and therefore, the calculation of the CER is the same:

$$CER = \frac{S + D + I}{N} \tag{2.13}$$

In many cases, the CER [10] is a complementary measuring metric to the WER metric. This extra measure shines especially when there is a need to get a full perspective with a greater differentiation capability of the *Substitutions* in the complete sentence of the suggested predicted transcript, also known as the hypothesis. While WER counts a mismatch between the reference and the predicted word, even in cases where only single characters or worse, punctuation marks are not correctly placed, CER can lead to more accurate grading per word.

2.3 Hardware Metrics

2.3.1 Power Estimation

Electrical circuits, components, and systems require power to function. The amount of power a device consumes from the power sources is subject to various parameters and mainly describes the rate of energy delivery from the source to the device or vice versa.

Due to the nature of conducting materials, whenever an electrical potential is applied between the conductor's terminals, electrical current goes through the conductor. The current that flows in the system feeds the different components with energy. However, the total supplied energy is not purely consumed over time, and some energy is lost and wasted due to power dissipation.

Power dissipation is a side effect of a conductors' resistive nature, which "resists" the transition of current through it. As a result, part of the energy in the system is converted to heat energy.

Since dissipated power is a waste of energy it is also considered as one of the main causes to electronic systems' performance degradation at high temperatures. Therefore, engineers want to mitigate as much as possible any dissipated power that is not used for the main functionality of the system. For that end, power analysis is crucial in any system design phase to ensure efficiency and correctness while maintaining robustness over time and under different working conditions.

Electrical circuit power dissipation depends on many arguments. However, in general, it can be modeled accurately according to three scenarios divided into two main groups:

1. Static Power
    - Intrinsic Leakage Power
2. Dynamic Power
    - Internal Power
    - Switching Power

2.3.1.1 Intrinsic Leakage Power

Leakage power is the power that dissipates due to the structure of a CMOS device, where a thin layer of metal oxide isolates between the semiconductor material and the gate metal and thus forming a capacitor. Leakage power

dissipates statically regardless of the CMOS device state, whether it is the active state or the off state (idle).

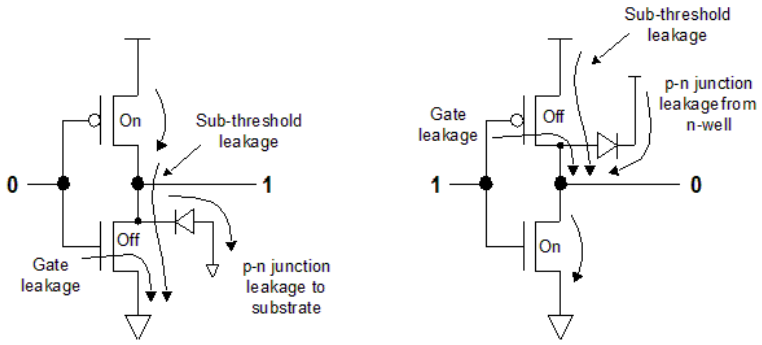


Figure 2.3: Leakage Power Illustration

Source: Adapted from Synopsys PrimePower Suit documentation [2]

Figure 2.3 describes three current leakages, the reverse bias current of the diode (p-n junction), sub-threshold current leakage, and the gate leakage.

With the recent advancement in process technologies, CMOS devices are minimized in size, but the leakage power is increasing as a side effect.

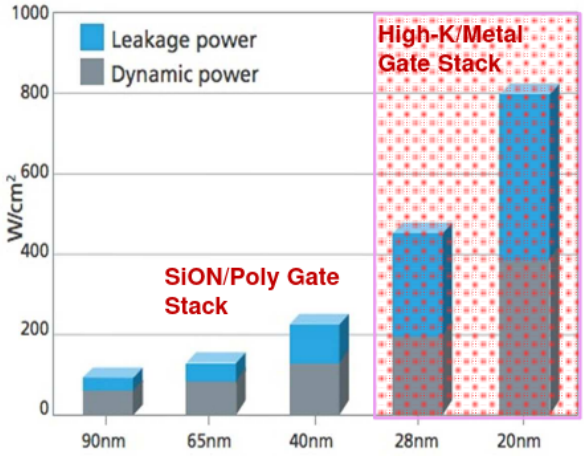


Figure 2.4: Leakage Power vs. Process Technology

Source: Adapted from Soitec FinFet presentation [1]

The overall leakage power is a function of the total number of voltage sources and their voltage levels, the physical dimensions of the CMOS device, and the threshold value set to switch between on-off states.

2.3.1.2 Internal Power

A CMOS device is a formation of two complementary MOS transistors, a p-type and an n-type, formed together as a symmetrical pair unit. Internal power dissipation happens due to the structure of CMOS devices. Whenever a transition at the CMOS gate occurs, both the NMOS and the PMOS drivers are active for a relatively small duration of time. As a result, a short circuit is formed directly from the power rail to the ground. Although not lasting for long periods of time, the amount of internal dissipated power in highly toggled designs becomes significant over time. To minimize the internal dissipated power, or in other words, minimizing the time duration where both devices are active and current flows from V_{dd} to GND, the transition times (both rising and falling) are set to be very fast.

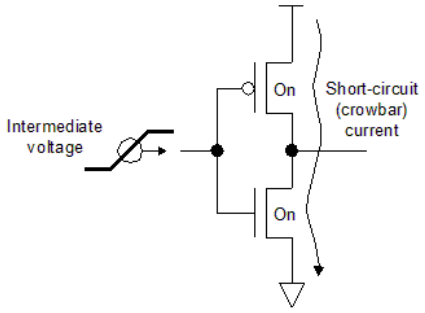


Figure 2.5: Internal Power Illustration

Source: Adapted from Synopsys PrimePower Suit documentation [2]

2.3.1.3 Switching Power

Switching power is the power dissipated as a result of charging and discharging loads during transitions. MOS devices introduce capacitance at their input gates due to their structure. Thus, whenever a low-to-high transition at the output occurs, the driver pushes the current to charge the capacitive load in order to set the desired logic level voltage. Likewise, the load capacitance discharge and sink into the device through the PMOS transistor to the ground for a high-to-low transition at the output. As a result, the charging and discharging currents eventually dissipate and are not delivered to the external load.

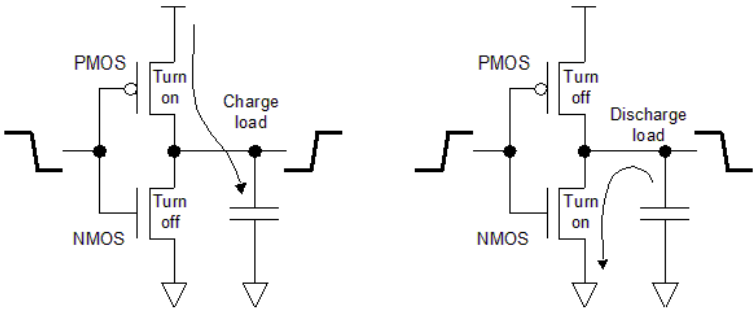


Figure 2.6: Switching Power Illustration

Source: Adapted from Synopsys PrimePower Suit documentation [2]

2.4 References

[1] Sung-Mo (Steve) Kang and Yusuf Leblebici. *CMOS Digital Integrated Circuits Analysis and Design*. 3rd ed. USA: McGraw-Hill, Inc., 2002. ISBN: 0072460539 (cit. on p. 24).

[2] M. Keating et al. "Low power methodology manual: For system-on-chip design". In: (2007), pp. 1–300. DOI: 10.1007/978-0-387-71819-4 (cit. on pp. 24–26).

[3] Dietrich Klakow and Jochen Peters. "Testing the correlation of word error rate and perplexity". In: *Speech Communication* () (cit. on p. 21).

[4] Shan Liang et al. "The analysis of the simplification from the ideal ratio to binary mask in signal-to-noise ratio sense". In: *Speech Communication* 59 (2014), pp. 22–30. DOI: 10.1016/j.specom.2013.12.002 (cit. on p. 15).

[5] Schuyler Reynier Quackenbush. "Objective Measures of Speech Quality (Subjective)". PhD thesis. USA, 1985 (cit. on p. 17).

[6] A.W. Rix et al. "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs". In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. Vol. 2. 2001, 749–752 vol.2. DOI: 10.1109/ICASSP.2001.941023 (cit. on pp. 19, 20).

[7] Jonathan Le Roux et al. *SDR - half-baked or well done?* 2018. arXiv: 1811.02508 [cs.SD] (cit. on p. 16).

[8] Cees H. Taal et al. "A short-time objective intelligibility measure for time-frequency weighted noisy speech". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 4214–4217. DOI: 10.1109/ICASSP.2010.5495701 (cit. on p. 18).

[9] E. Vincent, R. Gribonval, and C. Fevotte. "Performance measurement in blind audio source separation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.4 (2006), pp. 1462–1469. DOI: 10.1109/TSA.2005.858005 (cit. on p. 15).

[10] Y. Wang, A. Acero, and C. Chelba. "Is Word Error Rate a Good Indicator for Spoken Language Understanding Accuracy". In: (2003) (cit. on p. 22).

Chapter 3

Audio Frequency Scaling methods

3.1 Introduction

The human hearing system detects acoustic vibrations and translates those vibrations into sounds. The detectable range of frequencies by the human ear is referred to as audio or sonic. This range spans over approximately 20 kHz, starting at 20 Hz to about 20 kHz [7].

As a result of aging, the hearing system's dynamic range as to the detectable bandwidth decreases, and by middle-age are set at about 20 Hz to 14 KHz [8]. That is, the maximum hearable frequency declines with age.

The human ear's ability to distinguish between two different frequencies is not symmetric. For example, the spectral distance between two different frequencies in one distinct region does not equal the spectral distance between two additional frequencies in other regions. Due to that asymmetry, the conventional linear spectral mapping is impractical for speech analysis applications. Thus, a different spectral mapping based on a different scaling system that mimics the human hearing as possible is applied.

3.2 Mel-Scaling

The Mel-scaling method is a suggested solution to mapping standard audio frequencies to perceived frequencies. The basic idea that lies underneath it is that for different pitches we assign varying bandwidth, such that they are equal in distance from each other, as rated by listeners. The reference point has been chosen to be 1000 $Hz = 1000$ $Mels$.

Mel-scale was first described in [4] by Stevens and Volkmann, where the authors presented different curves for Mel-scaling.

Two common tables were composed according to the Mel-scaling curves. One table by Beranek in 1949 [1] and the second by Umesh et al. in 1999 [6].

The most popular equation that models the Mel-scale is typically referenced as the "Logarithm based Mel scale" [2]:

$$Mel = \ln\left(1 + \frac{f}{700}\right) \cdot \frac{1000}{\ln(1 + \frac{1000}{700})} \tag{3.1}$$

Equation [3.1] can be simplified as follows:

$$Mel = 1127 \ln\left(1 + \frac{f}{700}\right)$$

$$Mel = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.2}$$

Then, the reverse equation, converting Mels back to Hz, can be written as:

$$f[Hz] = 700\left(10^{\frac{Mel}{2595}} - 1\right) \tag{3.3}$$

3.2.1 Mel-scale approximations

Computing a logarithm for hardware devices, whether it is the natural logarithm or any other base, is not very straightforward. For example, this kind of computation might require special techniques or long LUTs (look-up tables), which are extraordinarily resource hungry.

Instead, other approximations that do not involve trigonometric or logarithms, but only simple arithmetic structures can be applied. By doing so, we benefit from low resource utilization while maintaining high accuracy.

Multiple approximation methods were studied in [6]. Two approximations are the most prominent for target HW devices.

$$Mel = a + b \cdot f \tag{3.4}$$

$$Mel = \frac{f}{a \cdot f + b} \tag{3.5}$$

Where a, b in Equation 3.4 are defined as follows:

$$a = \begin{cases} 127.7 & , f \leq 1000 \\ 1322 & , f > 1000 \end{cases}$$

$$b = \begin{cases} 0.9 & , f \leq 1000 \\ 0.19 & , f > 1000 \end{cases} \tag{3.6}$$

while in Equation 3.5 a, b are:

$$a = \begin{cases} 0.000244 & , f \leq 1000 \\ 0.0004 & , f > 1000 \end{cases}$$

$$b = \begin{cases} 0.741 & , f \leq 1000 \\ 0.603 & , f > 1000 \end{cases} \tag{3.7}$$

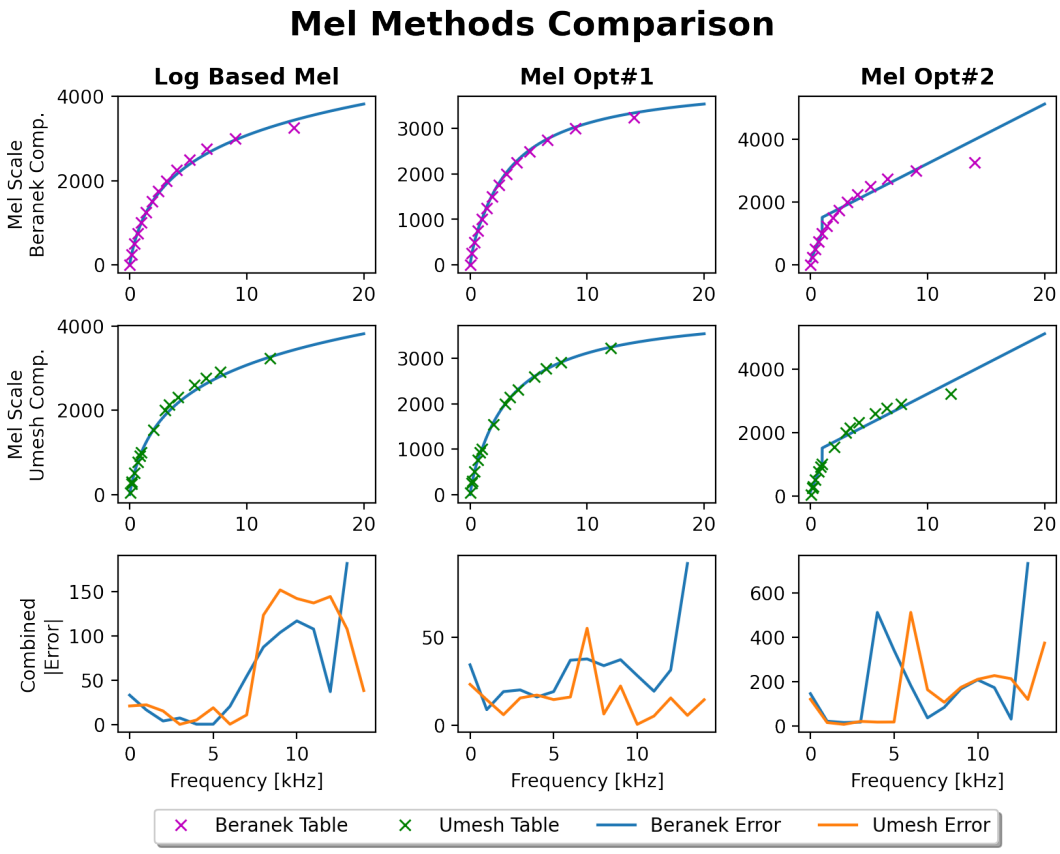


Figure 3.1: Mel-scale comparisons with Beranek & Umesh tables

Figure 3.1 has comparisons of three different Mel-scale implementations with Beranek & Umesh tables. The first column, Log Based Mel, represents O'Shaugnessy's famous log-based Mel modeling. The Mel option #1, and Mel

option #2 columns follow the suggested approximations given in Equations 3.4 and 3.5, respectively.

From the last row of graphs, we can deduce that the approximation in Equation 3.5 is the closest along with the range of audio frequencies to the tables provided by Beranek & Umesh. On the other hand, the more simplified approximation in Equation 3.4 seems to yield the highest errors.

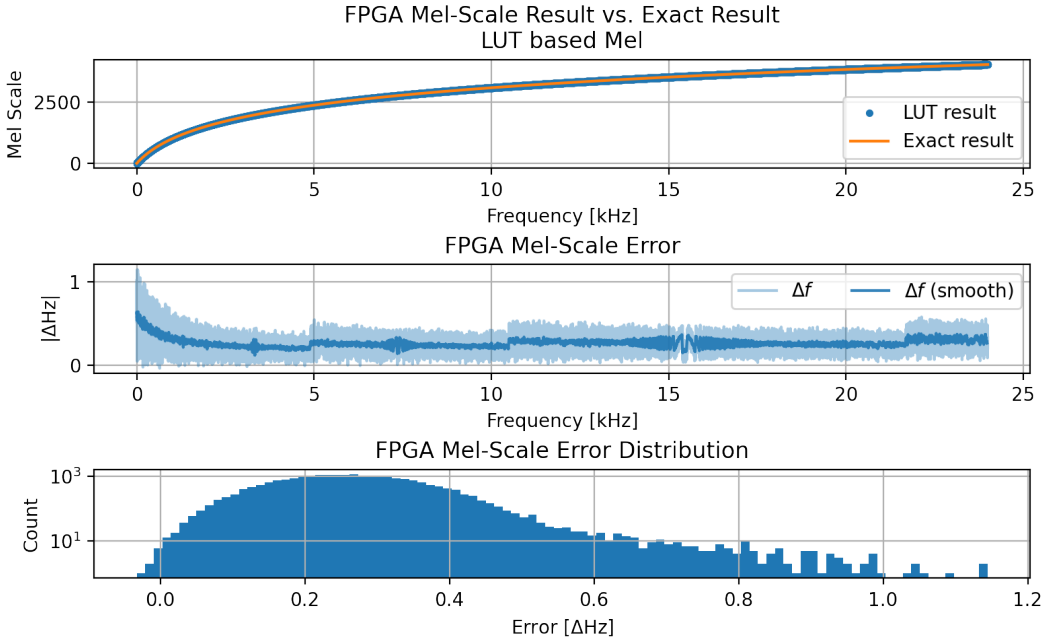


Figure 3.2: LUT based Mel FPGA implementation results

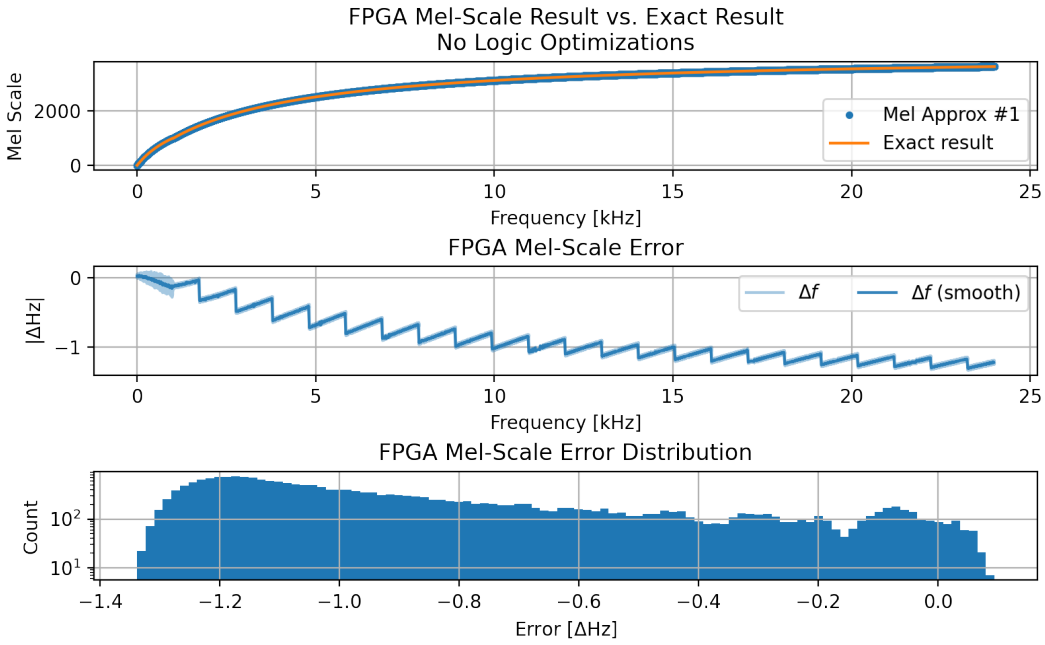


Figure 3.3: Mel approx. #1 FPGA implementation results

Figures 3.2, 3.3 present the results of FPGA implementations of O'Shaugnessy's log-based Mel and Mel approximation #1. Nonetheless, a high precision quantization, U32.22 was chosen for the Mel approximation, where the shifting error received is higher when compared to the conventional Log Mel scaling implementation. Although this error shift is compensated just by selecting the approximation method, the straightforward approach turned out to be the non-optimized solution in terms of HW resources and power consumption, which utilized four times higher wattage on top of $25-30\%$ additional resources.

Instead, two optimization workarounds were tested. The first is the multiplication of the a, b coefficients in Equation 3.5 by 1000. The second optimization is reorganizing the equation and storing the result in a sufficient precision

structure in memory for the fractional part but lower resolution for the integer part. These optimizations lead to a reduction in the required number of bits for the fractional part. As a result, both the frequency shifting error and the overall resource utilization are greatly improved.

Yet, the split in frequency bands results in two multiplied sets of coefficients for each band calculation. Therefore, choosing a more generalized set of coefficients for the entire audio band can help in the reduction of redundant LUTs and other combinational logic, such as selectors and multi-bus multiplexer cells.

Selection of $a = 0.24,\ b = 0.741$, showed better results as can be seen in Figure 3.5. The accuracy estimation for the non-generic implementation is shown in Figure 3.4.

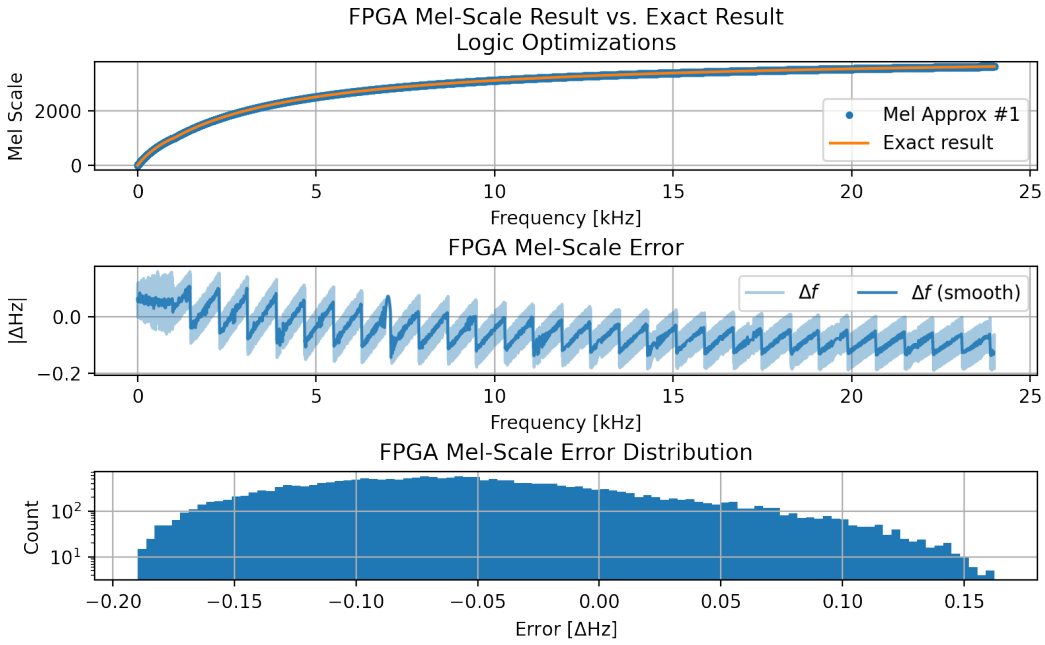


Figure 3.4: Mel approx. #1 optimized FPGA implementation results

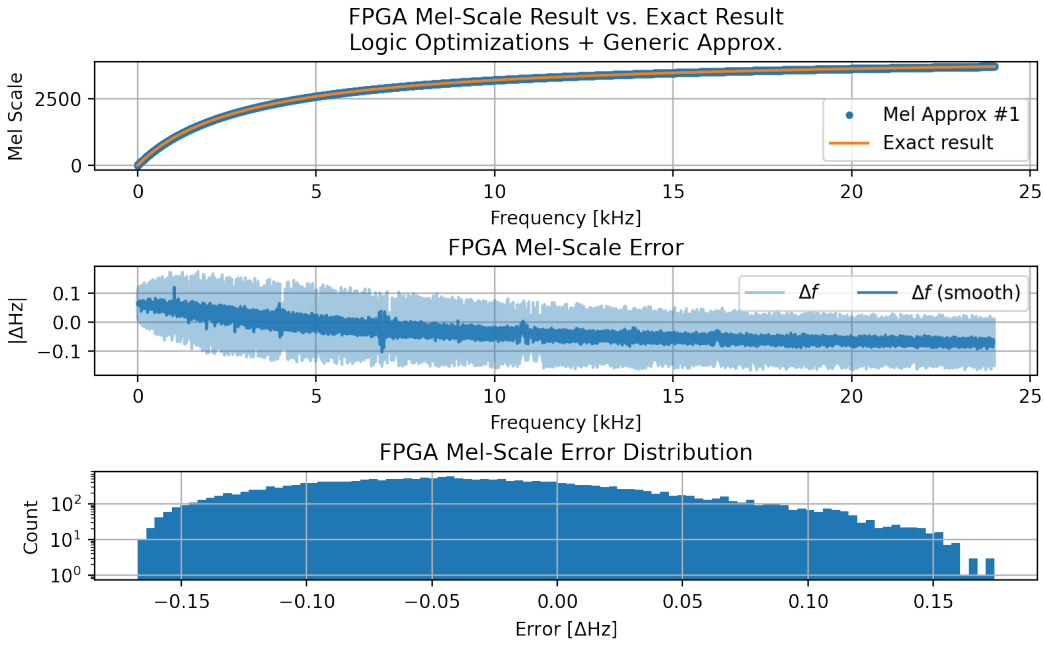


Figure 3.5: Mel approx. #1 optimized, generic FPGA implementation results

Algorithm	Latency $[ns]$	Quant.	Max Err. $[\Delta Hz]$	Mean Err.	Std Err.
Log LUT Mel	14 (3 C.C)	U16/4	1.146	0.268	0.106
Mel #1	9 (2 C.C)	U32/22	1.338	-0.873	-0.359
Mel #1 Opt	9 (2 C.C)	U16/4	0.190	-0.053	0.065
Mel #1 Generic	9 (2 C.C)	U16/4	0.174	-0.035	0.061

Table 3.1: Mel-Approx, log-based Mel performance comparison

Algorithm	FF	LUT	DSP	LUTRAM	BRAM
Log LUT Mel	82(0.04%)	276(0.23%)	1(<1%)	10(0.02%)	1.5(1.04%)
Mel #1	47(0.02%)	530(0.45%)	1(<1%)	0(0%)	0(0%)
Mel #1 Opt	47(0.02%)	271(0.23%)	1(<1%)	0(0%)	0(0%)
Mel #1 Generic	47(0.02%)	269(0.19%)	1(<1%)	0(0%)	0(0%)

Table 3.2: Mel scaling methods resource utilization table

Parameter	Log LUT Mel	Mel #1	Mel #1 Opt	Mel #1 Generic
Dynamic Power [W]				
Signals	4.947	5.221	3.011	2.916
Logic	6.50	6.792	3.751	3.070
DSP	0.014	0.013	0.014	0.014
I/O	18.236	26.263	7.207	6.378
$\mathbf{P_{dynmic}}$	**29.697**	**38.290**	**13.983**	**12.379**
Static Power [W]				
PL Static	2.364	2.466	0.499	0.499
PS Static	0.068	0.071	0.020	0.018
$\mathbf{P_{static}}$	**2.432**	**2.537**	**0.519**	**0.517**
Total Power [W]				
$\mathbf{P_{total}}$	**32.13**	**40.827**	**14.502**	**12.896**

Table 3.3: Mel-Approx, log-based Mel, Bark Scale Power consumption

The LUT implementation makes use of a log_2 look-up table. However, an additional step is needed for the natural logarithm or other log bases. Since the logarithm bases are constant, the LUT result is divided by the log_2 of the base, whether it be the natural base or base ten. This logarithm bases convention is described in Equation 3.8.

$$\log_b(a) = \frac{\log_x(a)}{\log_x(b)} \tag{3.8}$$

Table 3.1 summarizes the performance comparison between the different Mel scaling implementation approaches.

The HW setup is for the PYNQ-Z1 development board. Hence, the results are unique to that specific HW device and probably change for other FPGA devices and development boards.

Latencies were simulated with Xilinx Vivado Suite for an operating clock

frequency of 225 MHz. In this operating condition, no timing violations were reported.

Tables 3.2 and 3.3 show the synthesis and implementation results plus the power estimation reports. These reports were taken from the Xilinx Vivado Suite application.

3.3 Bark-Scaling

Another scaling method is the Bark scale which is based on the same principle of retaining perceptual distances. The Bark scale is divided into critical bands corresponding to the critical hearing bands in humans. Each band has a bandwidth similar to the psychoacoustic band of the corresponding "filter" in the human hearing system and is ranked with a unique number.

3.3.1 Bark Critical Bands

Like the Mel scale, several equations were proposed to model best the Bark scale and its critical bands.

The first method was introduced in [9]. A proposed approximation to the Bark scaling is described in [5]; this paper also introduces the correction of the band boundaries to ensure more correctness with the original Bark scaling.

Four different equations were proposed to model the Bark scale.

The first by Zwicker is well described in [9] and is given by the following

equation:

$$Bark = \tan^{-1}\{0.00073f\} + 3.5\tan^{-1}\left\{\left(\frac{f}{7500}\right)^2\right\} \tag{3.9}$$

Zwicker's bark scale and it's corresponding critical bands are shown in Figure 3.6

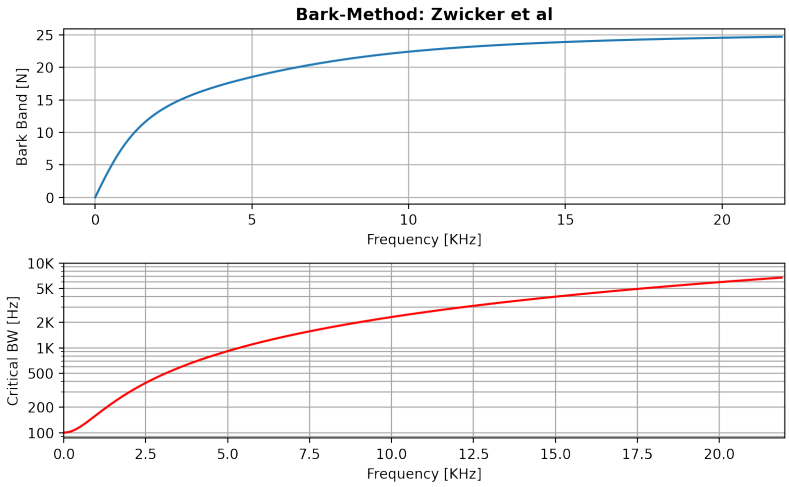


Figure 3.6: Zwicker's bark scale and critical bands

Another proposed equation by Traunmuller [5] is:

$$Bark = \frac{26.81f}{1960 + f} - 0.53 \tag{3.10}$$

Traunmuller's bark scale and it's corresponding critical bands are shown in Figure 3.7

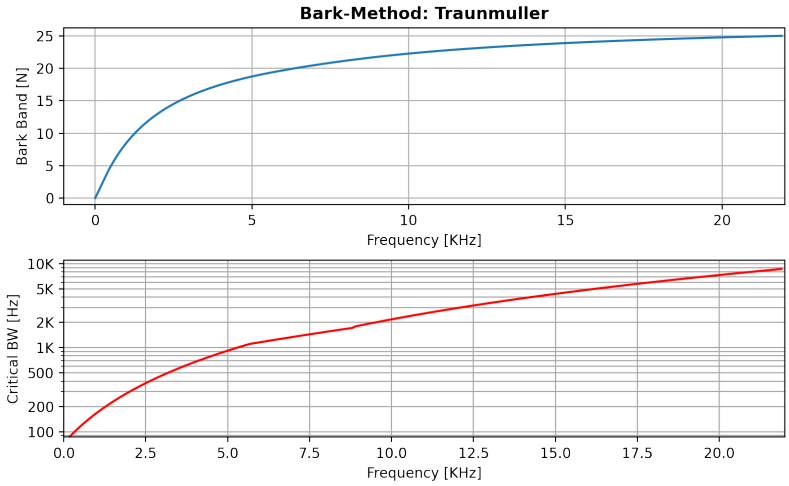


Figure 3.7: Traunmuller's original bark scale and critical bands

Denoting Traunmuller's original Bark scale given in Equation 3.10 as $Bark'$, the fixed form for Traunmuller's Bark equation is:

$$Bark = \begin{cases} 0.3 + 0.85 \cdot (Bark') & , \ Bark' < 2 \\ Bark' + 0.22 \cdot (Bark' - 20.1) & , \ Bark' > 20.1 \end{cases} \tag{3.11}$$

Traunmiller's fixed bark scale and its corresponding critical bands are shown in Figure 3.8

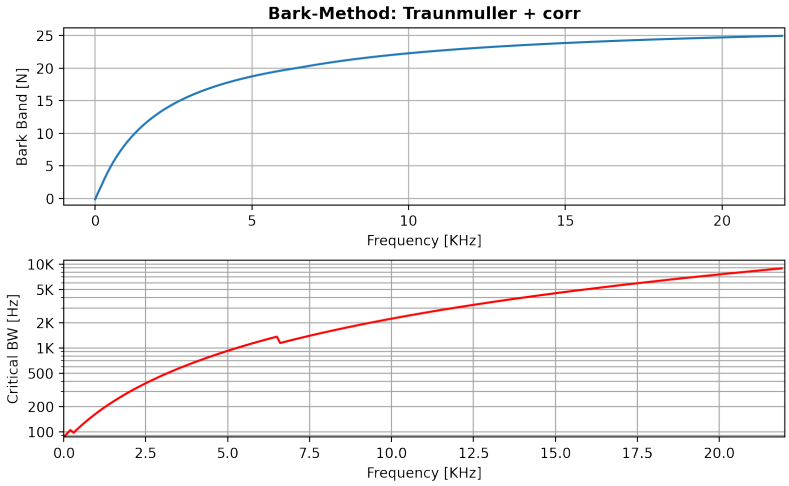


Figure 3.8: Traunmuller's fixed bark scale and critical bands

The fourth possible modeling equation is proposed by Schroeder in [3] and is as follows:

$$Bark = 7 \ln \left(\frac{f}{650} + \sqrt{1 + \frac{f^2}{422500}} \right) \tag{3.12}$$

Schroeder's bark scale and it's corresponding critical bands are shown in Figure 3.9

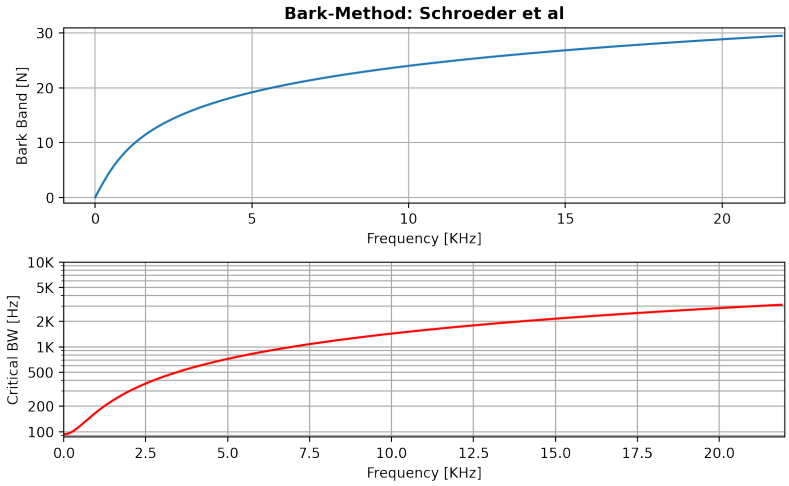


Figure 3.9: Schroeder's bark scale and critical bands

Figure 3.10 shows a comparison between the four bark scaling methods.

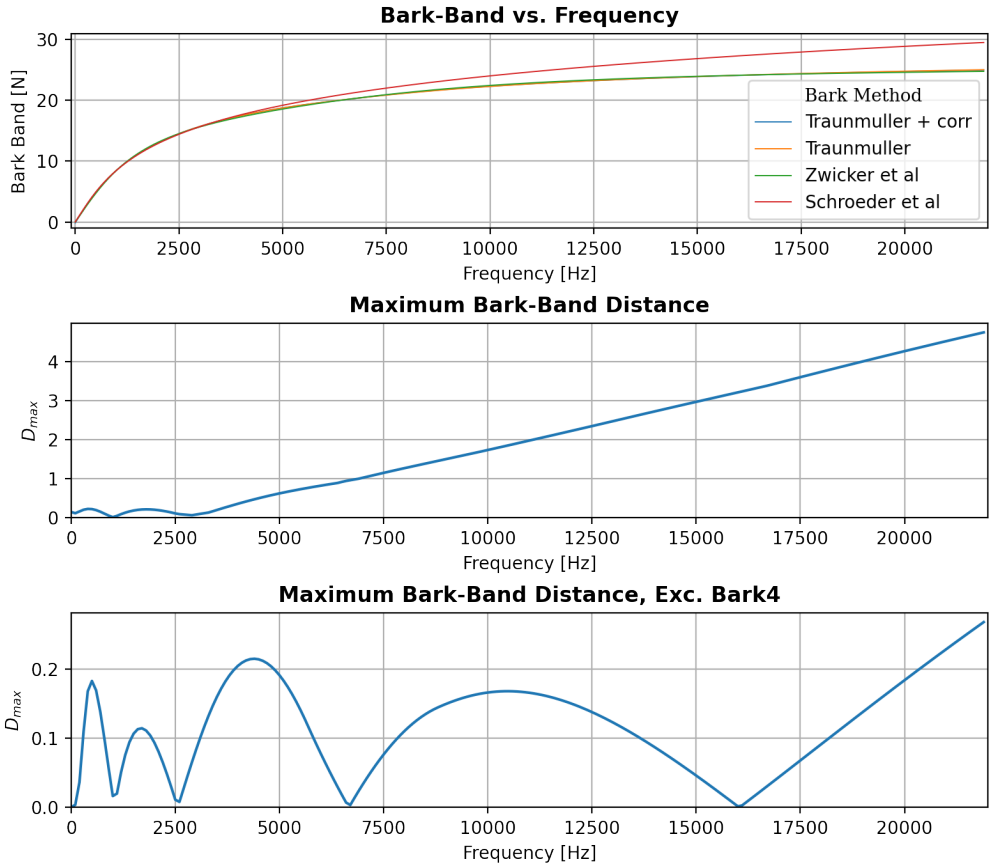


Figure 3.10: Bark Scale comparisons

We see in Figure 3.10 that Schroeder's bark scale has a large error term when compared to the other suggested bark scales. On the other hand, when we omit Schroeder's scale and measure the maximum distance between the other three suggested scales, we see that the error is comparatively lower.

We intend to simplify our implementations in hardware as much as possible. Simplifications are expressed in utilizing less hardware resources. This is possible especially when using basic arithmetic operations. Therefore, we decide to focus on Traunmuller's original bark scale given in Equation 3.10

The accuracy estimation for Traunmuller's original bark scale is shown in Figure 3.11.

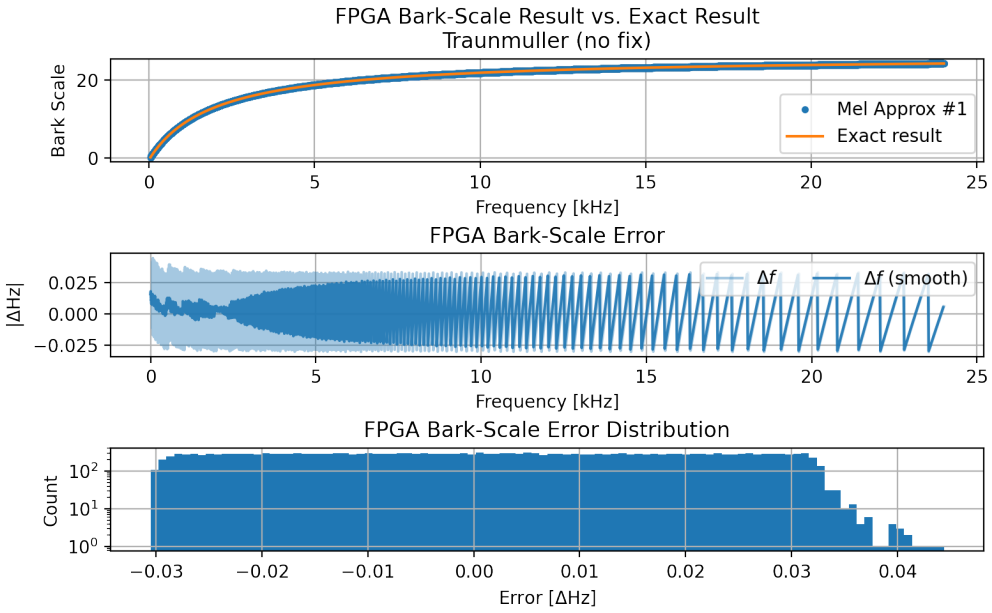


Figure 3.11: Traunmuller's original bark scale FPGA implementation results

Algorithm	Latency $[ns]$	Quant.	Max Err. $[\Delta Hz]$	Mean Err.	Std Err.
Trau. w/ Fix	14 (3 C.C)	U10/5	N.A	N.A	N.a
Trau. w/o Fix	9 (2 C.C)	U9/4	0.0443	0.0015	0.0180

Table 3.4: Traunmuller's Bark scale implementations performance comparison

Algorithm	FF	LUT	DSP	LUTRAM	BRAM
Traunmuller w/ Fix	43(0.02%)	302(0.26%)	1(<1%)	0(0%)	0(0%)
Traunmuller w/o Fix	27(0.01%)	284(0.24%)	1(<1%)	0(0%)	0(0%)

Table 3.5: Traunmuller's Bark scale implementations resource utilization table

Parameter	Traunmuller w/ Fix	Traunmuller w/o Fix
Dynamic Power [W]		
Signals	2.380	2.192
Logic	3.079	2.957
DSP	1.445	0.014
I/O	4.022	4.022
$\mathbf{P_{dynmic}}$	**10.923**	**9.186**
Static Power [W]		
PL Static	0.469	0.437
PS Static	0.017	0.016
$\mathbf{P_{static}}$	**0.486**	**0.453**
Total Power [W]		
$\mathbf{P_{total}}$	**11.412**	**9.639**

Table 3.6: Traunmuller's Bark scale implementations Power consumption

Table 3.4 summarizes the performance comparison between the different Bark scaling implementation approaches.

Tables 3.5 and 3.6 show the synthesis and implementation results plus the power estimation reports. These reports were taken from the Xilinx Vivado Suite application.

3.4 ERB - Equivalent Rectangular Bandwidth

Like the Bark scaling method, the ERB scale aims to rescale the audio spectrum in different bandwidths corresponding to the human hearing "filters".

This approach slightly differs from the Bark scale because the filter is modeled according to a rectangular filter with an equivalent bandwidth.

$$ERBs = 11.17 \ln(47.065 - \frac{676170.42}{f + 14678.5}) \tag{3.13}$$

A proposed approximation is given by:

$$ERBs = 21.4 \cdot \log_{10}(1 + 0.00437f) \tag{3.14}$$

The accuracy estimation for the LUT based ERB scale and the LUT based ERB approximation are shown in Figures 3.12, 3.13, respectively.

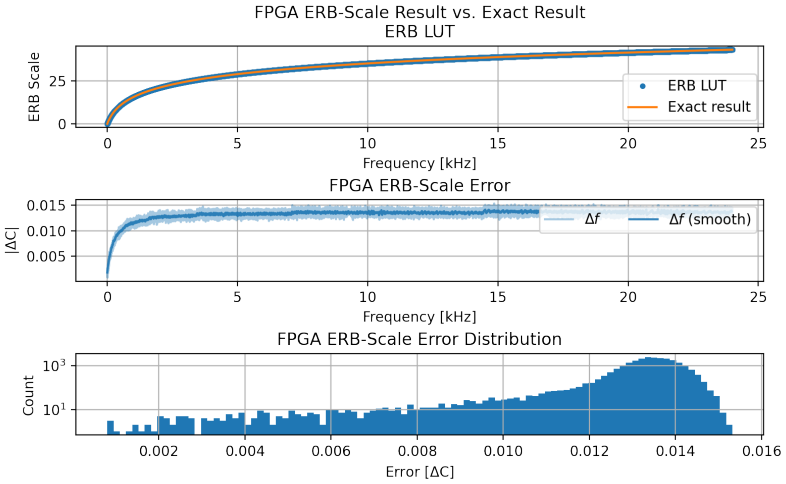


Figure 3.12: LUT based ERB FPGA implementation results

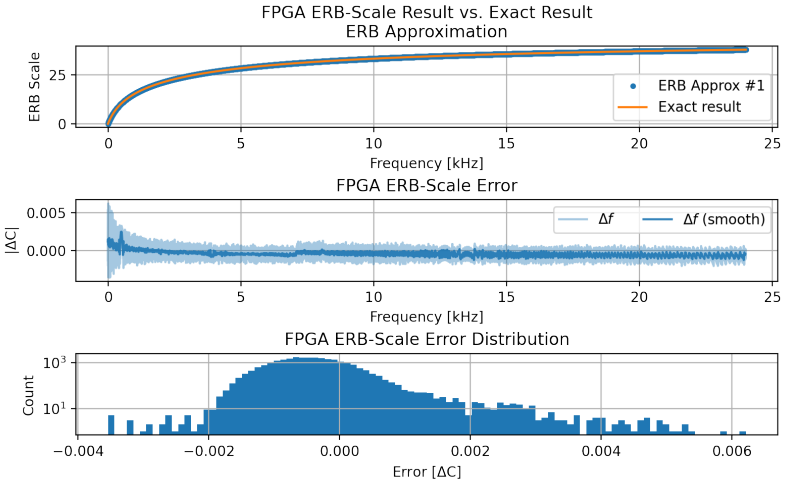


Figure 3.13: LUT based ERB approximation FPGA implementation results

Algorithm	Latency $[ns]$	Quant.	Max Err. $[\Delta C]$	Mean Err.	Std Err.
ERB	23 (5 C.C)	U16/10	0.0153	0.0130	0.0018
ERB Approx	14 (3 C.C)	U16/10	6.210m	-0.408m	0.657m

Table 3.7: ERB and ERB approx performance comparison

Table 3.7 presents the performance comparison between the implementations of the pure ERB and the ERB approximation given by Equations 3.13 and 3.14, respectively. The error terms for the ERBs are measured in ΔC, indicating the error in Cams rather than in frequency units (Hz). Equation 3.15 can be used to convert Cams to Hz.

$$f = \frac{676170.42}{47.065 - e^{0.0895 \cdot C}} - 14678.5 \tag{3.15}$$

Algorithm	FF	LUT	DSP	LUTRAM	BRAM
ERB	112(0.05%)	824(0.70%)	1(<1%)	1(0.02%)	1.5(1.04%)
ERB Approx	103(0.04%)	564(0.48%)	1(<1%)	1(0.01%)	1.5(1.04%)

Table 3.8: ERB scaling methods resource utilization table

Parameter	ERB	ERB Approx
Dynamic Power [W]		
Signals	12.691	5.525
Logic	17.576	7.478
DSP	0.067	0.049
I/O	17.379	17.373
P_{dynmic}	**47.713**	**31.519**
Static Power [W]		
PL Static	2.971	1.510
PS Static	0.084	0.046
P_{static}	**3.055**	**1.556**
Total Power [W]		
P_{total}	**50.768**	**33.075**

Table 3.9: ERB vs. ERB approx Power consumption

Tables 3.8 and 3.9 show the synthesis and implementation results plus the power estimation reports for the ERB implementations.

3.5 Summary

The following tables 3.10, 3.11 and 3.12, provide a summary of the resource utilization, timing and accuracy performances, and lastly the power consumption of each scaling implementation.

Algorithm	FF	LUT	DSP	LUTRAM	BRAM
Log-Based Mel	82(0.04%)	276(0.23%)	1(<1%)	10(0.02%)	1.5(1.04%)
Mel #1 Generic	47(0.02%)	269(0.19%)	1(<1%)	0(0%)	0(0%)
Bark Scale w/ fix	43(0.02%)	302(0.26%)	1(<1%)	0(0%)	0(0%)
Bark Scale w/o fix	27(0.01%)	284(0.24%)	1(<1%)	0(0%)	0(0%)
Approx. ERB	103(0.04%)	564(0.48%)	1(<1%)	1(0.01%)	1.5(1.04%)

Table 3.10: Mel-Approx, log-based Mel, Bark Scale, and approximated ERB resources utilization comparison

Algorithm	Latency $[ns]$	Quant.	Max Err. $[\Delta Hz]$	Mean Err.	Std Err.
Log-Based Mel	14 (3 C.C)	U16/4	1.146	0.268	0.106
Mel #1 Generic	9 (2 C.C)	U16/4	0.174	-0.035	0.061
Bark Scale w/ fix	14 (3 C.C)	U10/5	N.A	N.A	N.A
Bark Scale w/o fix	9 (2 C.C)	U9/4	0.0443	1.5m	0.0015
ERB Approx.	14 (3 C.C)	U16/10	0.292	0.104	0.134

Table 3.11: Mel-Approx, log-based Mel, Bark Scale performance comparison

Parameter	Mel LUT	Mel Gen	Bark w/	Bark w/o	Approx. ERB
Dynamic Power [W]					
Signals	4.947	2.916	2.380	2.192	5.525
Logic	6.50	3.070	3.079	2.957	7.478
DSP	0.014	0.014	1.445	0.014	0.049
I/O	18.236	6.378	4.022	4.022	17.373
$\mathbf{P_{dynmic}}$	**29.697**	**12.379**	**10.923**	**9.186**	**31.519**
Static Power [W]					
PL Static	2.364	0.499	0.469	0.437	1.510
PS Static	0.068	0.018	0.017	0.016	0.046
$\mathbf{P_{static}}$	**2.432**	**0.517**	**0.486**	**0.453**	**1.556**
Total Power [W]					
$\mathbf{P_{total}}$	**32.13**	**12.896**	**11.412**	**9.639**	**33.075**

Table 3.12: Mel-Approx, log-based Mel, Bark Scale, and approximated ERB Power consumption

We see that the implementations of the approximated Mel scale generic and the Traunmuller's bark scaling approaches were synthesized to the smallest amount of physical hardware resources. This also supports the measured power consumptions we see in Table 3.12, where these scaling implementations proved to be the most efficient.

Based on these results, we conducted the research using these most prominent implementations in the feature extraction modules and the ASR engine's internal acoustic scaling.

3.6 References

[1] L.L. Beranek and Acoustical Society of America. *Acoustical Measurements*. Acoustical Society of America, 1988. ISBN: 9780883185902. URL: https://books.google.co.il/books?id=c84qMQAACAAJ (cit. on p. 29).

[2] D. O'Shaughnessy. *Speech Communication: Human and Machine*. Addison-Wesley series in electrical engineering. Addison-Wesley Publishing Company, 1987. ISBN: 9780201165203. URL: https://books.google.co.il/books?id=mHFQAAAAMAAJ (cit. on p. 29).

[3] M. R. Schroeder, B. S. Atal, and J. L. Hall. "Optimizing digital speech coders by exploiting masking properties of the human ear". In: *The Journal of the Acoustical Society of America* 66.6 (1979), pp. 1647–1652. DOI: 10.1121/1.383662 (cit. on p. 40).

[4] S. S. Stevens and J. Volkmann. "The Relation of Pitch to Frequency: A Revised Scale". In: *The American Journal of Psychology* 53.3 (1940), pp. 329–353. ISSN: 00029556. URL: http://www.jstor.org/stable/1417526 (cit. on p. 29).

[5] Hartmut Traunmuller. "Analytical expressions for the tonotopic sensory scale". In: *The Journal of the Acoustical Society of America* 88.1 (1990), pp. 97–100. DOI: 10.1121/1.399849 (cit. on pp. 37, 38).

[6] S. Umesh, Leon Cohen, and Douglas Nelson. "Fitting the Mel scale". In: vol. 1. 1999, 217–220 vol.1. ISBN: 0-7803-5041-3. DOI: 10.1109/ICASSP.1999.758101 (cit. on pp. 29, 30).

[7] Mike Wereski. "The Threshold of Hearing." In: *The STEAM Journal* 2 (2015), Article 20 (cit. on p. 28).

[8] Terry L. Wiley et al. "Changes in hearing thresholds over 10 years in older adults." In: *Journal of the American Academy of Audiology* 19 4 (2008), 281–92, quiz 371 (cit. on p. 28).

[9] E. Zwicker. "Subdivision of the Audible Frequency Range into Critical Bands (Frequenzgruppen)". In: *The Journal of the Acoustical Society of America* 33.2 (1961), pp. 248–248. DOI: 10.1121/1.1908630 (cit. on p. 37).

Chapter 4

Features

4.1 Introduction

In data analysis craftsmanship, it is important to characterize the data so that variations are noticeable and more easily discernible during the analysis process. To that end, the input data being analyzed is reorganized according to selected features on which conclusions and distinctive deductions can be made. Similarly, in a supervised learning procedure, data reorganization is needed in order to make classification decisions accurately.

An intelligent choice of the learnable features [5] can drastically change a given model's outcomes quality. In supervised learning, the feature selection and extraction process, are preparatory steps to the learning or classification stages coming next. As a rule of thumb, the more features, the better the accuracy a learning model can yield theoretically[3]. That holds true to a great degree as long as the amount of sudden fluctuations do not characterize the data. In a case of heavy fluctuating data, or alternatively, in the case of a massive number of features, that some of which have minor contributions

to the classification part of the output, an increase in the number of features may become deteriorative. In terms of performance, enlarging the number of features leads to a bigger model, an increased number of learnable parameters, longer training times, and unnecessary extension of processing times. Also, a possible reduction in the accuracy is expected due to False-Negative (FN) or False-Positive (FP) misdetections resulting from the wrong classification of signals as noise and vice versa caused by additional redundant features.

For speech signals, a wide variety of meaningful feature sets exist. More or less useful, different speech features may better fit certain use-cases or fulfill a particular unerring task. Features for speech (including audio) are mainly from the following domains: spectral features, cepstral features, time domain features, and spatial features.

Speech features are selected to give the maximal accuracy in detecting utterances. That means a precise characterization of a word, utterance, or the pronunciation of a single character, making them distinguishable from other input streams.

4.2 Spectral Features

4.2.1 FB – FilterBanks

FilterBanks is a very common technique for spectral mapping of speech signals. By dividing the audio spectrum into multiple frequency bins with a pre-defined percentage of overlap, the spectrum is "framed" according to frequency. Thus, by a set of bandpass filters, each of which contain the confined information of the speech signal that corresponds to the filter's specific range

of frequencies.

The resolution can then be set as a function of the number of filters and the overall processed audio bandwidth. Increasing the number of filters, assuming the audio bandwidth and the overlap ratio are constant, means narrower allocated bandwidths for each individual filter or in other better spectral resolution.

Filterbanks by themselves are not the desired speech feature, but only the mean for feature extraction. The most common feature extracted by a filterbank set is the total sum of energy bounded by the filter's frequency response.

A set of filters is computed per frequency bin and remains the same for the entire signal length over time. Thus, the total sum of energy computed for each filter characterizes the speech over a finite defined duration of time.

The human hearing system is less sensitive to high frequencies than lower band frequencies, as described in Chapter 3. Therefore, in an attempt to emulate that same natural behavior and to resemble human hearing "filters" as much as possible, the filterbank set of filters is set with center frequencies according to the different scaling methods noted in Chapter 3. In that way, narrow-band filters are assigned to lower frequency ranges. Similarly, wide-band filters are for the higher hearable frequency ranges.

4.2.1.1 Mel FB

One way of mapping the audio spectrum is according to the Mel scale. This scaling method is described in Chapter 3. First, the center frequencies and

the bandwidths are received by the transformation between Hertz to Mels. Then a set of Bartlett filters with those center frequencies are generated with an overlap of 50% between adjacent filters.

The amplitude of the filters is bounded to "1" to maintain Nonzero Overlap Add (NOLA) compatibility. An example of a Mel Filterbank constructed by Bartlett filters is shown in Figure 4.1.

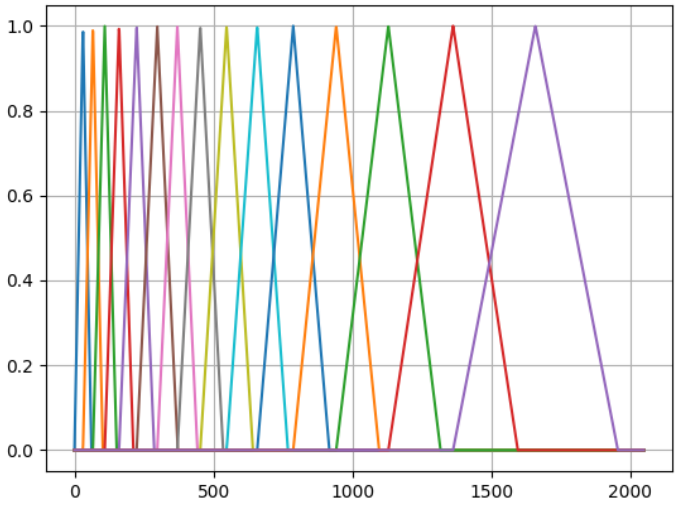


Figure 4.1: Mel FB

4.2.1.2 Bark FB

The Bark Filterbank is very similar to the Mel Filterbank except that it follows the Bark scale instead of the Mel Scale.

Different filter shapes are available but, triangular filter shapes are the most common, as described in [8].

An example of the Bark Filterbank constructed by Bartlett filters is shown

in Figure 4.2.

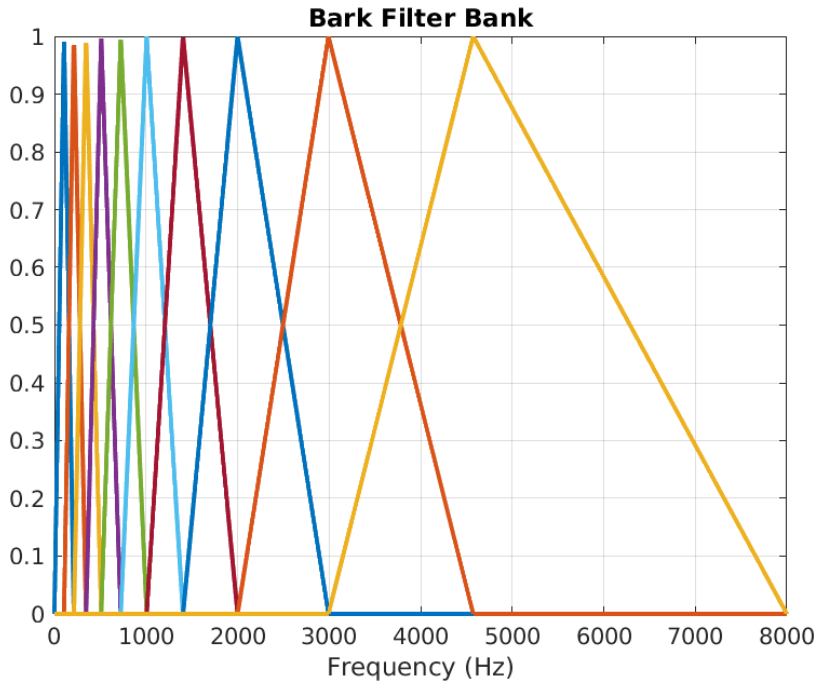


Figure 4.2: Bark FB

4.2.1.3 Gammatone FB

The Gammatone filter, as described in [1], has a response function as follows:

$$g(t) = \alpha t^{n-1} e^{-2\pi bt} \cos\left(2\pi f_c t + \phi\right) \tag{4.1}$$

Where α denotes the amplitude factor, n is the filter order, f_c is the center frequency, ϕ is the phase factor, and b is the bandwidth parameter computed according to the ERB scale mapping of $1.019 \cdot ERB\left(f_c\right)$.

An example of a Gammatone Filterbank is shown in Figure 4.3.

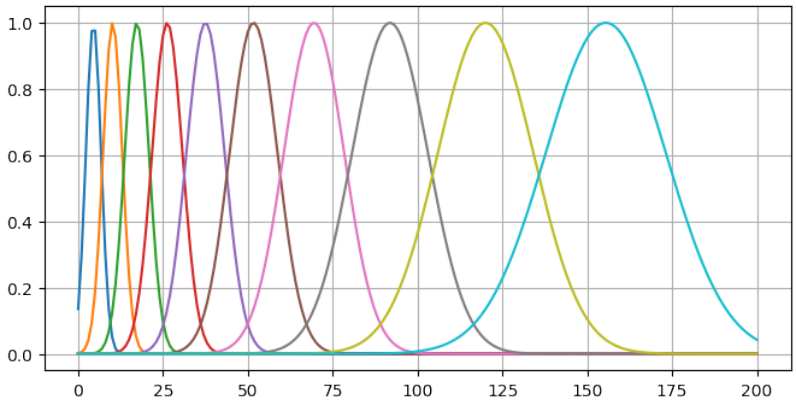


Figure 4.3: Gammatone FB

4.3 Cepstral Features

4.3.1 MFCCs – Mel-Frequency Cepstral Coefficients

4.3.1.1 Pre-emphasis

Speech signals have a roll-off frequency resembling a low-pass behavior [6]. Due to that physical nature, higher frequencies decay faster than lower speech frequencies. Compensation for this phenomenon is attainable with a pre-emphasis filter that boosts the higher [2] frequencies responses.

4.3.1.2 Framing

Speech signals vary in time. Although the variation over time can be relatively slow, a speech signal is not a pure stationary process but a quasi-stationary. Therefore, analysis of speech signals is taken on small portions of the signal to be less affected by randomness effects. To that end, a preliminary framing action is applied to speech signals in the time domain. Framing means dividing the signal into small fragments (frames) with some overlapping in between.

Each time frame is assumed to be stationary, and thus a measurement can be taken.

The shifting in time between frames is referred to as the hopping length and is set to contain a sufficient amount of temporal context that characterizes the natural characteristics of speech.

A very common sampling frequency of audio signals is $16KHz$. In order to work with a round number of sampling points, the frame lengths and the hopping lengths are set to $25ms$, and $6.25ms$ or $10ms$ for hopping size. These values translate to a frame length equals 400 sampling points at $16KHz$, and hopping size equals 100 or 160 sampling points, respectively. Another advantage of setting the hopping length as $6.25ms$ is that it gives a complete temporal context of 3 adjacent frames by definition.

4.3.1.3 Windowing

In order to extract each frame only whilst also minimizing the Gibbs phenomenon as much as possible, a windowing function is applied. As a result, the information confined in a given frame is extracted while the window's response function tapers the edges to reduce the adjacent frames' effect.

Usually, a Hamming or a Hann window is used as the window function due to their relatively decent trade-off between edge tapering, implementation simplicity, bandwidth, and spectral leakage, making them exceptionally suitable for speech signals.

The Hamming and Hann windows are given by Equations 4.2 and 4.3,

respectively.

$$W_{Hamming}[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \leq n \leq M-1 \tag{4.2}$$

$$W_{Hann}[n] = 0.5 - 0.5\cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \leq n \leq M-1 \tag{4.3}$$

4.3.1.4 DFT spectrum

Each one of the windowed frames is converted to the frequency domain by applying the Discrete-Time Fourier Transform (DTFT). Later the DTFT results are discretized giving the Discrete Fourier Transform (DFT). The composition of both windowed framing and the DFT is referred to as the Short-Time Fourier Transform (STFT). A technique to visualize the STFT outcome is called a spectrogram. The STFT outcome contains multiple frequency bins per time frame, making it a function of f and t.

The DTFT is given by:

$$X_m(f) = \sum_{n=-\infty}^{\infty} x[n]g[n - mR]e^{-j2\pi fn} \tag{4.4}$$

Where $X_m(f)$ denotes the DFT transformation of a given time frame windows signal, $g[\circ]$ denotes the window function of size M, and R represents the hopping size.

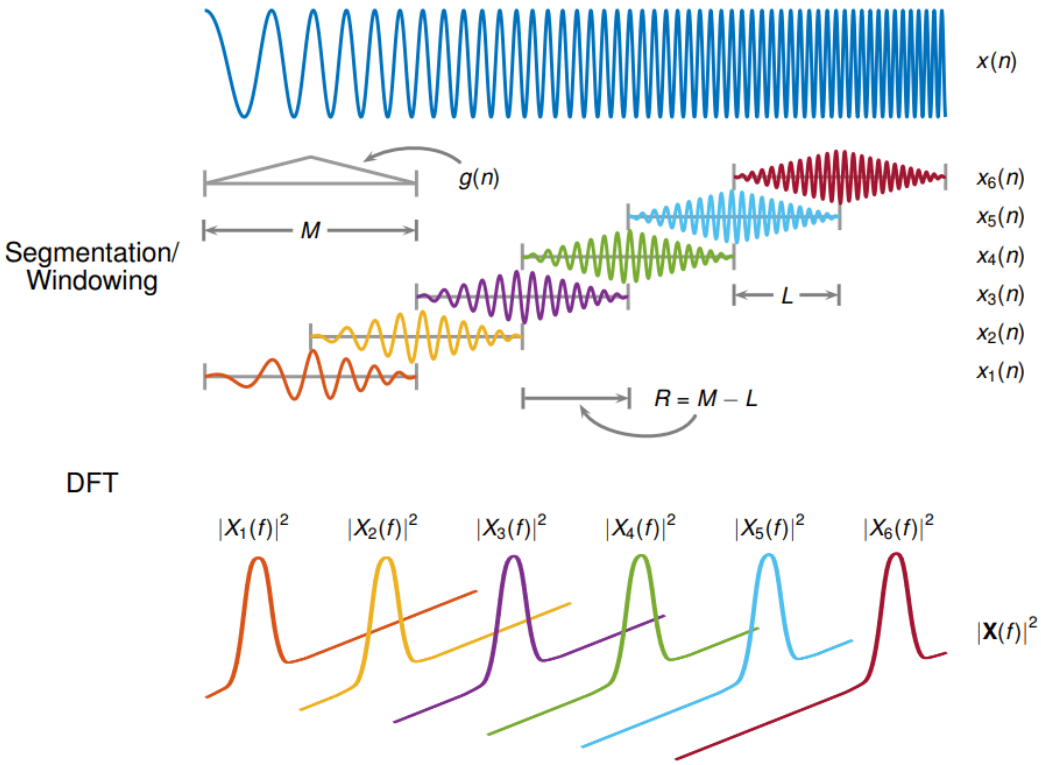


Figure 4.4: STFT demonstration diagram

Source: Adapted from Matlab's STFT documentation [4]

4.3.1.5 Mel-spectrum

Once the spectrogram is received, the frequencies are rescaled in accordance with the Mel Scale. A comprehensive description of the Mel Scale was detailed in Chapter 3.

Next, the cepstral coefficients are extracted from each frequency bin, by taking the energies summed of each filter and multiplying it with the Discrete

Cosine Transform (DCT).

$$MFCC[n] = \sqrt{\frac{2}{K}} \sum_{k=1}^{K} \{e[k] \cdot DCT\,(k)\}$$

$$= \sqrt{\frac{2}{K}} \sum_{k=1}^{K} \left\{ e[k] \cdot \cos\left(\frac{\pi n}{K}(k+0.5) \right) \right\} \tag{4.5}$$

It is very common to extract the log-Mel energies and then have Equation 4.5 written as:

$$MFCC[n] = \sqrt{\frac{2}{K}} \sum_{k=1}^{K} \left\{ \log\left(e[k]\right) \cdot \cos\left(\frac{\pi n}{K}(k+0.5) \right) \right\} \tag{4.6}$$

4.3.2 RFCCs – Root-Frequency Cepstral Coefficients

An alternative to the log-Mel extraction of the cepstral coefficients has been suggested in [7] and [9]. The motivation to put this proposed technique under test is that the root function can be less computationally demanding than the traditional log function..

$$RFCC[n] = \sqrt{\frac{2}{N}} \sum_{k=1}^{K} \left\{ \left(e[k]\right)^{\gamma} \cdot \cos\left(\frac{\pi n}{K}(k+0.5) \right) \right\} \tag{4.7}$$

4.3.3 GFCCs – Gammatone-Frequency Cepstral Coefficients

GFCCs follow the same basic steps as the MFCCs extraction. But, instead of translating the frequencies to Mels, the spectrum is translated to the ERB scale. ERB scale implies a Gammatone Filterbank as described in Chapter 3.

4.3.4 BFCCs – Bark-Frequency Cepstral Coefficients

Like the Gammatone-FCCs (GFCCs), the BFCCs follow the Bark scale.

4.4 Time-Domain Features

4.4.1 Dynamic FCC features

During the framing process, the speech signal is divided into small fragments of the speech over time. These time unit fragments are called frames. Each frame spans over a finite time duration. The extracted cepstral coefficients are computed statically for a given frame. However, the original speech is framed in multiple number of frames. Therefore, coefficients extractions have to be dynamic for the entire signal, i.e., all frames.

Additional information about the temporal changes between adjacent coefficients can also be extracted to include the dynamics and transitions within a frame.

4.4.1.1 Deltas

The first derivate of the cepstral coefficients, known as the Deltas (Δ), represents the velocity of the MFCCs' dynamics and is a sub-set of the cepstral coefficients.

Deltas (Δ) are extracted by:

$$\Delta[n] = \frac{\sum_{i=-T}^{T} k_i c_m[n+i]}{\sum_{i=-T}^{T} |i|} \tag{4.8}$$

Where n denotes the frame time index, k marks the coefficient weight, T stands for the number of temporally adjacent frames used for the calculation, and c_m denotes the mth coefficient in the given frame n.

4.4.1.2 Delta-Deltas

Delta-Deltas ($\Delta\Delta$) is an extra layer of information representing the acceleration at which the MFCCs' dynamics change within a given time frame. The extraction of the Delta-Deltas follows the same principale as the first derivatives. Yet, instead of taking the cepstral coefficients as the input feature, we replace c_m in Equation 4.8 with the first-order derivatives $\Delta[n]$.

4.4.2 Temporal Context

Temporal context is an attachment of raw unprocessed feature data from adjacent frames together with the currently selected feature. Whether spectral, cepstral, or spatial features, the concatenation of past and future features can infer decisions based on a memorized characteristics. Furthermore, different languages introduce contextual constraints such as relative positions of adjectives or nouns to verbs, plurals, affiliations, possessions, and other lingual principles.

Usually, the number of adjacent past and future frames to concatenate is based on the framing and DFT parameters. It is plain to understand one would not want to exaggerate and excessively use redundant frames that do not have any significant impact on the accuracy of detection. The downside of overly using temporal context frames is potentially having orders of magnitude

larger amounts of information to process.

4.5 References

[1] Ad Aertsen and P.I.M. Johannesma. "Spectro-temporal receptive fields of auditory neurons in the grassfrog - I. Characterization of tonal and natural stimuli". In: *Biological Cybernetics* 38 (1980), pp. 223–234. DOI: 10.1007/BF00337015 (cit. on p. 54).

[2] Amol Chaudhari, Amol Rahulkar, and S. B. Dhonde. "Combining dynamic features with MFCC for text-independent speaker identification". In: *2015 International Conference on Information Processing (ICIP)*. 2015, pp. 160–164. DOI: 10.1109/INFOP.2015.7489370 (cit. on p. 55).

[3] Stefanos Georganos et al. "Less is more: optimizing classification performance through feature selection in a very-high-resolution remote sensing object-based urban application". In: vol. 55. 2017. DOI: 10.1080/15481603.2017.1408892 (cit. on p. 50).

[4] Mathworks. *Matlab's STFT documentation webpage*. 2022. URL: https://www.mathworks.com/help/signal/ref/stft.html (cit. on p. 58).

[5] Shibli Nisar and Muhammad Tariq. "Intelligent feature selection using hybrid based feature selection method". In: *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*. 2016, pp. 168–172. DOI: 10.1109/INTECH.2016.7845025 (cit. on p. 50).

[6] J.W. Picone. "Signal modeling techniques in speech recognition". In: *Proceedings of the IEEE* 81.9 (1993), pp. 1215–1247. DOI: 10.1109/5.237532 (cit. on p. 55).

[7] Ruhi Sarikaya and John Hansen. "Analysis of the root-cepstrum for acoustic modeling and fast decoding in speech recognition." In: 2001, pp. 687–690 (cit. on p. 59).

[8] Hartmut Traunmuller. "Analytical expressions for the tonotopic sensory scale". In: *The Journal of the Acoustical Society of America* 88.1 (1990), pp. 97–100. DOI: 10.1121/1.399849 (cit. on p. 53).

[9] V. Tyagi and C. Wellekens. "On desensitizing the Mel-cepstrum to spurious spectral components for robust speech recognition". In: *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* Vol. 1. 2005, I/529–I/532 Vol. 1. DOI: 10.1109/ICASSP.2005.1415167 (cit. on p. 59).

Chapter 5

Time-Frequency Masking

5.1 Introduction

Speech separation makes use of various algorithms in order to distinguish in one way or another between desired speech and interferences. It is similar to other prevalent enhancement applications, such as the Computational Auditory Scene Analysis (CASA) [1] and Blind Speech Separation (BSS) [3]. The advantages of precise distinction between speech and interference include, among others, the ability to apply speech enhancements, noise cancelation or reduction, speech corrections, and more. With ASR systems, these abilities are expressed in improved performance of precisely detecting words at higher rates.

A typical algorithm in CASA applications is time-frequency masking (T-F masking). Since speech signals vary with time, as described in Chapter 4, they are not considered stationary. Hence, a unique representation is required for conducting an accurate analysis of such signals.

A T-F representation means presenting a speech signal in time-frequency

composition, where each T-F unit contains the speech's spectral elements at a certain time window bin. As described in Chapter 4, the T-F presentation of a signal is produced by using the STFT or auditory filtering [5].

Examples of STFT outputs for a noisy speech mixture and a clean reference of the same speech are given in Figures 5.1 and 5.2 respectively.

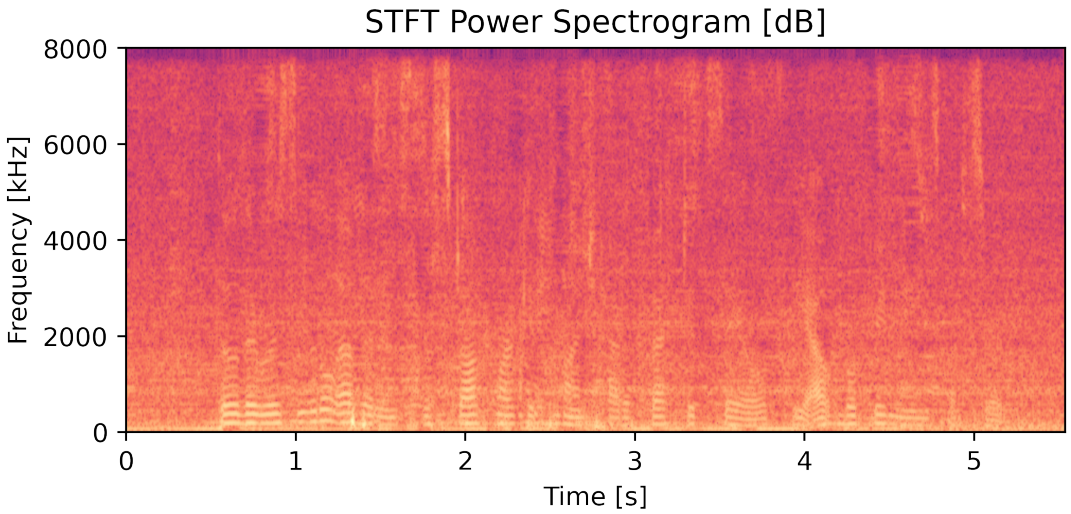


Figure 5.1: Noisy mixture spectrogram

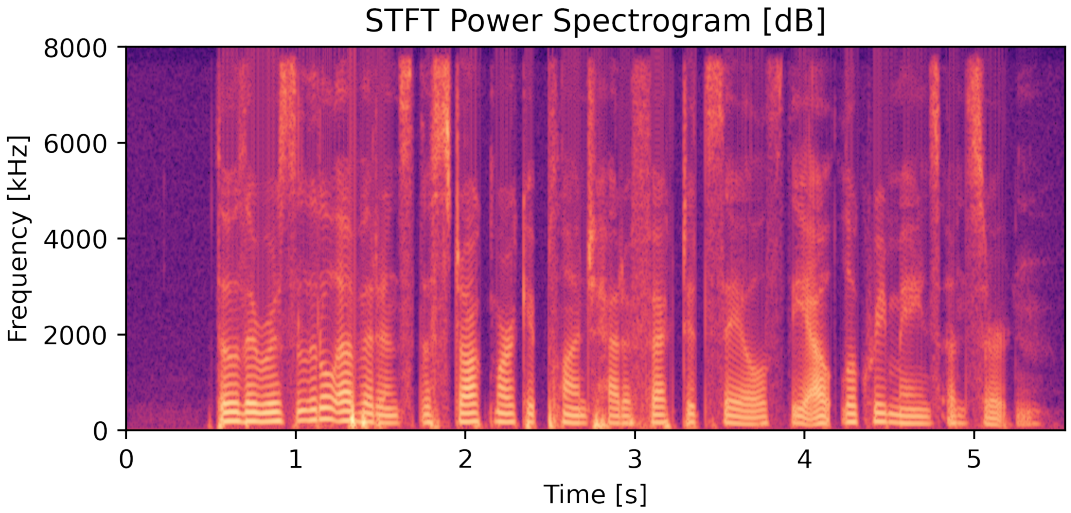


Figure 5.2: Reference clean speech spectrogram

Examination of the T-F units provides a possibility to classify an entire unit or parts of a unit as a speech-centric bin or, on the contrary, as interference (mainly noise). Based on this classification, appropriate weights are associated with each segment bin. In that way, speech elements can be extracted by separating the classified speech out of the mixture, or alternatively, attenuation the non-speech elements in certain activity areas along the length of the audio signal.

In general, the association of weights for classifying different elements in a signal is called T-F masking. This process of T-F masking takes a significant role in CASA and BSS applications, and in recent years, they have proved to be very useful in E2E-ASR systems.

An additional use of T-F masking is to tie it as input to a beamformer that in turn, filters interferences based on the classification received by the masking operation.

5.2 IBM – Ideal Binary Mask

Assuming that a voice activity detection algorithm can separate speech from noise effectively, the IBM technique is based on power differences. Whenever the speech's power spectral density (PSD) is higher than any of the interferences PSD, the speech is masked binarily to "1".

$$\mathbf{M}_s(j\omega, t) = \begin{cases} 1, & if\ |\mathbf{S}(t, j\omega)|^2 - |\mathbf{N}(t, j\omega)|^2 > \epsilon \\ 0, & otherwise \end{cases} \tag{5.1}$$

Where ϵ marks a changeable threshold value for the speech activity detection over the noise.

Equation 5.1 indicates that as a result of applying the IBM masks, the speech and interference separated sources become complementary in such a way that

$$\mathbf{M}_n = 1 - \mathbf{M}_s$$

and thus:

$$y(t) = \begin{cases} \hat{x_s}, & if \;\; \mathbf{M}_s = 1 \\ \hat{x_n}, & if \;\; \mathbf{M}_s = 0 \end{cases} \tag{5.2}$$

5.3 IRM – Ideal Ratio Mask

Unlike IBM, where hard decisions in terms of boolean values are made, by marking speech or noise elements with "true" or "false", the Ideal Ratio Mask (IRM) provides a soft decision mechanism with values in the range of $[0, 1]$ [2]. The IRMs at any T-F unit of the clean speech and noise artifacts, $\mathbf{M}_s(j\omega, t)$ and $\mathbf{M}_n(j\omega, t)$, are given in Equations 5.3 and 5.4, respectively.

$$\mathbf{M}_s(j\omega, t) = \left(\frac{|\mathbf{S}(t, j\omega)|^2}{|\mathbf{S}(t, j\omega)|^2 + |\mathbf{N}(t, j\omega)|^2} \right)^{\beta} \tag{5.3}$$

$$\mathbf{M}_n(j\omega, t) = \left(\frac{|\mathbf{N}(t, j\omega)|^2}{|\mathbf{S}(t, j\omega)|^2 + |\mathbf{N}(t, j\omega)|^2} \right)^{\beta} \tag{5.4}$$

Where β is a tunable parameter that controls the strength of the mask estimations. A value of 0.5 is used for a fair trade-off between speech and

noise mask estimations.

$$\mathbf{M}_s(j\omega, t) = \sqrt{\left(\frac{|\mathbf{S}(t, j\omega)|^2}{|\mathbf{S}(t, j\omega)|^2 + |\mathbf{N}(t, j\omega)|^2}\right)} \tag{5.5}$$

$$\mathbf{M}_n(j\omega, t) \approx 1 - \mathbf{M}_s \tag{5.6}$$

Example of the IRM masked outputs of a speech signal are given in Figures 5.3 and 5.4. Figure 5.3 shows the IRM speech mask $\mathbf{M}_s(j\omega, t)$ and Figure 5.4 shows the IRM noise mask $\mathbf{M}_s(j\omega, t)$.

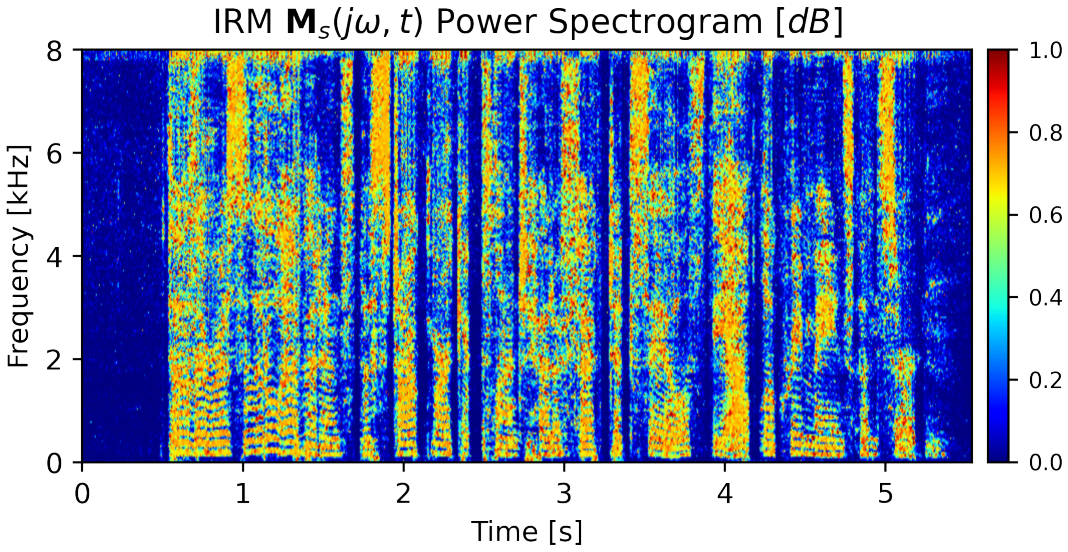


Figure 5.3: IRM Speech Mask $\mathbf{M}_s(j\omega, t)$

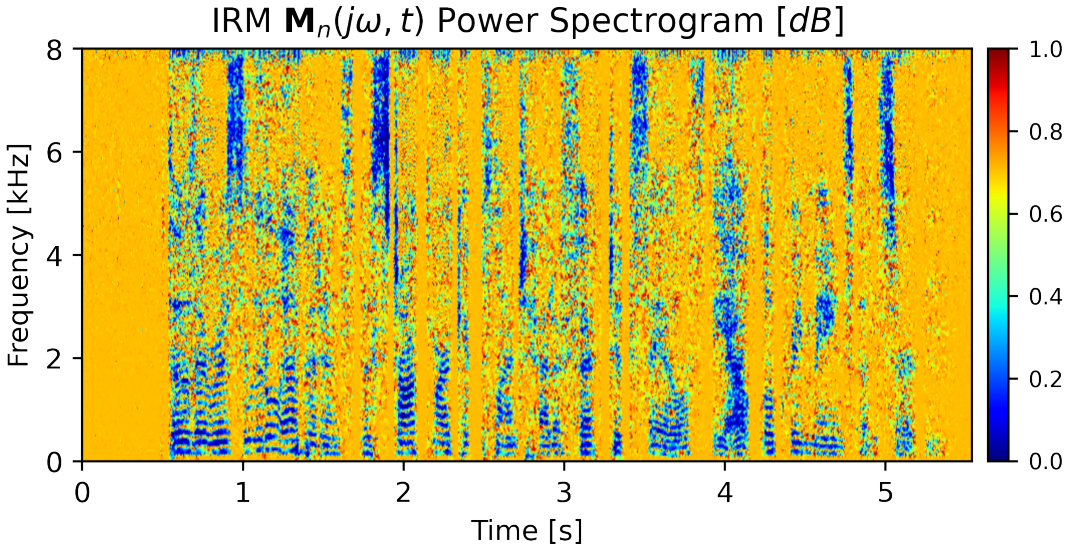


Figure 5.4: IRM Noise Mask $\mathbf{M}_n(j\omega, t)$

Thus, the covariance matrices are extracted by:

$$\mathbf{R}_{NN}(j\omega) = \frac{1}{\sum_{t=1}^{T} \mathbf{M}_{n_{(jw,t)}}} \sum_{t=1}^{T} \mathbf{M}_{n_{(jw,t)}} |\mathbf{R}_{x_{(jw,t)}}|^2 \tag{5.7}$$

$$\mathbf{R}_{SS}(j\omega) = \frac{1}{\sum_{t=1}^{T} \mathbf{M}_{s_{(jw,t)}}} \sum_{t=1}^{T} \mathbf{M}_{s_{(jw,t)}} |\mathbf{R}_{x_{(jw,t)}}|^2 \tag{5.8}$$

5.3.1 Masks Predictions

Predicting T-F masks is possible using neural network models that are trained against a training dataset of ideal masked signals. The IRM masks, $\mathbf{M}_s(j\omega, t)$, $\mathbf{M}_n(j\omega, t)$ given in Equations 5.3 and 5.4 are bounded in the range of [0, 1]. Since both the speech and noise masks are bounded in that range, it is easier to use a Sigmoid activation function as the output layer, thus ensuring the output indeed remains in this range.

A simple neural network of six to eight fully-connected (FC) layers, as seen in Figure 5.5, is used for classifying the input features to the desired IRM masks. The network's input features are the MFCCs of the microphone-array. Each input signal is framed into 25ms lasting frames with a hopping length of 6.25ms. With a sampling frequency of 16KHz, the framing settings are 400 samples per frame and a total of 300 overlapping samples.

A BLSTM layer is placed as the first block in the network, giving weight to the temporal context as well. In that way, three additional adjacent feature maps are concatenated from both sides of the currently processed T-F unit feature map. The BLSTM is then followed by several fully connected (FC) layers holding in between the Rectified Linear Unit (ReLU) activation functions.

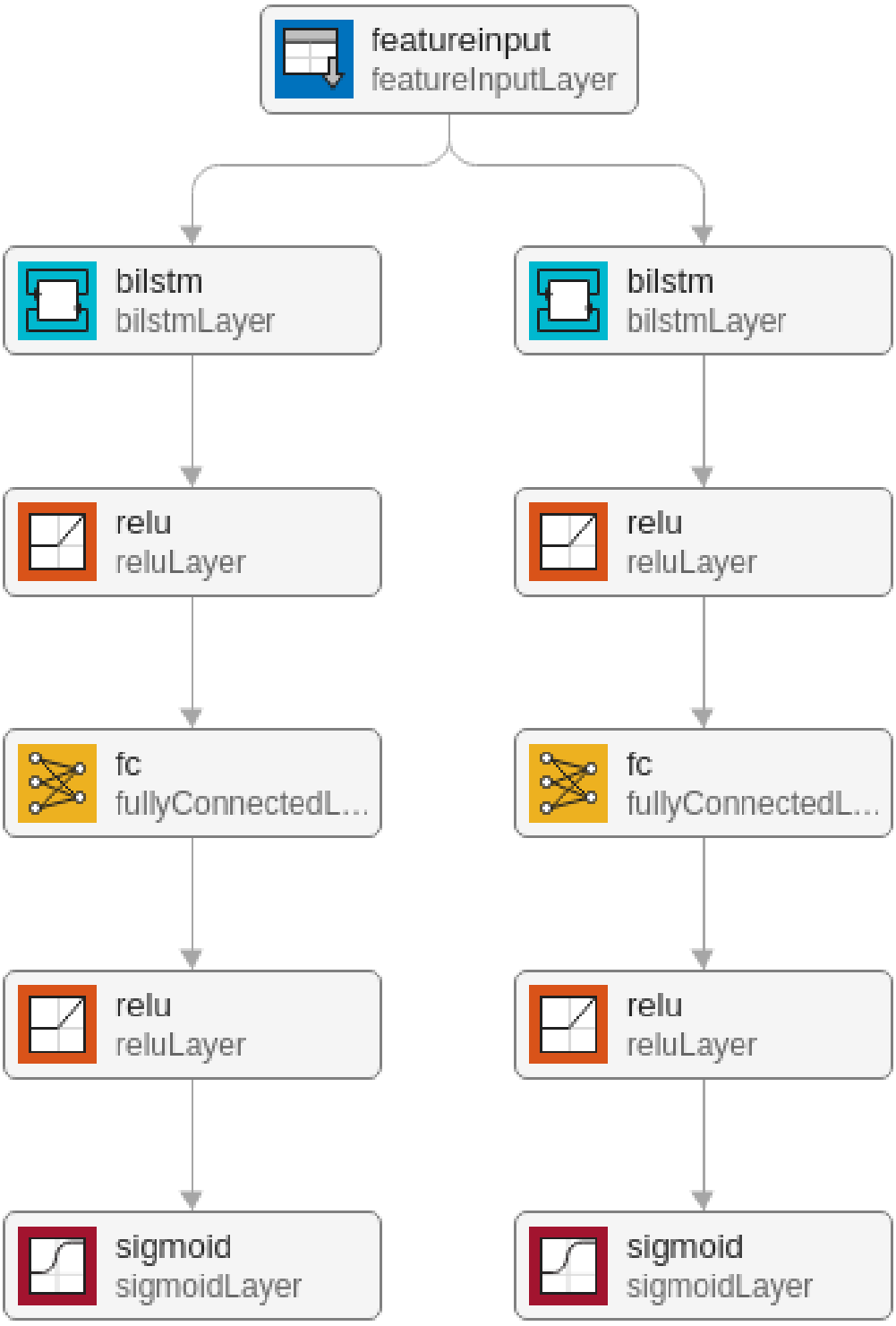


Figure 5.5: IRM estimation DNN blocks diagram

Production of both the speech and noise estimations co-occurs. Consequently, the cost function should consider the two masks in the minimization process of the error term. To that end, the cost function is defined as:

$$\ell(\widehat{\mathbf{M}}_{j\omega,t},\ \mathbf{M}_{j\omega,t}) = \frac{1}{2N}\sum_{j\omega,t}\left[\beta\left(\widehat{\mathbf{M}}^{(s)}_{j\omega,t} - \mathbf{M}^{(s)}_{j\omega,t}\right)^2 + (1-\beta)\left(\widehat{\mathbf{M}}^{(n)}_{j\omega,t} - \mathbf{M}^{(n)}_{j\omega,t}\right)^2\right] \tag{5.9}$$

Here, β denotes the weighing factor. For a fair and evenly consideration between the masks, β is set to 0.5.

The proposed IRM estimation network blocks diagram is shown in Figure 5.6.

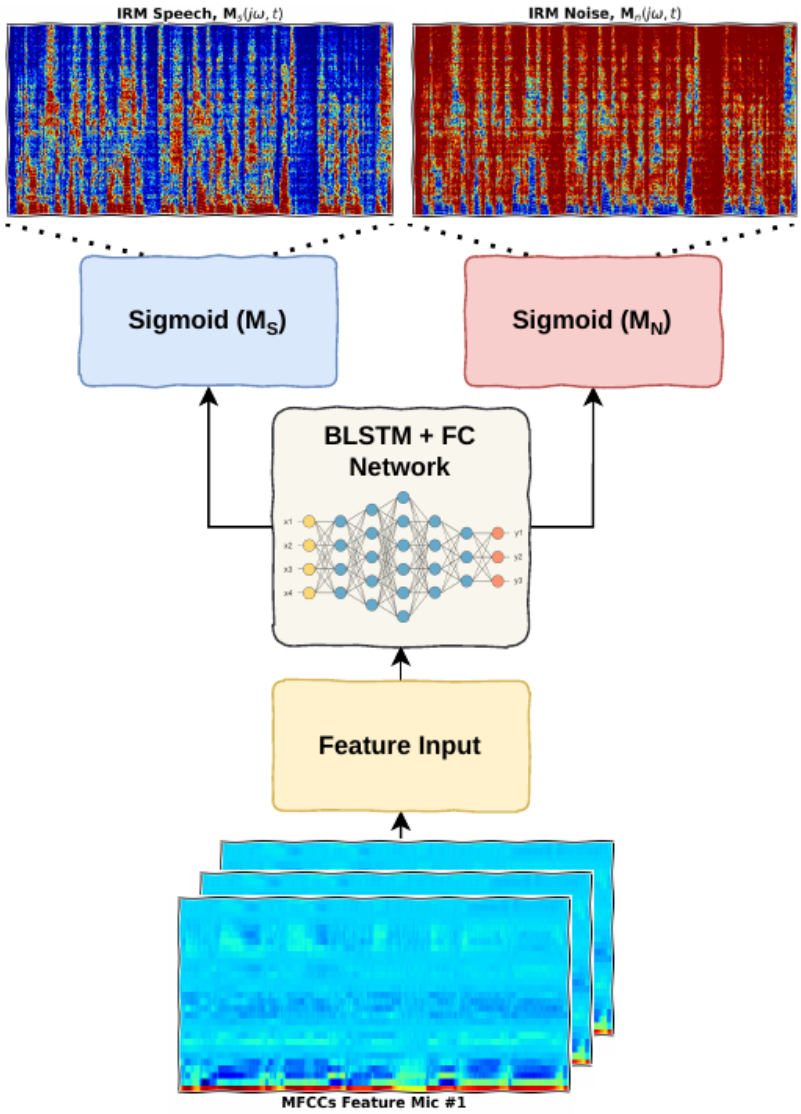


Figure 5.6: Implemented DNN for IRM T-F masking estimations

Figures 5.7a and 5.7c shows the speech and noise spectrograms results of an ideally IRM masked signal. We use these masks as a reference for comparison with the estimated speech and noise masks produced by our DNN model shown in Figures 5.7b and 5.7d.

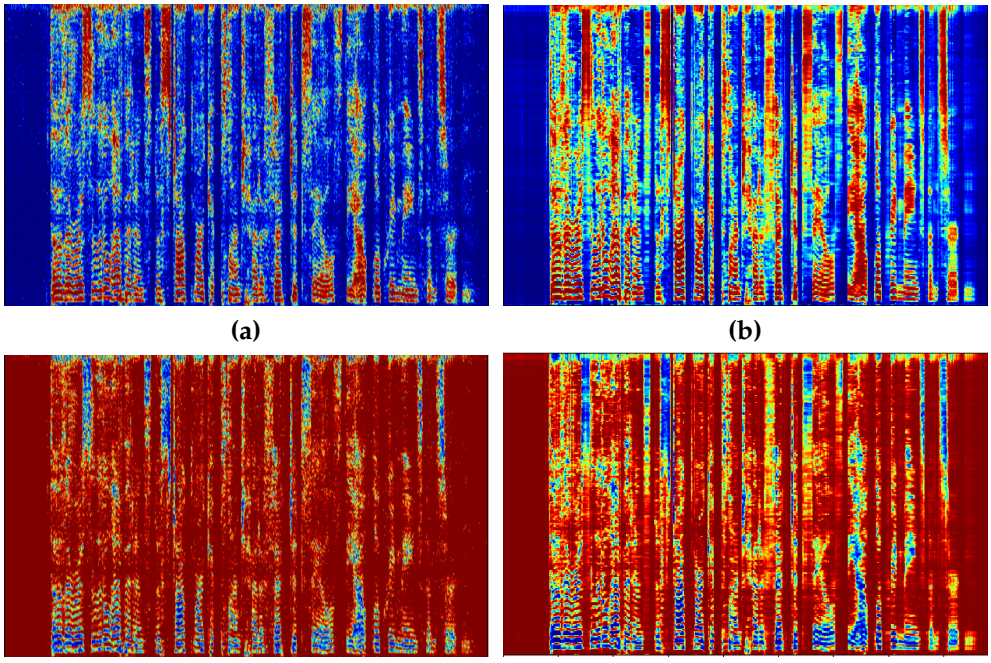


Figure 5.7: (a) and (b) are the "IRM" reference vs. estimation masks of the speech $\mathbf{M}^{(s)}_{j\omega,t}$, $\widehat{\mathbf{M}}^{(s)}_{j\omega,t}$; (c) and (d) are the "IRM" reference vs. estimation masks of the noise $\mathbf{M}^{(n)}_{j\omega,t}$, $\widehat{\mathbf{M}}^{(n)}_{j\omega,t}$.

5.3.2 Measurements

5.3.2.1 SNR

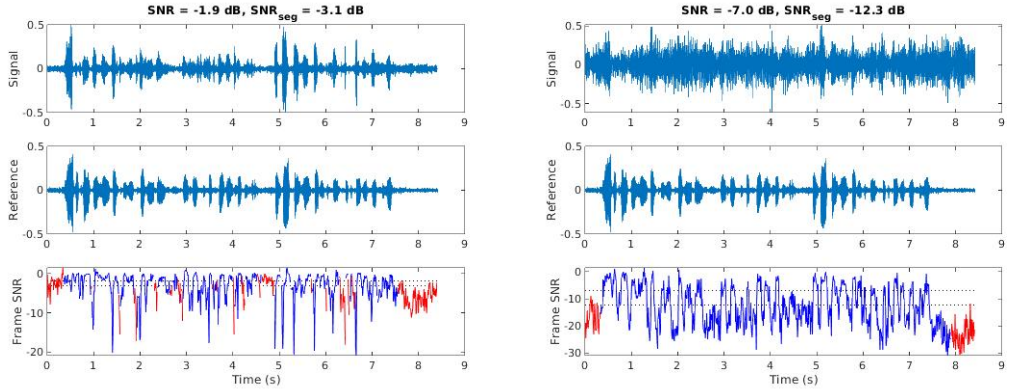


Figure 5.8: (a) "IRM" enhanced beamformed SNR & Segmental-SNR (bottom) degradation with (top) respect to the clean reference speech (middle), with the extracted SNR per frame (bottom); (b) The SNR & Segmental-SNR ratios between the noisy mixture (top) and the clean reference speech (middle), with the extracted SNR per frame (bottom).

Figure 5.8 shows two measurements of the SNR & Segmental-SNR metrics for two subject signals. We took both the clean reference and the noisy mixture of a randomly selected entry from the CHiME dataset (additional details about the CHiME dataset in Chapter 8). The noisy mixture is enhanced by a beamformer with filter coefficients resulting from the implemented IRM mask predicting DNN. Figure 5.8 shows the "IRM" enhanced beamformed version of the noisy mixture and the noisy mixture itself with respect to the clean reference speech. The overall improvements in SNR and Segmental-SNR are the ratios between the enhanced beamformed signal and the noisy mixture. Thus, from Figure 5.8 the accumulated improvement in SNR for is extracted

from the difference between the two measurements as follows:

$$|SNR_{noisy}| - |SNR_{enh}| = 5.1725\ [dB]$$

Likewise, the improvement in Segmental-SNR is 9.2675 $[dB]$. It is important to note that the enhanced beamformed signal does not align in time with the clean speech reference. In a worst-case scenario, we measured a delay of $-0.0014s$, which equals approximately 22 samples. A perfectly aligned comparison would be more accurate, but since the frame lengths' (400 samples) are much larger than the worst measured delay, the impact of this time misalignment is less acute. The Segmental-SNR calculation makes use of a VAD (voice activity detection) algorithm for dropping segments where the detected speech activity is negligible. These dropped segments are red colored in Figure 5.8.

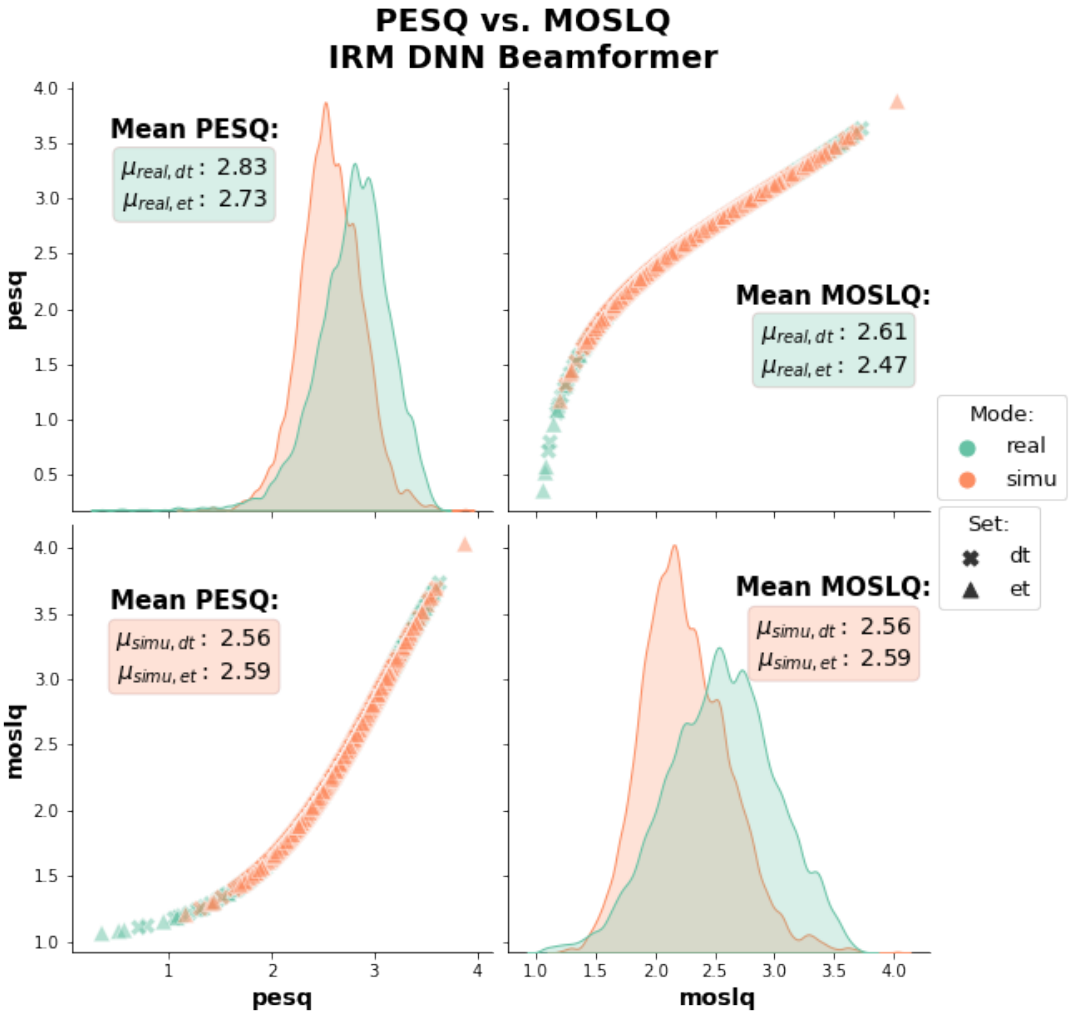


Figure 5.9: IRM beamformer PESQ vs. MOSLQ

Figure 5.9 presents the PESQ and the Mean Opinion Score Listening Quality (MOSLQ) metrics for the IRM beamformed output. The results are shown in a pair-plot diagram, where the correlations between each pair of metrics are plotted. On the diagonal, the distributions of the measured values are shown. The mean values for both the PESQ and MOSLQ under the test cases of the real and simulation datasets from the CHiME4 dataset are also shown on the plots. where "et" stands for the evaluation subset and "dt" marks

the development subset. More details about the datasets are provided in Chapter 8.

A strong correlation between the PESQ and MOSLQ is seen in the plot, and this behavior is both understandable and desirable since the MOSLQ metric is based on the PESQ measurement.

Another pair-plot for the SNR, Segmental-SNR, SI-SNR and STOI metrics is shown in Figure 5.10. The "simulation" subset yields better results in terms of performance. It is important to note that faulty microphones in the "real" subset are dropped prior to taking the measurements. Nevertheless, the ambient noise absorbed is characterized by a real scenario and a non-artificial room impulse response. On the other hand, the "simulation" subset is composed of a recording taken in a relatively clean environment (recording booth) and an artificial mixture of the recorded background noises.

Because the SNR and Segmental-SNR are measured as the improvement compared to the noisy mixture, and since the background noises are truncated to match the booth recorded speech for the "simulation" subset, the results for those metrics are more sparsely distributed compared to the "real" subset measurements. With the SI-SNR, the measurements are taken regardless of the noisy mixture. Hence, the "simulation" results are far better and more spatially dense in contrast to the "real" subset SI-SNR results.

Likewise, in the same manner, the STOI is measured regardless of the noisy mixture but relatively to the clean speech. Therefore, the "simulation" subset distribution is narrower with lower variance.

Figure 5.10: IRM beamformer SNR vs. Segmental-SNR vs. STOI vs. SI-SNR

In order to evaluate the generalization of the prediction model, the DNN estimations are compared to the ideal mask enhancement derived from directly applying the masking algorithm on the noisy mixture. Figure 5.11 shows the measurements of the SNR and Segmental-SNR for an ideal IRM mask.

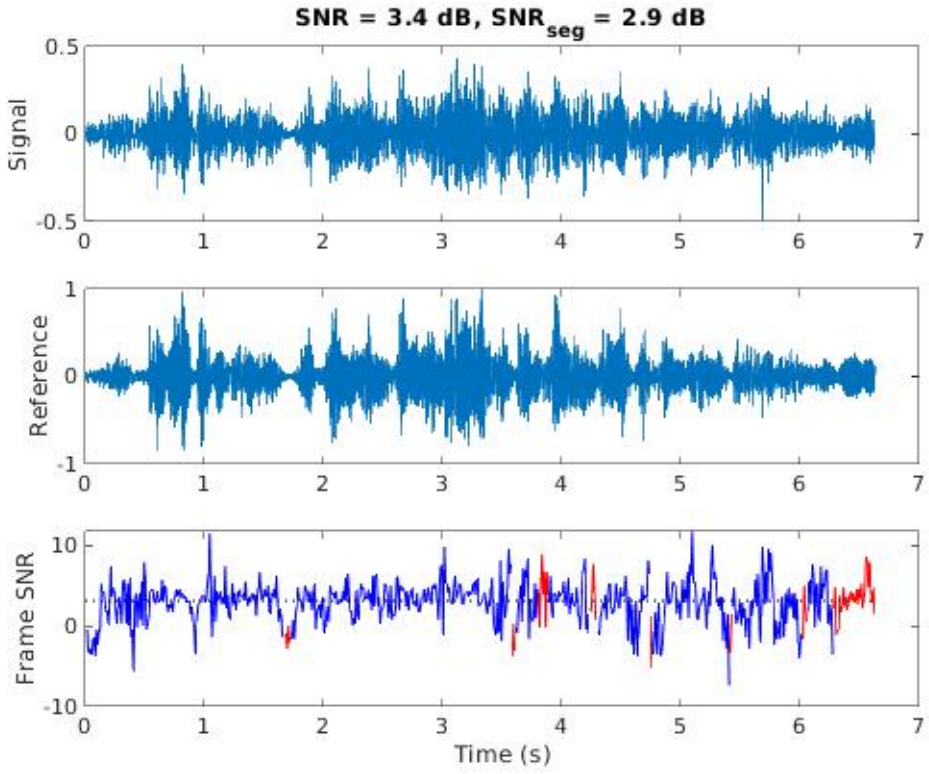


Figure 5.11: Ideal IRM beamformer enhancement SNR & Segmental-SNR.

5.4 cIRM – Complex Ideal Ratio Mask

Both IBM and IRM utilize the noise and speech magnitudes only while completely neglecting the phase arguments. The outcome of the STFT operation is a complex pair representation of each T-F unit. Recent studies have shown the importance of the phase element in speech seperation [4, 5].

In that manner, further extension of the IRM mask to include the complex pair to form the cIRM. Thus, instead of having a magnitude only based mask matrix, a complex pair of masking matrices that makes use of the phase element is given.

The mixture $y(t)$ after being processed by the STFT operation can be described as a complex sum whether in polar coordinates as seen in Equation 5.10 or in the general form as in Equation 5.11.

$$\begin{aligned}\mathbf{Y}(j\omega,t) &= \mathcal{STFT}\{y(t)\} := \mathbf{Y}_{j\omega,t} \\ &= |\mathbf{A}_\mathbf{Y}|\cos(\boldsymbol{\theta}_\mathbf{Y}) + j|\mathbf{A}_\mathbf{Y}|\sin(\boldsymbol{\theta}_\mathbf{Y}) \qquad (5.10) \\ &= \mathbf{Y}_r + j\mathbf{Y}_i \qquad (5.11)\end{aligned}$$

The real and imaginary parts can be summarized as:

$$\mathbf{Y}_r := \mathfrak{R}_\mathfrak{e}\{\mathbf{Y}_{j\omega,t}\} = |\mathbf{A}_\mathbf{Y}|\cos(\boldsymbol{\theta}_\mathbf{Y}) \qquad (5.12)$$

$$\mathbf{Y}_i := \mathfrak{T}_\mathfrak{m}\{\mathbf{Y}_{j\omega,t}\} = |\mathbf{A}_\mathbf{Y}|\sin(\boldsymbol{\theta}_\mathbf{Y}) \qquad (5.13)$$

Extraction of the magnitude and phase from the complex representation is accomplished by the following set of Equations 5.14 and 5.15:

$$|\mathbf{A}_\mathbf{Y}| = \sqrt{\mathfrak{R}_\mathfrak{e}\{\mathbf{Y}_{j\omega,t}\}^2 + \mathfrak{T}_\mathfrak{m}\{\mathbf{Y}_{j\omega,t}\}^2} \qquad (5.14)$$

$$\boldsymbol{\theta}_\mathbf{Y} = \tan^{-1}\left\{\frac{\mathfrak{T}_\mathfrak{m}\{\mathbf{Y}(j\omega,t)\}}{\mathfrak{R}_\mathfrak{e}\{\mathbf{Y}(j\omega,t)\}}\right\} \qquad (5.15)$$

Rearranging the equations above, we can write the cIRM mask's real and imaginary parts as:

$$\mathbf{M}_r = \frac{\mathbf{Y}_r\mathbf{S}_r + \mathbf{Y}_i\mathbf{S}_i}{\mathbf{Y}_r^2 + \mathbf{Y}_i^2} \qquad (5.16)$$

$$\mathbf{M}_i = \frac{\mathbf{Y}_r\mathbf{S}_i - \mathbf{Y}_i\mathbf{S}_r}{\mathbf{Y}_i^2 + \mathbf{S}_r^2} \qquad (5.17)$$

Thus, the complex masks for the speech and noise are:

$$\begin{aligned}\mathbf{M}_{j\omega,t}^{(s)} &= \mathbf{M}_r^{(s)} + j\mathbf{M}_i^{(s)} \\ &= \frac{\mathbf{Y}_r\mathbf{S}_r + \mathbf{Y}_i\mathbf{S}_i}{\mathbf{Y}_r^2 + \mathbf{Y}_i^2} + j\frac{\mathbf{Y}_r\mathbf{S}_i - \mathbf{Y}_i\mathbf{S}_r}{\mathbf{Y}_i^2 + \mathbf{S}_r^2}\end{aligned} \tag{5.18}$$

$$\begin{aligned}\mathbf{M}_{j\omega,t}^{(n)} &= \mathbf{M}_r^{(N)} + j\mathbf{M}_i^{(N)} \\ &= \frac{\mathbf{Y}_r\mathbf{N}_r + \mathbf{Y}_i\mathbf{N}_i}{\mathbf{Y}_r^2 + \mathbf{Y}_i^2} + j\frac{\mathbf{Y}_r\mathbf{N}_i - \mathbf{Y}_i\mathbf{N}_r}{\mathbf{Y}_i^2 + \mathbf{N}_r^2}\end{aligned} \tag{5.19}$$

Therefore, applying a cIRM T-F masking requires a complex multiplication for separation as opposed to the more basic IBM and IRM where magnitudes multiplications are taken in the real domain only.

The reference compressed cIRM masks are shown in Figure 5.12;

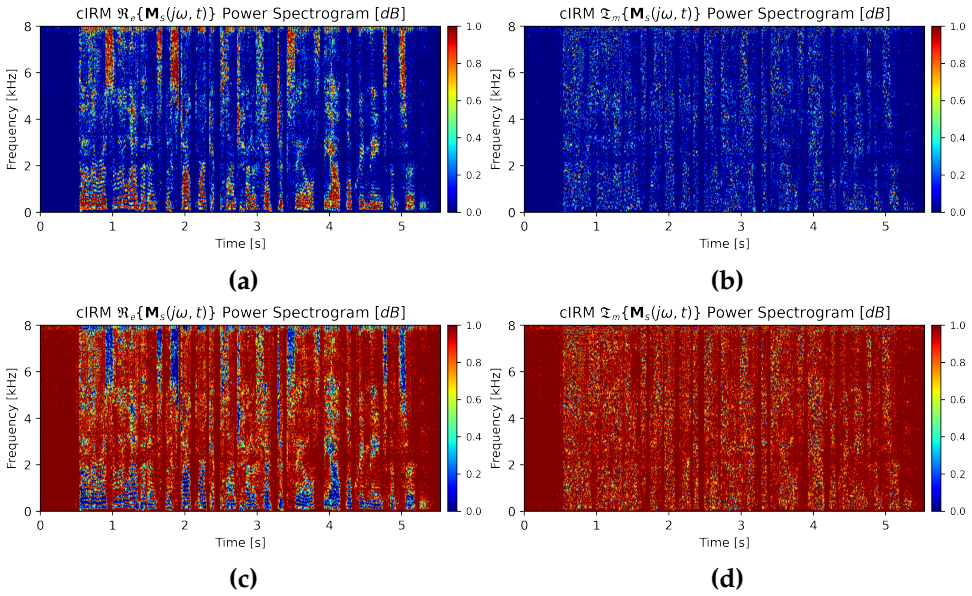


Figure 5.12: (a) and (b) are the cIRM real and imaginary references of the speech $\mathbf{M}_r^{(s)}$, $\mathbf{M}_i^{(s)}$; (c) and (d) are the cIRM real and imaginary references of the noise $\mathbf{M}_r^{(n)}$, $\mathbf{M}_i^{(n)}$.

5.4.1 Masks Estimations

In Section 5.4, the $cIRM$ masks are described in Equations 5.18 and 5.19. In contrast to the IRM masks which are bounded in the range $[0, 1]$, the $cIRM$ masks are unbounded, and have the range $(-\infty, \infty)$.

A neural network cannot train for unbounded values. Hence, an alternative presentation to the mask values is needed. One possibility is to compress the real and imaginary masks with a hyperbolic tangent as suggested in [4]:

$$cIRM_x = K\frac{1 - e^{-C \cdot M_x}}{1 + e^{-C \cdot M_x}} \tag{5.20}$$

Where x stands for the real or the imaginary parts of the mask. By applying this compression, the mask values are bounded in the range $[-K, K]$, while C controls the steepness. For that purpose, we replaced the simpler IRM DNN sigmoid layers placed at the output with linear layers.

Then, the cost function is defined to include both real and imaginary parts of both the noise and speech masks.

$$\ell(\widehat{\mathbf{M}}_{j\omega,t}^{(x)},\ \mathbf{M}_{j\omega,t}^{(x)}) = \frac{1}{2N}\sum_{j\omega,t}\left[\beta\left(\widehat{\mathbf{M}}_{j\omega,t}^{(s\in\mathbb{C})} - \mathbf{M}_{j\omega,t}^{(s\in\mathbb{C})}\right)^2 + (1-\beta)\left(\widehat{\mathbf{M}}_{j\omega,t}^{(n\in\mathbb{C})} - \mathbf{M}_{j\omega,t}^{(n\in\mathbb{C})}\right)^2\right] \tag{5.21}$$

The proposed cIRM estimation network blocks diagram is shown in Figure 5.13.

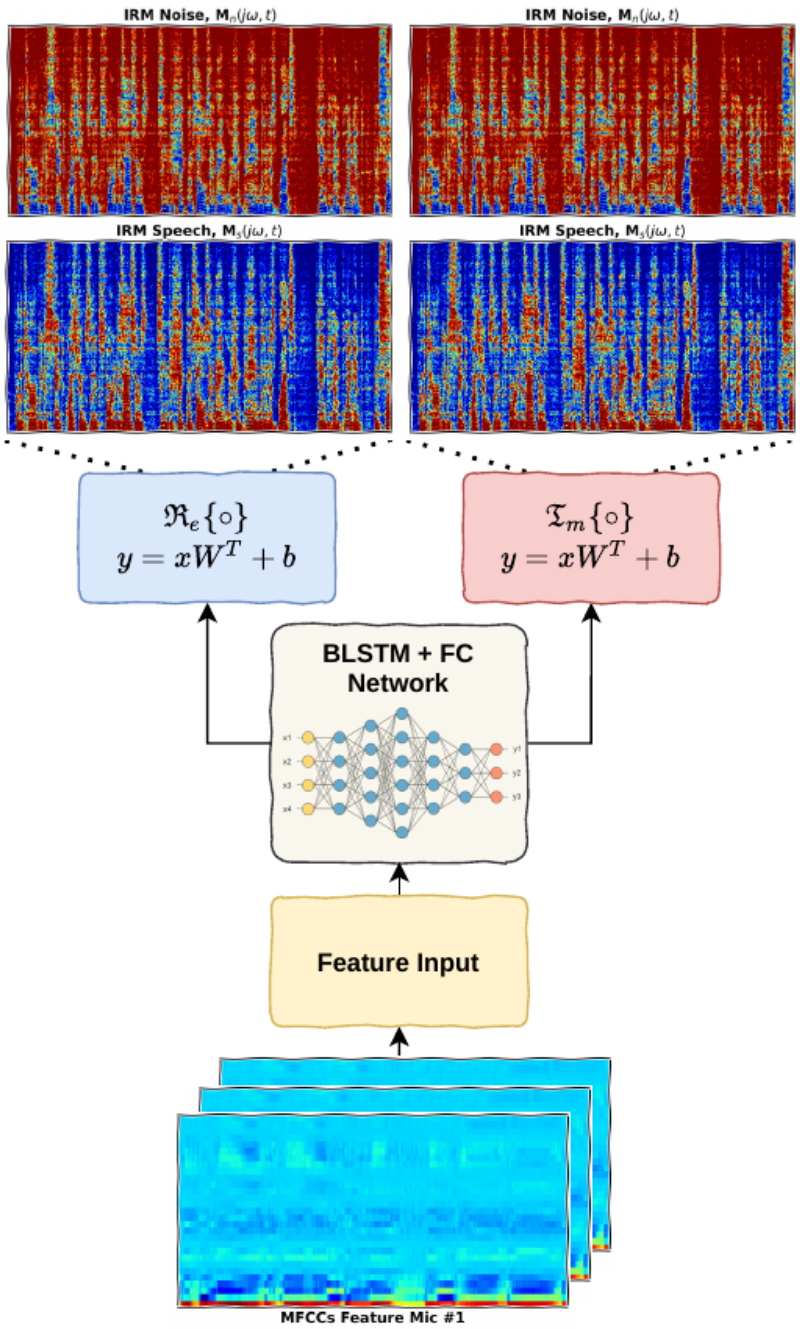
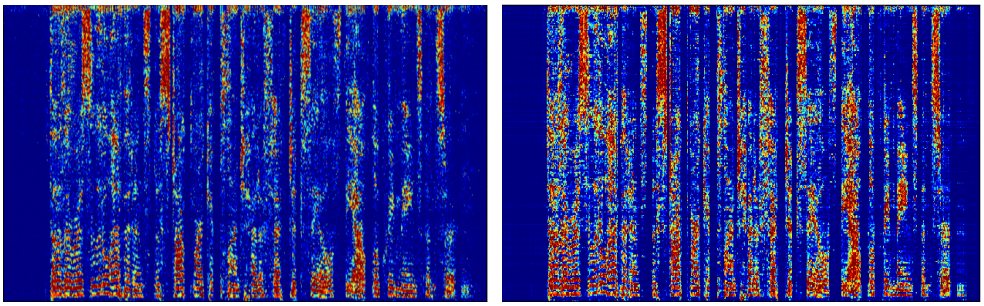


Figure 5.13: Proposed DNN for cIRM T-F masking estimations



(a) (b)

Figure 5.14: (a) Reference cIRM real speech mask $\mathbf{M}_r^{(s)}$; (b) The estimated cIRM real speech $\widehat{\mathbf{M}}_r^{(s)}$.

Figure 5.14 shows the reference real speech cIRM mask compared to the estimated real speech mask predicted by the DNN model. Carefully examining the figure, one can conclude that the model generalized correctly as the estimated mask resembles the reference to a great extent. Compared to the "IRM" DNN performance, the cIRM model with a compression in the range of $[-10, 10]$ performs better in producing mask predictions that are closer to the ideally computed masks.

5.4.2 Measurements

Due to the complexity of the cIRM DNN model, it did not generalize properly for the evaluation dataset with 50 epochs. The results hence, are not accurate as a reference nor for comparison, unless a retraining process is initialized with a sufficient number of epochs to let the model generalize.

Due to the uncertainty of the impact the low number of epochs have on the model performance, and the lack of time for a re-evaluation of the DNN mask estimation, we decide to skip the results and to drop the cIRM measurements.

5.5 PSM – Phase Sensitive Mask

The PSM T-F masking makes use of the magnitude ratios and phase differences rather than requiring a complex multiplication. In that way, the multiplication is taken in the real domain only, while utilizing the phase contribution directly.

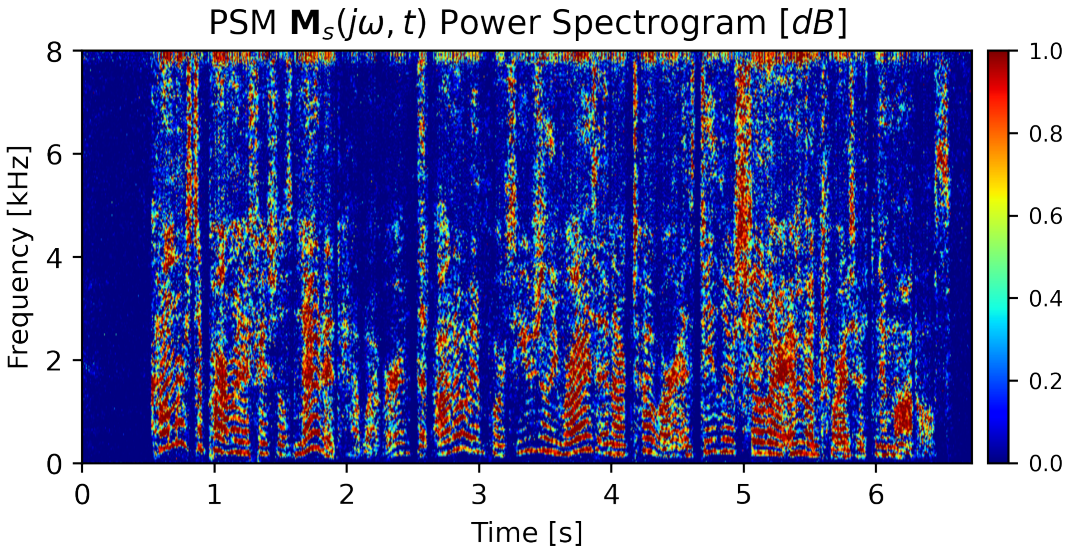


Figure 5.15: PSM Speech Mask $\mathbf{M}_s(j\omega, t)$

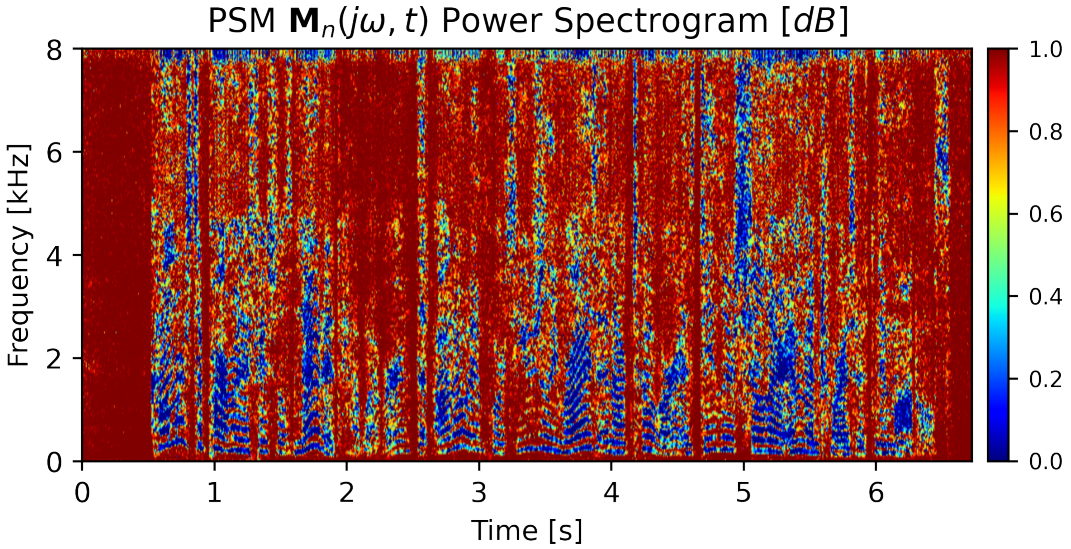


Figure 5.16: PSM Noise Mask $\mathbf{M}_n(j\omega, t)$

The following equations give the PSM masks for the reference speech and noise, $\mathbf{M}^{(s)}_{j\omega,t}$ and $\mathbf{M}^{(n)}_{j\omega,t}$:

$$\mathbf{M}^{(s)}_{j\omega,t} = \frac{|\mathbf{S}(t, j\omega)|}{|\mathbf{Y}(t, j\omega)|} \cos(\boldsymbol{\theta}_{\mathbf{S}} - \boldsymbol{\theta}_{\mathbf{Y}}) \tag{5.22}$$

$$\mathbf{M}^{(n)}_{j\omega,t} = \frac{|\mathbf{N}(t, j\omega)|}{|\mathbf{Y}(t, j\omega)|} \cos(\boldsymbol{\theta}_{\mathbf{N}} - \boldsymbol{\theta}_{\mathbf{Y}}) \tag{5.23}$$

5.5.1 Masks Estimations

Unlike the cIRM complex masks, the PSM masks are real valued. However, the masks are not bounded similarly to the cIRM masks. Therefore, the proposed model for the cIRM masks can be reused with a slight difference in implementation. Now, only two outputs are created instead of branching to four outputs at the final layer. One output for the speech mask and the other for the noise mask. The same as we did for the "IRM" but without

the Sigmoids. In addition, to deal with the unbounded range of values, we also apply a compression algorithm to the reference masks as well same as described in Equation 5.20.

Following this architecture, the cost function can be set to take two masks in the calculation of the error term, like with the "IRM" masks:

$$\ell(\widehat{\mathbf{M}}_{j\omega,t},\ \mathbf{M}_{j\omega,t}) = \frac{1}{2N}\sum_{j\omega,t}\left[\beta\left(\widehat{\mathbf{M}}^{(s)}_{j\omega,t} - \mathbf{M}^{(s)}_{j\omega,t}\right)^2 + (1-\beta)\left(\widehat{\mathbf{M}}^{(n)}_{j\omega,t} - \mathbf{M}^{(n)}_{j\omega,t}\right)^2\right] \tag{5.24}$$

The proposed PSM estimation network blocks diagram is shown in Figure 5.17.

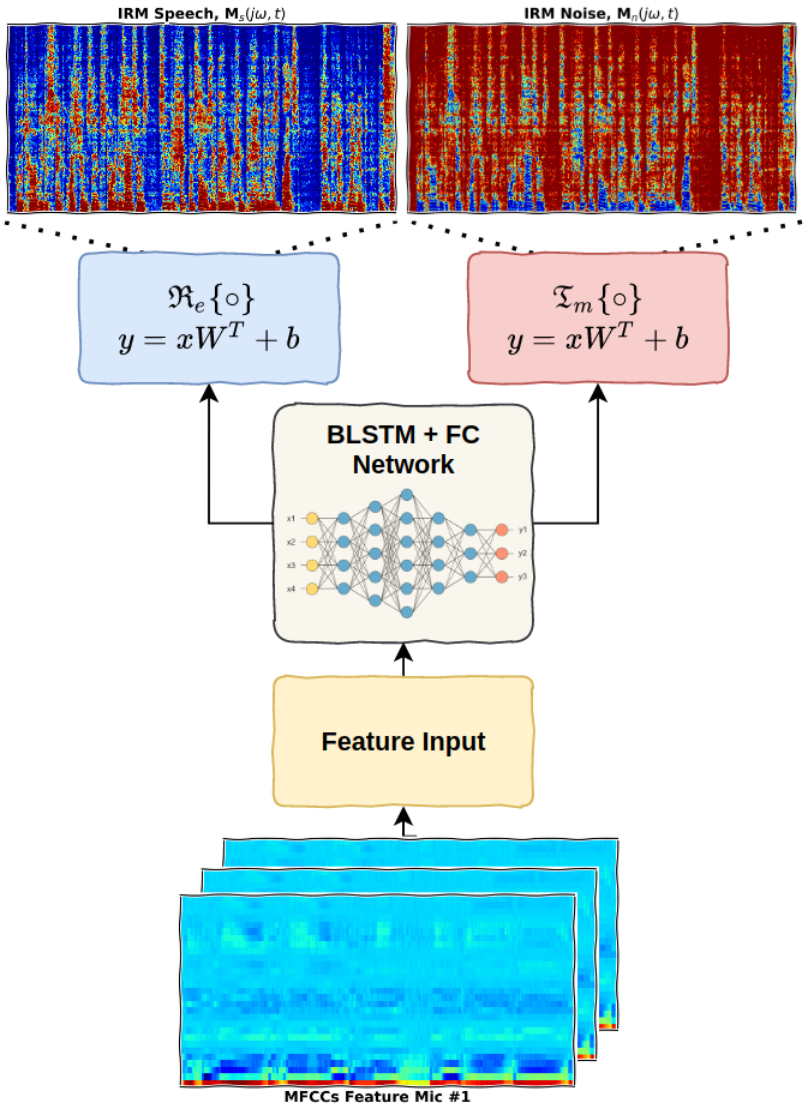


Figure 5.17: Proposed DNN for PSM T-F masking estimations.

5.5.2 Measurements

Figure 5.18 shows a pair-plot for the SNR, Segmental-SNR, SI-SNR and STOI metrics for the PSM T-F masking.

Figure 5.18: PSM beamformer SNR vs. Segmental-SNR vs. STOI vs. SI-SNR.

5.19 shows the measurements of the SNR and Segmental-SNR for an ideal ORM mask.

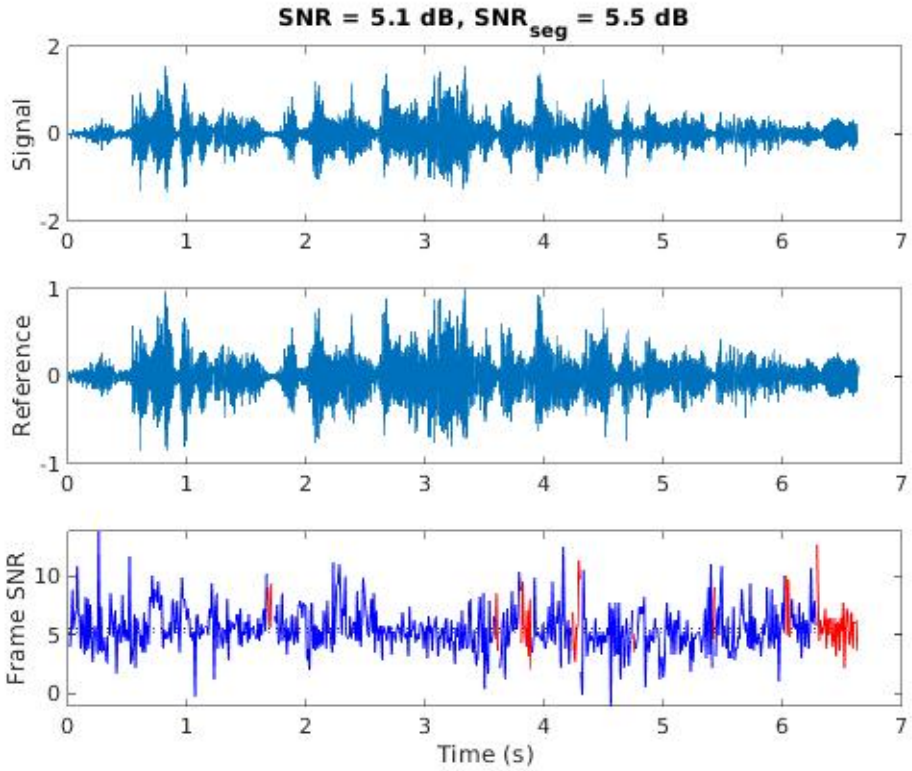


Figure 5.19: Ideal PSM beamformer enhancement SNR & Segmental-SNR.

5.6 ORM — Optimal Ratio Mask

IRM masking can be furtherly improved following the realization of the phase component importance.

Trying to minimize the general MSE loss function for T-F masking, which is same as with the "Weiner filter" that the IRM masking approximates, we

get[5]:

$$\mathbf{M}_{j\omega,t}^{(s)} = \frac{|\mathbf{S}(t,j\omega)|^2 + \Re_e\{\mathbf{S}(t,j\omega)\cdot\mathbf{N}^*(t,j\omega)\}}{|\mathbf{S}(t,j\omega)|^2 + |\mathbf{N}(t,j\omega)|^2 + 2\Re_e\{\mathbf{S}(t,j\omega)\cdot\mathbf{N}^*(t,j\omega)\}} \tag{5.25}$$

$$\mathbf{M}_{j\omega,t}^{(n)} = \frac{|\mathbf{N}(t,j\omega)|^2 + \Re_e\{\mathbf{N}(t,j\omega)\cdot\mathbf{S}^*(t,j\omega)\}}{|\mathbf{S}(t,j\omega)|^2 + |\mathbf{N}(t,j\omega)|^2 + 2\Re_e\{\mathbf{N}(t,j\omega)\cdot\mathbf{S}^*(t,j\omega)\}} \tag{5.26}$$

Looking at the equation above, it resembles the IRM form but also introduces the real part of the multiplication between the speech spectrum and the conjugate noise spectrum.

5.6.1 Masks Estimations

The ORM masking in terms of the estimation process resembles the PSM completely. Therefore, the same compression and DNN architecture are used to estimate the ORM masks.

The proposed DNN model block diagram is shown in Figure 5.17, and the cost function is given in Equation 5.24.

5.6.2 Measurements

Figure 5.20 shows a pair-plot for the SNR, Segmental-SNR, SI-SNR and STOI metrics for the ORM T-F masking.

Figure 5.20: ORM beamformer SNR vs. Segmental-SNR vs. STOI vs. SI-SNR.

5.21 shows the measurements of the SNR and Segmental-SNR for an ideal ORM mask.

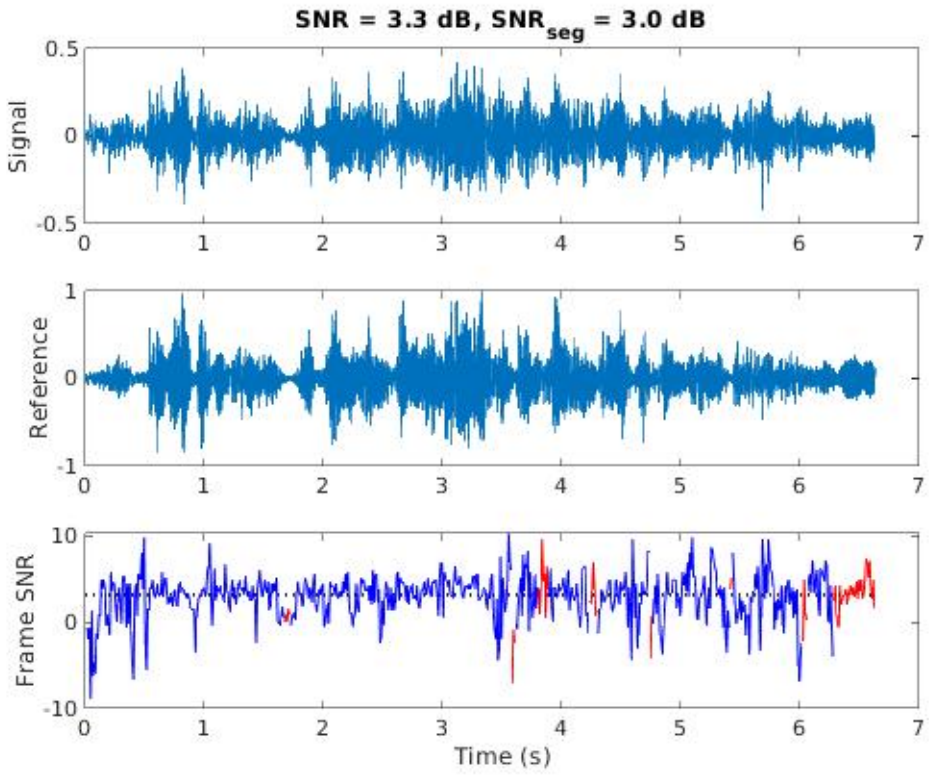


Figure 5.21: Ideal ORM beamformer enhancement SNR & Segmental-SNR.

5.7 T-F masks Conclusions

Table 5.1 presents the model's sizes in terms of memory requirement and the number of learnable parameters. The model's parameters were quantized to the form of signed 16 bits. The MSB (most significant bit) is for the sign; the following four bits are set for the integer part; and the other eleven bits present the fractional part. All of the tested models were trained with and without the $\Delta, \Delta\Delta$ features. Usage of the additional $\Delta, \Delta\Delta$ features enlarges the input size of the model and thus leads to a larger number of learnable parameters, resulting in a bigger model. Bigger models take a longer time to

train and are more complex to fit in limited resources hardware devices. A trade-off decision can be made with respect to the memory size and desired performance in the design phase of T-F masks based applications. Although presenting better results in audio metrics, the PSM and ORM masks require $\sim$ 43% larger memory space compared to the IRM masks, when the feature set does not include the $\Delta, \Delta\Delta$ features. A less severe increase of $\sim$ 22% in memory size requirement has been observed when the $\Delta, \Delta\Delta$ features were not excluded from the feature set.

Targets	Learnable Parameters [Mil]		Quant. (S16.11) Mem [Mb]	
	W/o (Δ,$\Delta\Delta$)	W/ (Δ,$\Delta\Delta$)	W/o (Δ,$\Delta\Delta$)	W/ (Δ,$\Delta\Delta$)
IRM	2.44	4.67	39.0	74.8
cIRM	3.76	5.99	60.1	95.9
PSM	3.49	5.73	55.9	91.7
ORM	3.50	5.74	56.0	91.8

Table 5.1: T-F Masks models learnable parameters vs. Required memory size

5.8 References

[1] Guy J. Brown and Martin Cooke. "Computational auditory scene analysis". In: *Computer Speech & Language* 8.4 (1994), pp. 297–336. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.1994.1016. URL: https://www.sciencedirect.com/science/article/pii/S0885230884710163 (cit. on p. 64).

[2] Wenbin Jiang, Fei Wen, and Peilin Liu. "Robust Beamforming for Speech Recognition Using DNN-Based Time-Frequency Masks Estimation". In: *IEEE Access* 6 (2018), pp. 52385–52392 (cit. on p. 67).

[3] Madhab Pal et al. "Blind source separation: A review and analysis". In: *2013 International Conference Oriental COCOSDA held jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE)*. 2013, pp. 1–5. DOI: 10.1109/ICSDA.2013.6709849 (cit. on p. 64).

[4] Donald S. Williamson, Yuxuan Wang, and DeLiang Wang. "Complex Ratio Masking for Monaural Speech Separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.3 (2016), pp. 483–492. DOI: 10.1109/TASLP.2015.2512042 (cit. on pp. 80, 83).

[5] Shasha Xia, Hao Li, and Xueliang Zhang. "Using optimal ratio mask as training target for supervised speech separation". In: *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (2017), pp. 163–166 (cit. on pp. 65, 80, 92).

Chapter 6

Beamformers

6.1 Introduction

In general, a fundamental part of signal processing involves filtering in one way or another, and the same is the case for speech signals processing. The motivation to filter incoming signals is to form a given signal so that the unwanted components, referred to as interferences, are eliminated, and the desired component of the signal remains.

A filter can be in the form of an analog or digital circuit. In the sense of a fixed-tap or static digital filter, the discrete signal samples are tapered due to a convolutional multiplication with the filter's weighted coefficients. These coefficients are preset according to desired characteristics of the desired output signal.

Since speech signals can vary in tone, pitch, formants, and frequencies, statically setting the filter's coefficients won't be sufficient and robust enough. To that end, the coefficients should be set dynamically to suit the input signal or the model to be generalized to a great end; hence, the filter's performance

would be suitable for a wide range of inputs rather than a singular case scenario. Changing the filter's coefficients dynamically in accordance with the input or the use case is called "adaptive filtering". In opposed to the case of "static filtering", where the coefficients are computed once.

In a case where multiple sources are presented such as a sensors array, a "spatial factor" is introduced such that the filtered signal is the outcome of a combination of all the receiving elements in the array that maximize the filtering effect. That "spatial filtering" in the presence of multiple sensors is called "beamforming". The guideline of beamforming is the optimization of the received signal utilizing all the sensors to maximize, in most cases, the SNR. In this research, we show that in E2E-ASR systems, the introduction of beamformers at the front-end before the ASR model manifests in higher detection results and better model accuracy.

The beamforming principle relies on acknowledging that the desired signal is absorbed and received by multiple sensors along with the unwanted interferences. In the case of beamforming speech signals, the sensors (microphones) are spread in space in a dedicated formation and set at a known distance from each other. Each sensor's input can be filtered, delayed, amplified, or undergo any form of manipulation until eventually summing up with the other sensors' inputs. In other words, the ability to manipulate each sensor's input signal in a multi-sensor array gives the ability to "form" a given signal utilizing the properties of constructive and destructive interferences.

We apply beamforming in the frequency domain:

$$Y(j\omega) = \mathbf{W}^H(j\omega)\mathbf{X}(j\omega) \tag{6.1}$$

The time difference of arrival between microphone 1 and m can be estimated using the Generalized Cross-Correlation with Phase Transform (GCC-PHAT) with the following expression:

$$\tau_m = argmax_\tau \int_{-\pi}^{+\pi} \frac{X_1(j\omega)X_m(j\omega)^*}{|X_1(j\omega)||X_m(j\omega)|} e^{j\omega\tau} d\omega \tag{6.2}$$

The power corresponds to:

$$E(\mathbf{u}) = \frac{\mathbf{A}(j\omega,\mathbf{u})^H\mathbf{A}(j\omega,\mathbf{u})}{\sqrt{\mathbf{A}(j\omega,\mathbf{u})^H\mathbf{U}(j\omega)\mathbf{U}(j\omega)^H\mathbf{A}(j\omega,\mathbf{u})}} \tag{6.3}$$

The DOA with the maximum power is selected as the DOA of sound:

$$\mathbf{u}_{max} = argmax_{\mathbf{u}} E(\mathbf{u}) \tag{6.4}$$

6.2 Multi sensors use-cases

A multi-microphone array is often named multi-sensors. Every sensor absorbs the speech differently due to the sensor's spatial location relatively to the speech source. In this work, spacing between sensors and the array's formation are pre-known and permanent. Knowing the microphone array's shape, forming a constructive interference is only a matter of the direction of arrival since all the other parameters are static.

The human speech frequencies are considered "long-length", which means that the wavelength of the frequencies in the audible band is between single

centimeters to several meters. A speaker standing in front of a microphone typically stands at a distance that does not exceed several meters, nor is it millimeters away. In either situation, the absorbed speech will be distorted or barely usable. As a result, the propagated speech waves arrive in a planar shape once they reach the microphone. In a microphone array, these characteristics become handy when applying Direction Of Arrival (DOA) and Time Difference Of Arrival (TDOA) applications.

A planar wave propagating towards a microphone array is depicted in Figure 6.1.

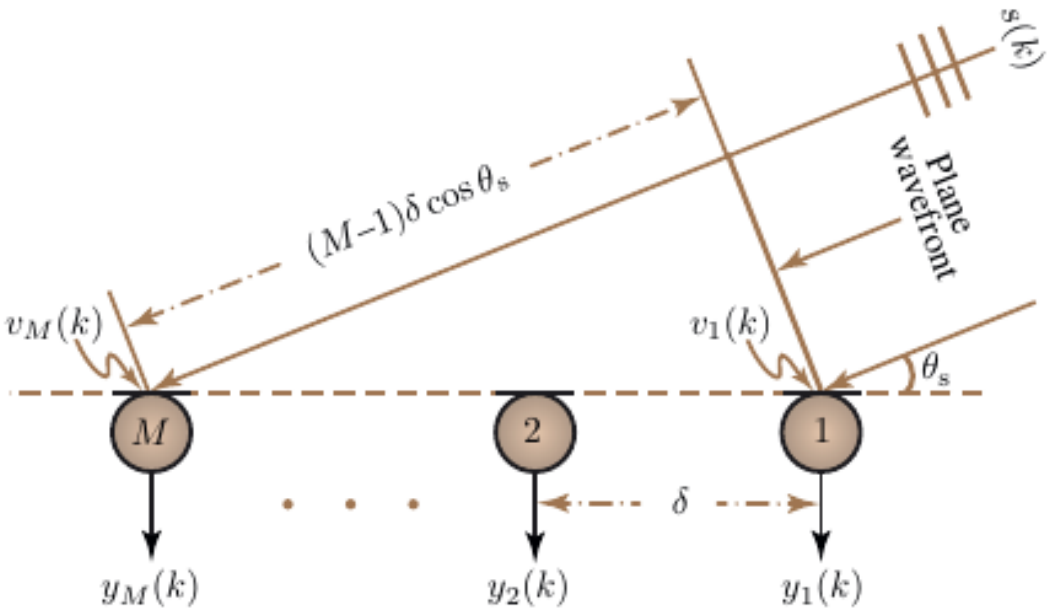


Figure 6.1: Microphone array planar wave scenario

We distinguish between two possible scenarios of interest depending on the number of sensors and the number of sources. In one case, multiple connected microphones are placed in space, forming a microphone array, and

in the same space, only a single speaker is present. The other case is where multiple speakers are present. Naming these use cases, we say that a speaker is referred to as the input to the system, and the microphone array stands for the system's multiple outputs. Therefore, we name the single speaker scenario a Single Input Multiple Output (SIMO) and the latter a Multiple Inputs Multiple Outputs (MIMO). Demonstrations of these use cases are shown in Figures 6.2a and 6.2b, respectively.

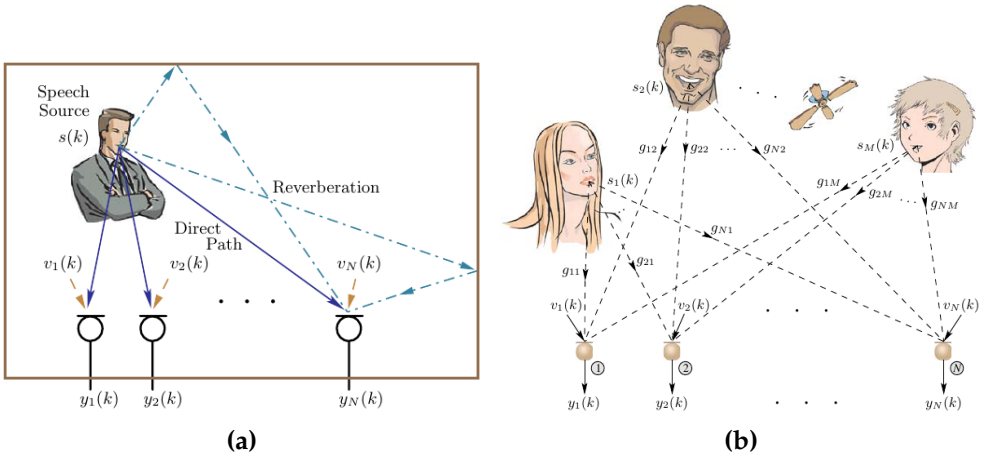


Figure 6.2: (a) Single Input Multiple Output (SIMO) scenario; (b) Multiple Inputs Multiple Output (MIMO) scenario

6.3 Types

6.3.1 Delay-and-Sum

Delay and sum (DAS) beamformer is a basic beamformer that serves as an infrastructure for other beamforming architectures. These beamformers aim to align the speech part from multiple sensors and thus creating the formed beam to conduct constructive interference. The speech is then enhanced while the

other artifacts are attenuated, whether it is ambient noise or other unwanted speech from different speakers.

The general architecture for a DAS beamformer is given in Figure 6.3.

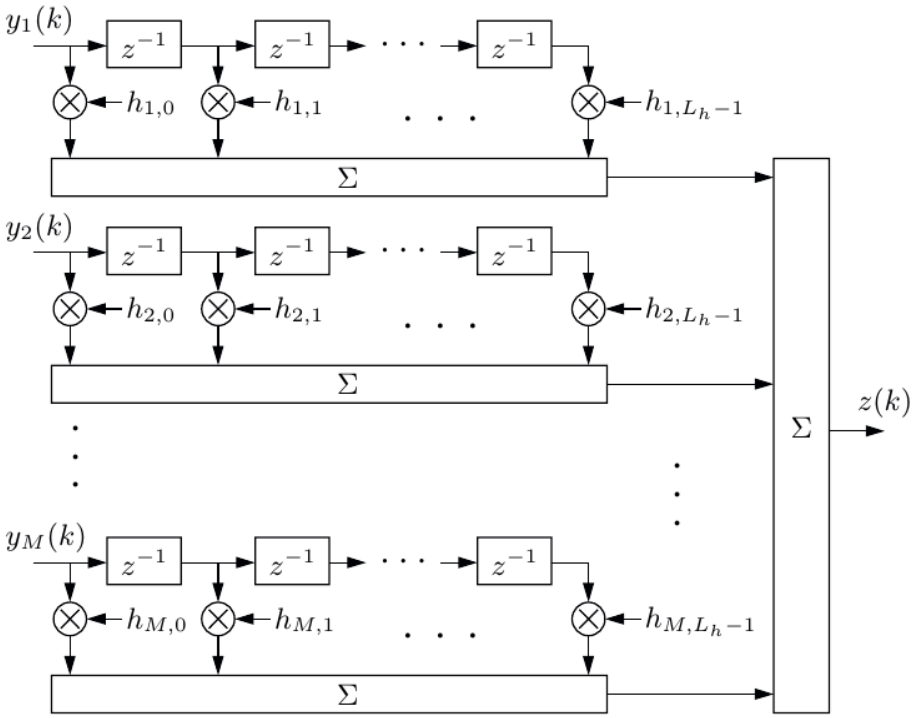


Figure 6.3: Delay-and-Sum Beamformer architecture

Combining the DAS architecture with the microphone array as the input we get a system capable of dealing with SIMO and MIMO, as shown in Figure 6.4.

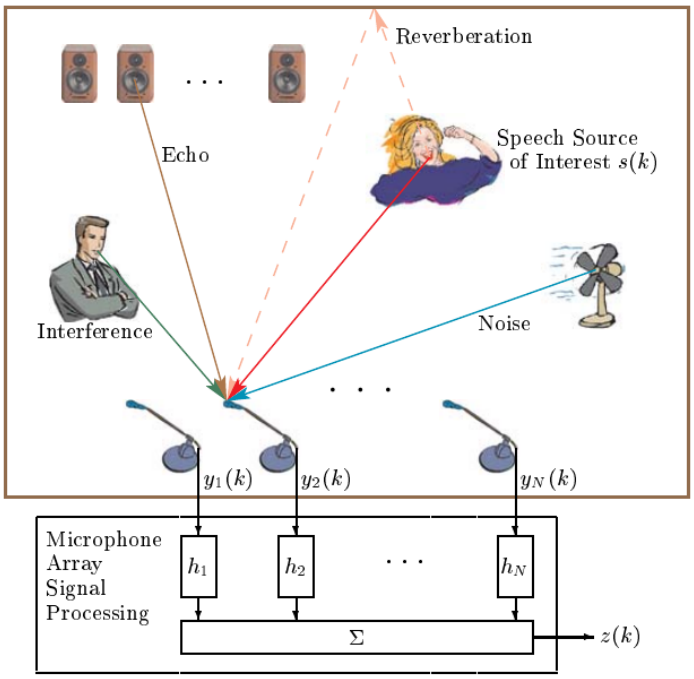


Figure 6.4: Microphone array processing

The coefficients for a DAS beamformer are chosen such that:

$$\mathbf{W}(j\omega) = \frac{1}{M}\mathbf{A}(j\omega) \tag{6.5}$$

Where M is a calculated weight given to each coefficient, and $\mathbf{A}$ is the steering vector that points to the same direction as the direction of arrival.

6.3.2 MVDR – Minimum Variance Distortionless Response

Another type of beamformer is the MVDR beamformer. These beamformers aim to minimize the sampled variance at the beamformer's output. Modeling

a microphone array, we can write the propagation of sound in the form of:

$$
\begin{aligned}
r_1(t) &= s(t - D_1) + n_1(t) \\
r_2(t) &= s(t - D_2) + n_2(t) \\
&\vdots \\
r_m(t) &= s(t - D_m) + n_m(t)
\end{aligned}
\tag{6.6}
$$

Where r_m models the absorbed signals by microphone m, s stands for the desired speech, and n marks all the other artifacts that are considered noise. The lagging time D_m corresponds to the room impulse response h_m. Therefore, we can write the model for absorbed propagation sound as:

$$
r_m[n] = h_m[n] \star s[n] + n_m[n] \tag{6.7}
$$

Transforming Equation 6.8 to frequency domain, we get:

$$
\mathbf{R}_m(j\omega; t) = \mathbf{H}_m(j\omega)\mathbf{S}(j\omega; t) + \mathbf{N}_m(j\omega; t) \tag{6.8}
$$

MVDR works on minimizing the variance at the output resulting from the summation of the speech and noise variations at the inputs. Therefore, we need a notation for the variances and covariances to model the MVDR beamformer.

Equation 6.9 shows the covariance matrices of the speech, noise, and the

extracted room impulse response, respectively.

$$\mathbf{R}_{SS} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{S}(j\omega;t)\mathbf{S}^{\acute{H}}(j\omega;t)$$

$$\mathbf{R}_{NN} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{N}(j\omega;t)\mathbf{N}^{\acute{H}}(j\omega;t)$$

$$\mathbf{R}_{HH} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{H}(j\omega)\mathbf{H}^{\acute{H}}(j\omega)|\mathbf{S}(j\omega;t)|^2 \tag{6.9}$$

Hence, an MVDR beamformers coefficients can be selected following:

$$\mathbf{W}(j\omega) = \frac{\mathbf{R}_{XX}^{-1}(j\omega)\mathbf{A}(j\omega)}{\mathbf{A}^{H}(j\omega)\mathbf{R}_{XX}^{-1}(j\omega)\mathbf{A}(j\omega)} \tag{6.10}$$

Similar to the DAS beamformer, here, $\mathbf{A}(j\omega)$ denotes the steering vector.

6.3.3 GEV – Generalized Eigenvalue

The GEV beamformer is based on the principle of generalized eigenvalues decomposition so that the beamformer's coefficients are set accordingly by the extraction of the eigenvalue vector λ.

$$\mathbf{R}_{SS}(j\omega)\mathbf{W}(j\omega) = \lambda\mathbf{R}_{NN}(j\omega)\mathbf{W}(j\omega) \tag{6.11}$$

GEV beamformers are favorable over MVDR and DAS beamformers especially when the microphone array formation is unknown. Another case when we should consider a GEV beamformer is when modeling the steering vector becomes a factor due to complex room impulse responses.

Chapter 7

ASR – Automatic Speech Recognition

7.1 Introduction

Speech recognition (SR) stands for recognizing natural speech and translating it to readable text named transcript. Often speech recognition is referred to as Speech-to-Text (STT). Speech recognition that is accomplished automatically by a computing machine or software is known as ASR (Automatic Speech Recognition).

ASR systems employ numerous algorithms and techniques to conduct the speech to text translation. Some algorithms aim for precision improvements and higher detection ratios measured with *WER*, *CER* and *PER*, as described in Chapter 4. Other algorithms target increasing the system's robustness, which means how well the system can still function adequately given that the environmental conditions are not static and change.

Naturally spoken speech has inherent variabilities depending on many factors that make a speech signal classified as non-stationary and inconsistent.

Natural speech heavily depends on the speaker's gender, whether a male or a female, and the speaker's age. Moreover, different speakers of the same gender and age range may still present variances in the vocal range, pitch, and formant frequencies. Likewise, we can say that gender-independent characteristics such as accent, style of speech, emotional state, or health conditions, which are examples of social and geographical factors, all impact naturally spoken sentences and introduce some degree of variability.

Due to the high dependency that natural speech has on so many factors, an ASR system must have high endurance to fluctuations in these previously mentioned environments and states. Also, the ability to perform adequately in a wide range of scenarios is very crucial. In particular, the cases of high background noise levels in low SNR scenarios and multiple reverberations due to poor room acoustics.

Traditional ASR Engines are usually built of several modules designated for a specific task. An ASR Engine as a whole is a chain of these modules connected in a pipeline. A common structure of an ASR system is shown in Figure 7.1a. This type of ASR system translates a speech waveform into a transcript by chaining *phonemes* together to construct words.

A phoneme is a discrete and distinctive unit of language that can be used to differentiate between words. A word is a sequence of phonemes chained together in how the word is actually pronounced. The motivation of detecting phonemes instead of entire words by the ASR Engine relies on the fact that the number of words in a common language crosses the tens of thousands and sometimes even larger than hundreds of thousands. Extensive word

vocabulary implies a huge training dataset requirement covering most of the words in a selected language. On the contrary, most languages have 20-60 phonemes [1, 3]. In that way, a much smaller set of dedicated phonemes can be used to construct the vocabulary utilizing a less demanding training dataset requirement.

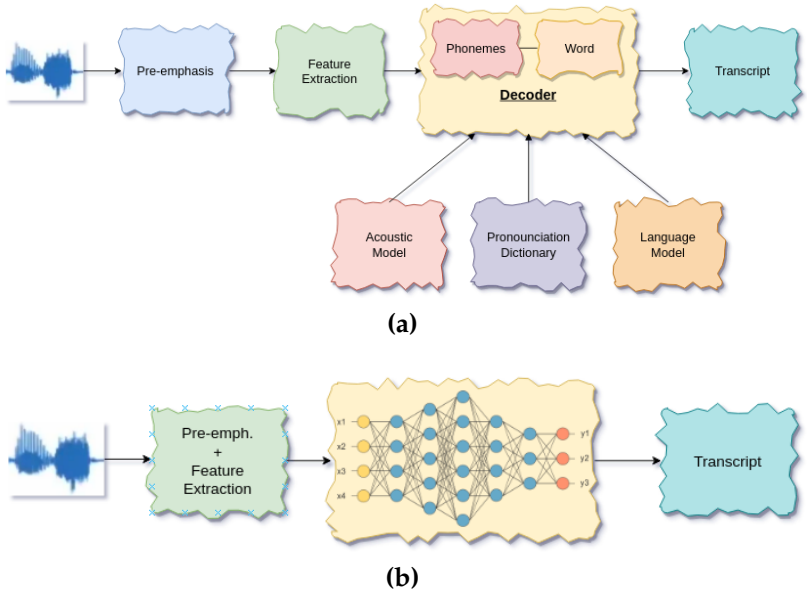


Figure 7.1: (a) Traditional ASR system blocks diagram; (b) End-to-end ASR system blocks diagram.

The system presented in Figure 7.1a shows a pipeline that starts with a pre-emphasis and features extraction modules. The pre-emphasis includes the speech analysis, noise cancellation, speech enhancements, windowing, and framing of the speech into sub-frames in which it is closer in characteristics to a stationary signal. The feature extraction module extracts the individual features representing the speech from the incoming frames as MFCCs, FilterBanks, and other techniques.

The next module in line is the decoder. The decoder's responsibility is to take the features, convert them to a set of phonemes and chain them together to detectable words, composing the transcript at the output. A decoder is built from three sub-models tied together that work in harmony. The first is the Acoustic Model (AM), which maps the acoustic features extracted in the earlier module to phoneme sequences with probabilistic weights. These phonemes are an intermediate representation in the process of word decoding. An additional module is the pronunciation model. This module is a dictionary, handcrafted by an expert linguist and tailor-made to each language explicitly. This dictionary links phoneme sequences to words. Lastly, the Language Model (LM) calculates the likelihood a given word is detected based on the perceived phoneme sequences and how likely it is for this word to occur given the past temporal context. The words or N-Grams with the highest scores are selected and written into the output transcript.

A newer more modernized approach known as End-to-End Automatic Speech Recognition (E2E ASR) has been proposed in [4]. This approach replaces the conventional decoder structure with a deep neural network of some kind that maps acoustic features to characters. An E2E ASR system structure is presented in Figure 7.1b.

With this architecture, there is no longer a need for expert dictionaries or complex models of chaining phonemes to words. That spares the need to acquire new pronunciation models or maintain an existing one, which can be expensive. Instead, character probabilities are given at the output according to the learning process of mapping acoustic features to corresponding letters.

That leads to another advantage of E2E ASR systems over traditional ASRs. Retraining an E2E model with a new dataset of annotated speech in different languages is possible without changing the architecture at all. On the other hand, using E2E systems means a more extensive training dataset is required for the model to generalize well, only to maintain comparable detection ratios in terms of WER and CER as the traditional ASRs do.

However, despite having an E2E system trained against a huge dataset, the direct estimations of characters and, later on, the construction of a full transcript don't work as expected compared to traditional ASRs. To overcome the degradation in performance, several algorithms can be applied to improve different aspects of the model. Such a helpful technique that is called CTC (Connectionist Temporal Classification) by Maas et al. is presented in Reference [5]. For example, this method demonstrates which characters get the highest probability for specific phonemes, as shown in Figure 7.2.

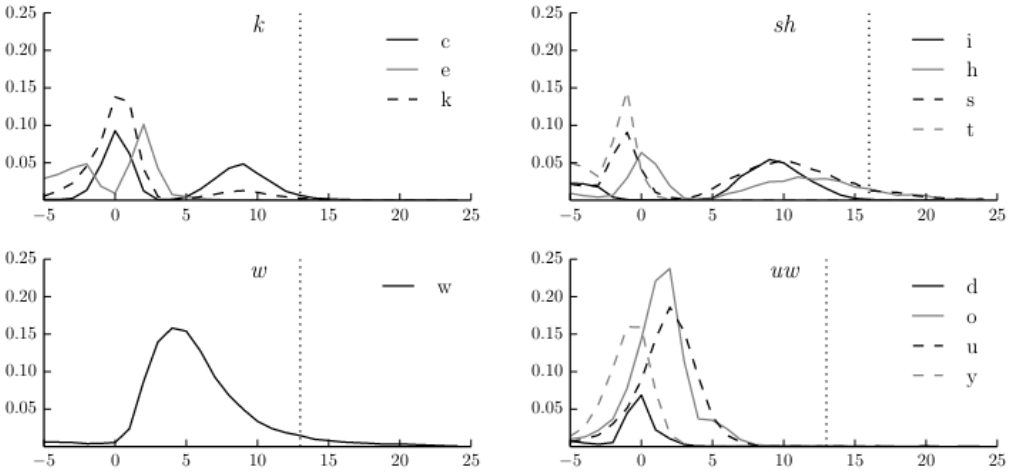


Figure 7.2: Maas et al. [5] phonemes vs characters graphs

Additional paradigm by Chan et al. known as Seq2Seq (Sequence-to-Sequence) or sometimes as Attention Encoder-Decoder (AED) networks is presented in [2]. This approach is based on an LSTM transducer. A more advanced model that is known for its self-attention mechanism was later suggested by Vasawani et al. in 2017 [7]. This model, which also uses the Encoder-Decoder architecture, is called a transformer.

A sophisticated combination of these techniques is practically used in this work to construct and evaluate the different ASR Engines.

7.2 The ASR Engine

The proposed general architecture for the suggested ASR Engines listed in Table 7.1 is constructed with a CNN front-end connected to a self-attention based transformer, implying an Encoder-Decoder infrastructure.

7.2.1 Front-End

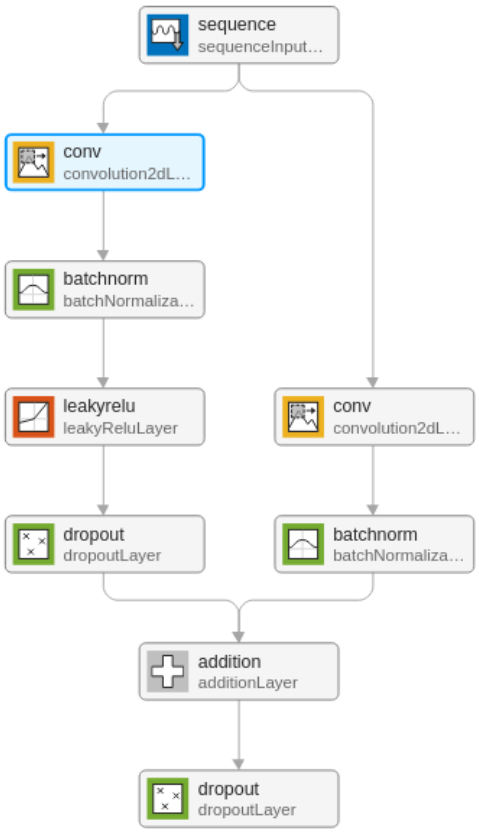


Figure 7.3: CNN front-end general ConvBlock architecture.

7.2.2 Transformer

A transformer is a self-attention Encoder-Decoder based model whose architecture is shown in Figure 7.4. The left side is the Encoder, and the right side is the Decoder.

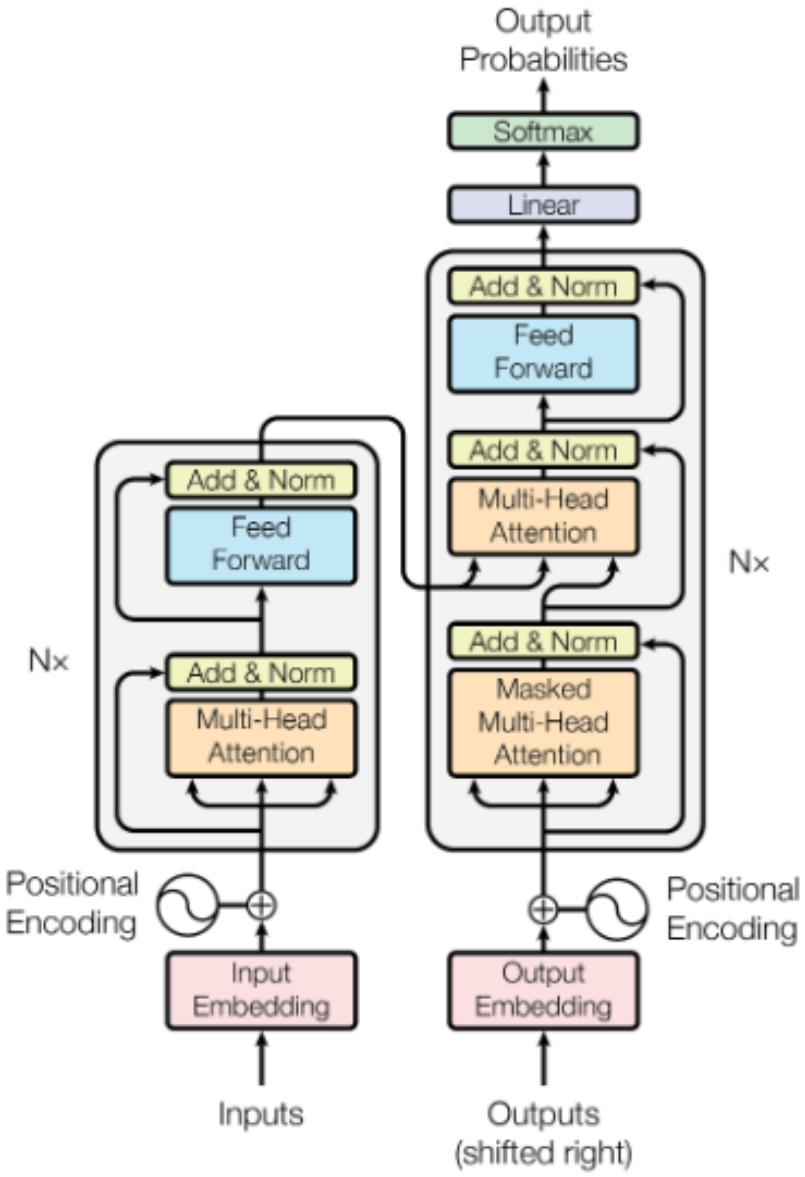


Figure 7.4: Transformer

Source: Adapted from "Attention Is All You Need" [7]

The attention function that translates the Query (Q) and the keys-values pairs (K), (V), respectively, can be implemented in Scaled Dot-Product Attention or in Multi-Head Attention. The transformer backend used in this work

is a Multi-Head Attention-based utilizing eight heads in the attention model. The Encoder and Decoder parts consist of a variable number of linear layers ranging from four to eight. The activation function used with the transformer is the *GELU* (Gaussian Error Linear Unit).

7.2.3 CTC

The CTC loss is the negative logarithm of the probabilities that resulted from the CTC algorithm. Minimizing the CTC loss is in fact selecting the characters that end up with the highest probabilities. As a result, the predicted label collapses, meaning that any repeated characters not separated by the "blank" character unite into a single character.

$$\hat{\theta} = arg\min_{\theta} -\sum_{i=1}^{N} \left[\sum p\left(\pi | x^{(i)}; \theta\right) \right] \tag{7.1}$$

7.2.4 Sequence-to-sequence — Seq2Seq

Sequence to sequence (Seq2seq) [2], or more commonly known as encoder-decoder, is a DNN model that translate a sequence in one domain at the encoder's input into another sequence at the decoder's output.

We use the Kullback-Leibler Divergence (kldiv) loss function for the Seq2seq model. The kldiv loss function is given by:

$$\ell(x, y) = L = \{\ell_1, \ell_2, ..., \ell_N\} \tag{7.2}$$

Where:

$$\ell_n = y_n \left[\log\left(y_n\right) - x_n\right] \tag{7.3}$$

The kldiv works with the log-probabilities and can be reduced with respect to the mini-batch size, as follows:

$$\ell(x,y) = \begin{cases} \frac{\sum L}{batch\_size}, & red = meanbatch \\ \sum L, & red = sum \end{cases} \tag{7.4}$$

Combining both the CTC and the Seq2Seq components, the engine's loss function used for training is as follows:

$$\ell = \omega_{ctc}\ell_{ctc} + (1 - \omega_{ctc}) \cdot \ell_{seq} \tag{7.5}$$

Where ω_{ctc} is set to 0.3 and a greater weight of 0.7 is given to the Seq2Seq loss ℓ_{seq}.

Id(seed)	Feature(s)	Params	Scale	Fbank	#Filt.	#Coeff.
Root Mean Cepstral Coefficients						
#1(5)	RMFCC(0.1), Δ, ΔΔ	146.2M	Mel	Bartlett	80	20
#2(15)	RMFCC(0.08), Δ, ΔΔ	146.2M	Mel	Bartlett	80	20
#3(12)	RBFCC(0.1), Δ, ΔΔ	146.0M	Bark	Bartlett	28	18
#4(13)	RGFCC(0.1), Δ, ΔΔ	146.0M	ERB	Gammatone	28	18
Conventional Cepstral Coefficients						
#5(70)	MFCC, Δ, ΔΔ	146.2M	Mel	Bartlett	80	20
#6(72)	MFCC, Δ, ΔΔ	146.0M	Mel	Bartlett	28	18
#7(22)	BFCC, Δ, ΔΔ	146.0M	Bark	Bartlett	28	18
#8(23)	GFCC, Δ, ΔΔ	146.0M	ERB	Gammatone	28	18
#9(100)	MFCC, Δ, ΔΔ, Context(3, 3)	78.8M	Mel	Bartlett	26	26
#10(101)	MFCC, Δ, ΔΔ, Context(3, 3)	78.8M	Mel Approx.	Bartlett	26	26

Table 7.1: ASR Engines Table

ASR Engines #9, #10 were heavily optimized in terms of the neural network structures, leading to smaller models having two times fewer learnable parameters. For comparison, differences between ASR Engines #10 and #6 are summarized in Table 7.2.

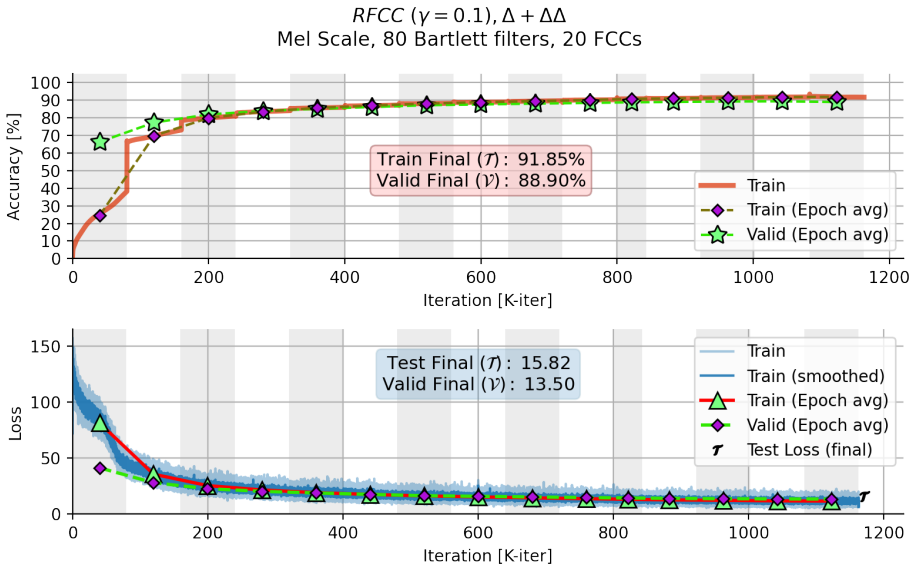
Parameter	Engine #10	Engine #6
CNN Settings		
CNN Blocks [#]	3	3
CNN Shapes	[64, 100, 128]	[128, 200, 256]
Transformer Settings		
FFN layers	2048	3072
Input size	2560	3584
Enc. layers	8	12
Dec. layers	4	6
Implementation Settings		
Filters [#]	26	28
Coeff. [#]	2048	3072
Precision	U16/8	FP64

Table 7.2: ASR Engines #10, #6 comparison table

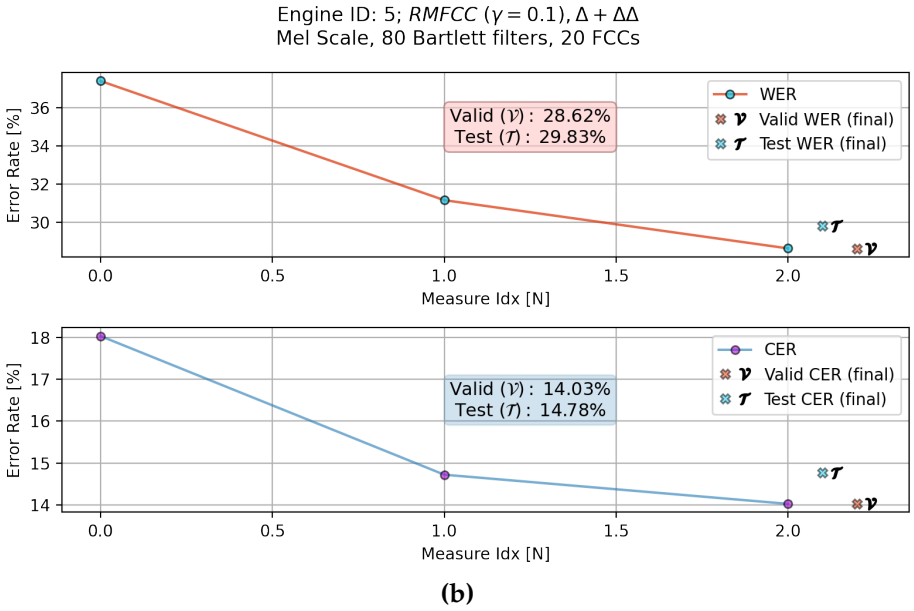
7.2.5 Trained Engines

7.2.5.1 ASR Engine #1 – RFCC(0.1), Δ, $\Delta\Delta$

ASR Engine #1 had been trained over fifteen epochs, against a $400K$ recordings table, with a minibatch size of four per iteration. The engine was trained against the RFCCs, when $\gamma = 0.1$, with the Δ, $\Delta\Delta$ features, utilizing the Log-Mel scale. A preprocessing STFT enframing block was set with 80 Bartlett filters for each T-F bin. Overall the number of features per T-F unit is 20 cepstral coefficients plus the first and second derivatives. Figure 7.5a shows ASR Engine #1 training results for the test and validation subsets. In Figure 7.5b, we present the evaluated WER and CER measures taken every five epochs during the training process.



(a)



(b)

Figure 7.5: (a) ASR #1 training accuracy and loss plot; (b) ASR #1 WER, CER evaluation plot.

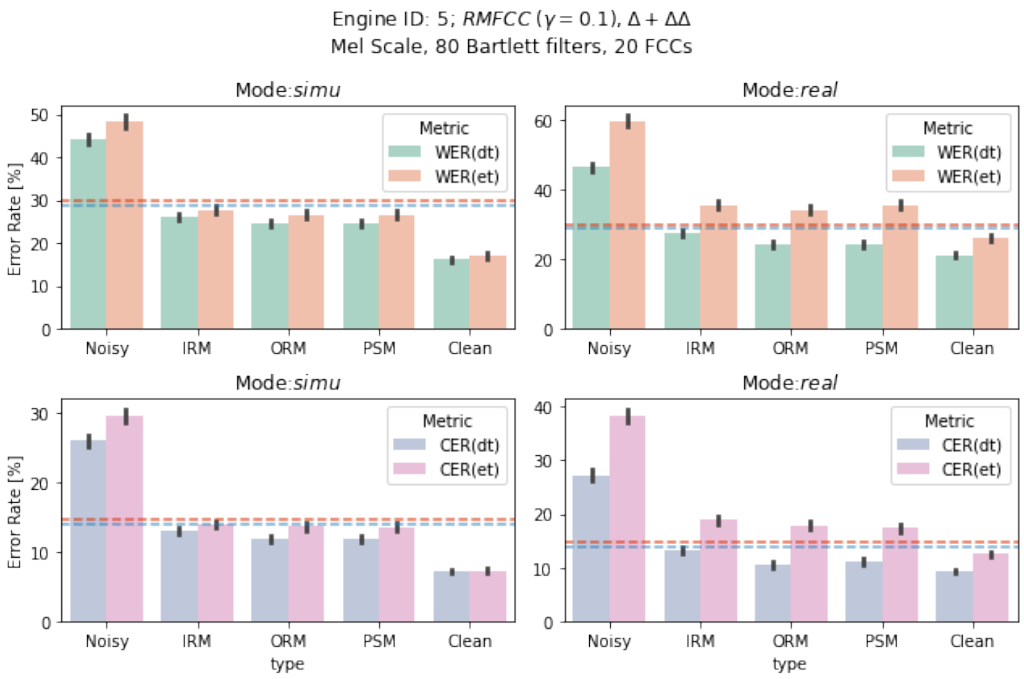


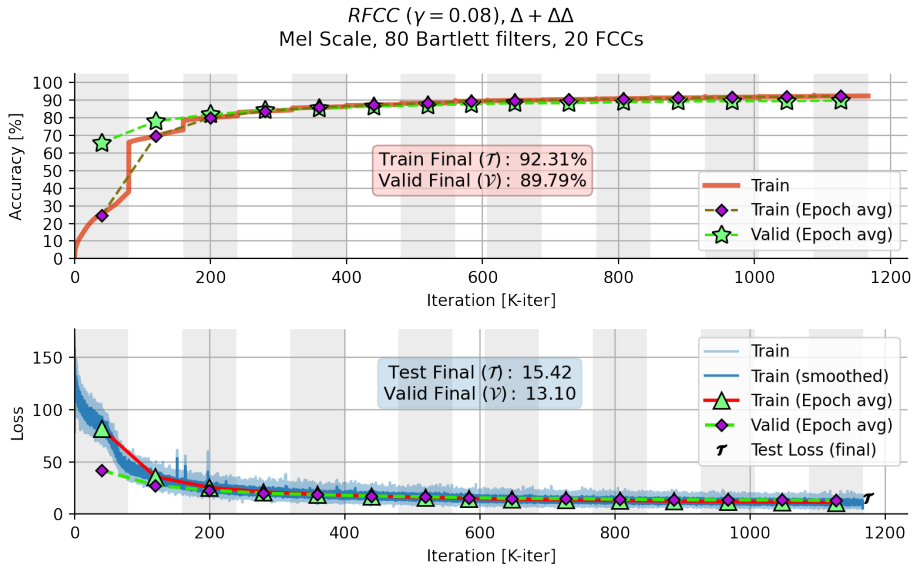
Figure 7.6: ASR #1 WER, CER vs. T-F masks, noisy and clean inputs

In Figure 7.6, we present the engine's results in terms of WER and CER per each subset of the multi microphone CHiME dataset. This figure emphasizes the comparison between the four T-F masking algorithms. Results are plotted side-by-side with the evaluated noisy mixture and clean speech metrics as reference. Two horizontal lines, red and blue, are stretched across the x-axis of each plot. These lines mark the plot's metric, whether WER or CER, taken from the training evaluations of CommonVoice datasets validation and test subsets. The red line is for the test measurement and the blue line marks the validation measurement.

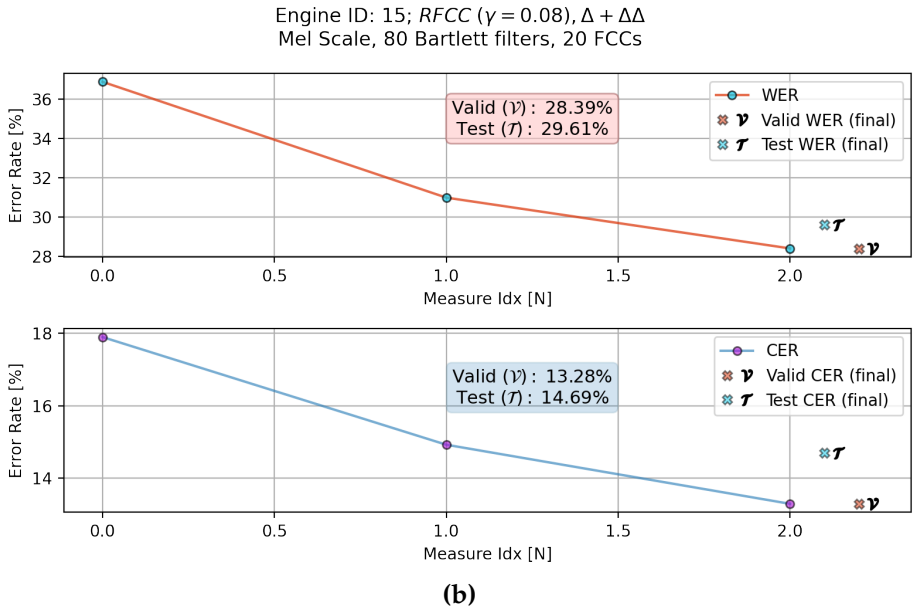
7.2.5.2 ASR Engine #2 – RMFCC(0.08), Δ, $\Delta\Delta$

ASR Engine #2 had been trained over fifteen epochs, along a $400K$ recordings table, with a minibatch size of four per iteration. The engine was trained against the RMFCCs, utilizing the Log-Mel scale with the Δ, $\Delta\Delta$ features. Gamma has been set to $\gamma = 0.08$, as this setting proved to be the optimized value as stated in [6]. The rest of the settings are similar to ASR Engine #1.

Figures 7.7a and 7.7b show the training results of ASR Engine #2 and its WER and CER benchmarks for the validation and test subsets.



(a)



(b)

Figure 7.7: (a) ASR #2 training accuracy and loss plot; (b) ASR #2 WER, CER evaluation plot.

Figure 7.8 presents the T-F mask based beamformed enhanced recordings evaluations for ASR #2.

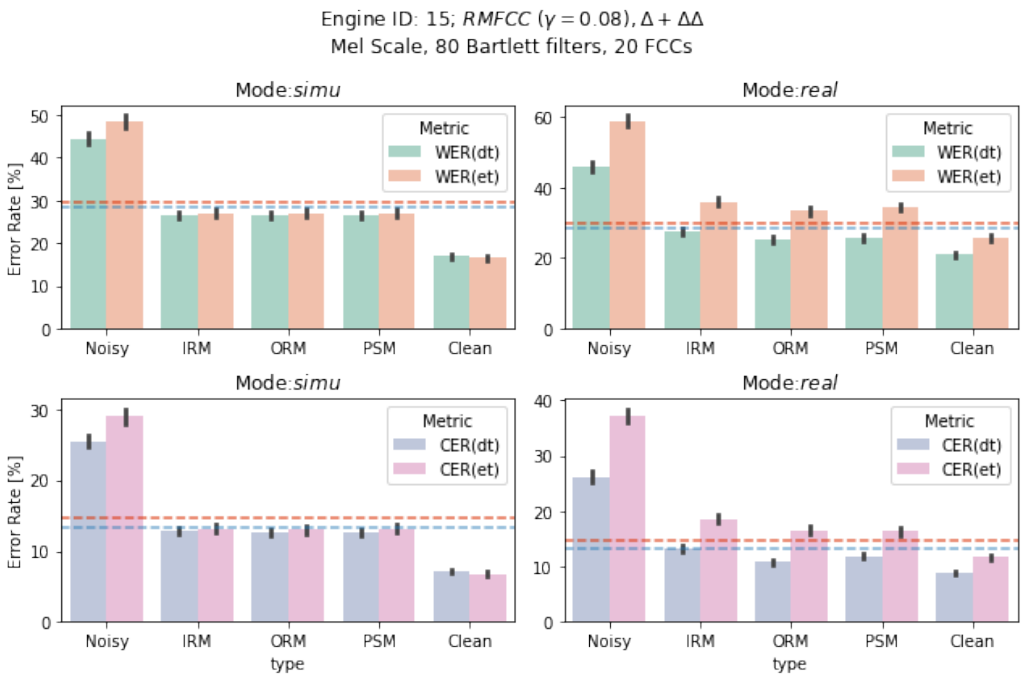
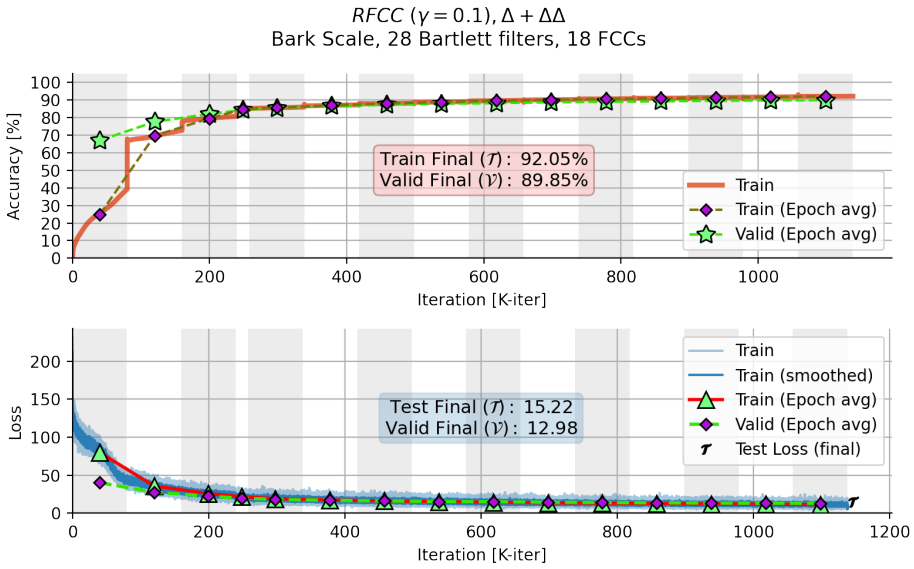


Figure 7.8: ASR #2 WER, CER vs. T-F masks, noisy and clean inputs

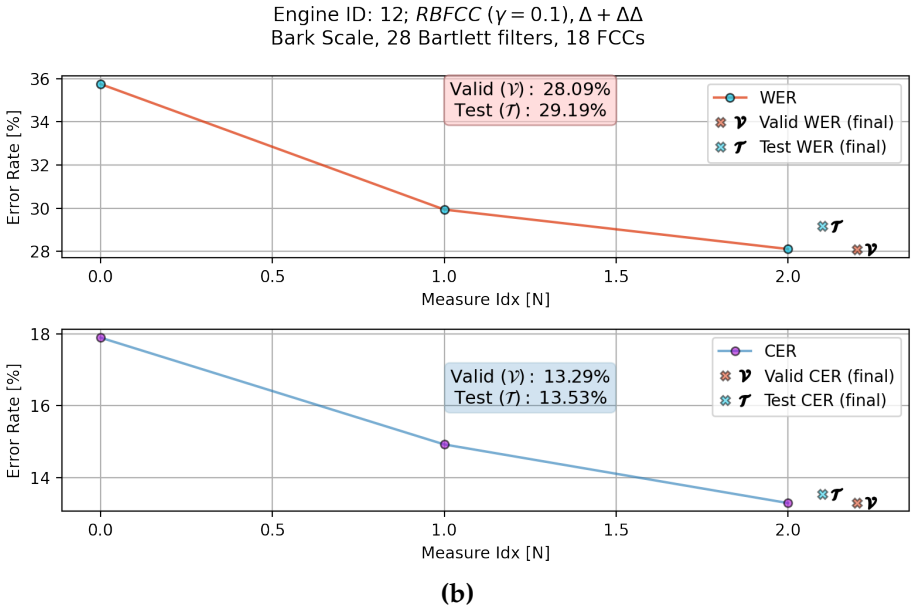
7.2.5.3 ASR Engine #3 – RBFCC(0.1), Δ, $\Delta\Delta$

ASR Engine #3 trained similarly to the previous engines, #1 and #2 but with some distinctions. The feature set utilizes the Bark scale instead of the previously used Log-Mel scale, and the number of extracted cepstral coefficients has been reduced to 18. The number of Bartlett filters per T-F unit has been set to 28 instead of 80.

Figures 7.9a and 7.9b show the training results, and Figure 7.10 presents the beamforming effect per T-F algorithm, compared to the noisy mixture and the clean speech references.



(a)



(b)

Figure 7.9: (a) ASR #3 training accuracy and loss plot; (b) ASR #3 WER, CER evaluation plot.

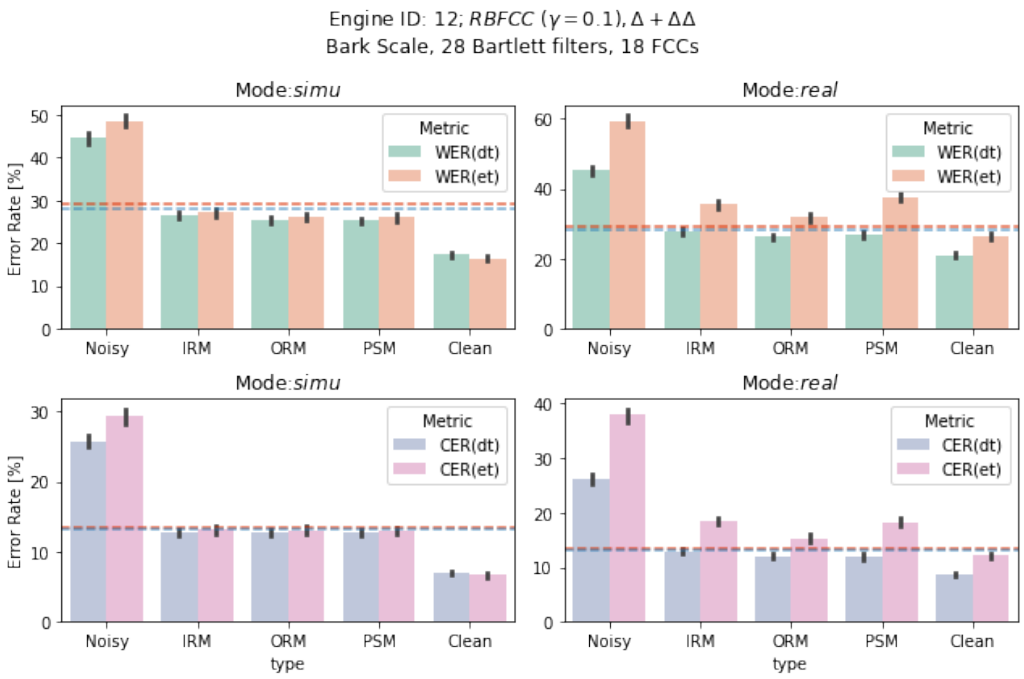
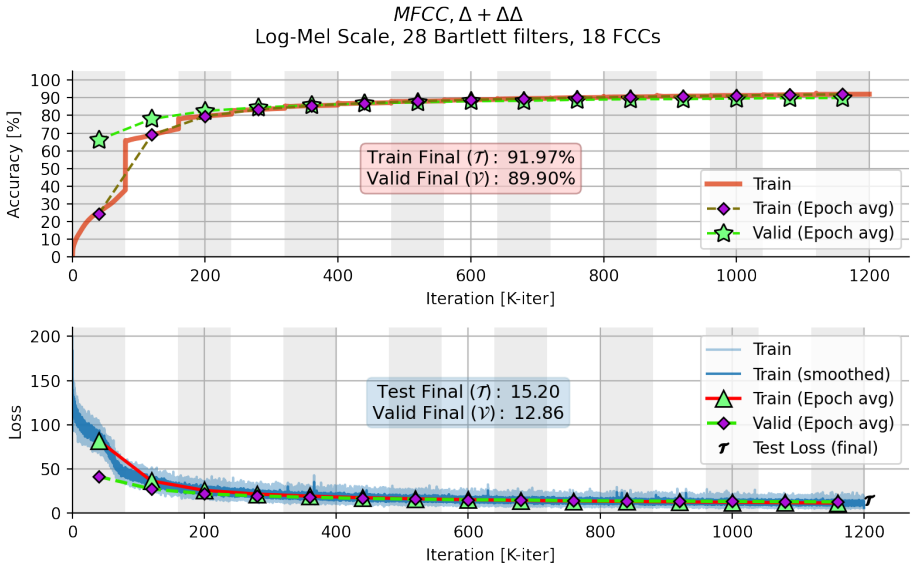


Figure 7.10: ASR #3 WER, CER vs. T-F masks, noisy and clean inputs

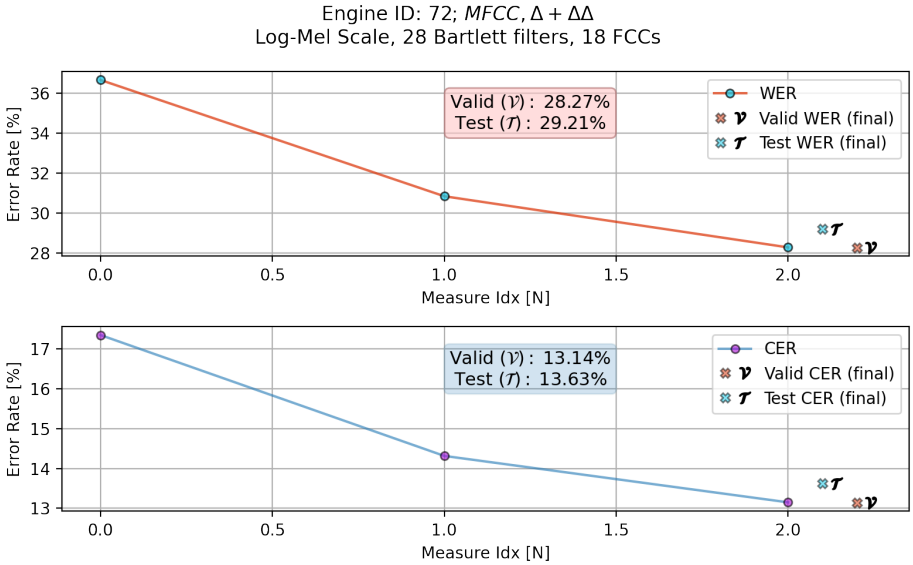
7.2.5.4 ASR Engine #6 – MFCC, Δ, $\Delta\Delta$

ASR Engine #6 as opposed to ASR Engine #3, had been trained against the Mel cepstral coefficients. All the other settings are retained.

Figures 7.11a, 7.11b, and 7.12 present the training, evaluation, and the T-F based beamforming results, respectively.



(a)



(b)

Figure 7.11: (a) ASR #6 training accuracy and loss plot; (b) ASR #6 WER, CER evaluation plot.

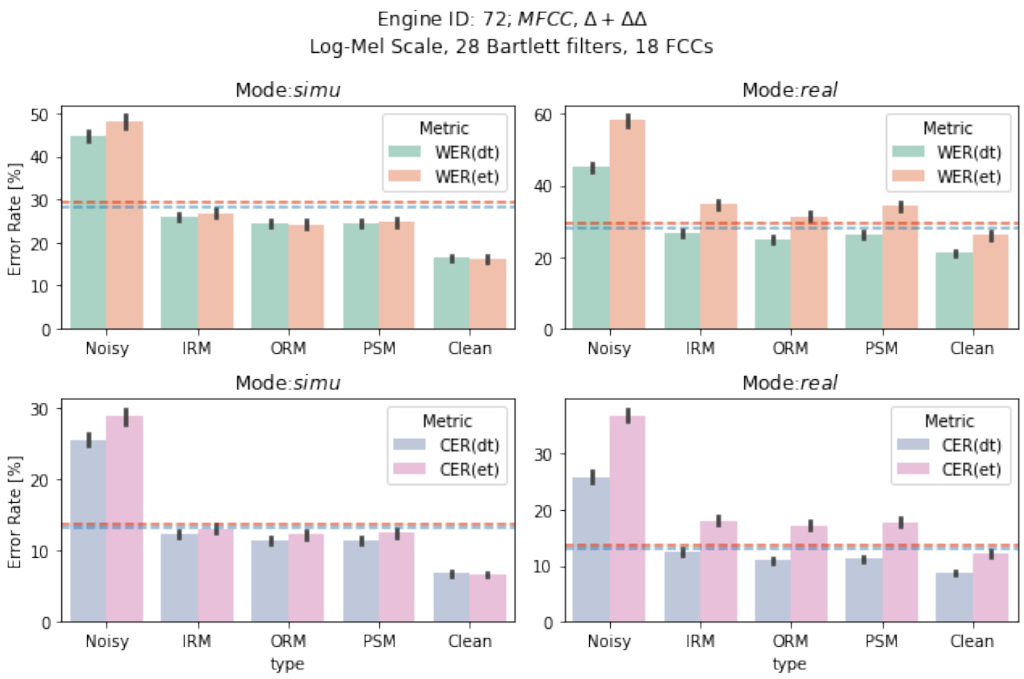


Figure 7.12: ASR #6 WER, CER vs. T-F masks, noisy and clean inputs

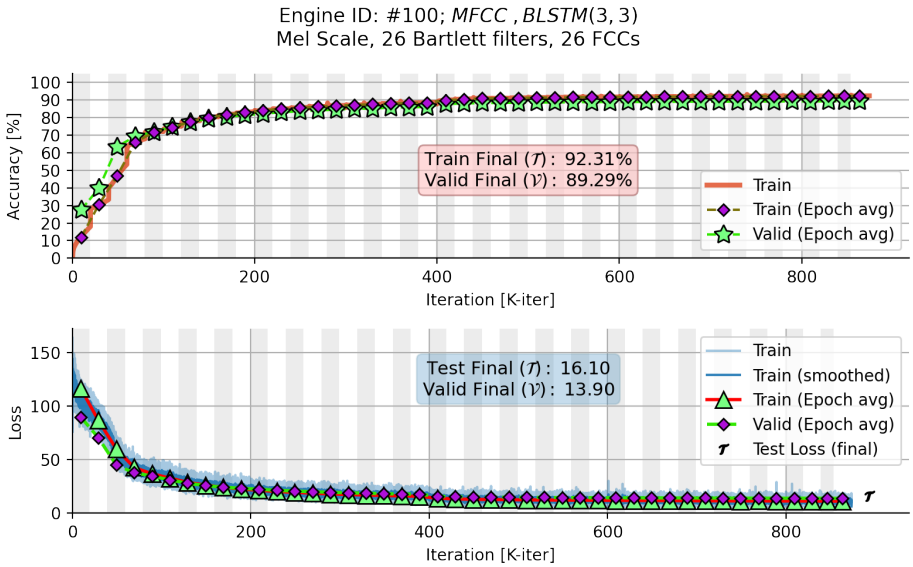
7.2.5.5 ASR Engine #9 – MFCC, $\Delta, \Delta\Delta$, Context(3, 3)

ASR Engine #9 is an optimized Log-Mel scale based engine that has been trained against the Mel cepstral coefficients with their first and second derivatives. These extracted features were also taken from six additional adjacent T-F units, three prior and another three past the currently analyzed unit. The number of extracted coefficients is set to 26, same as the total number of Bartlett filters covering the spectrum of each T-F bin.

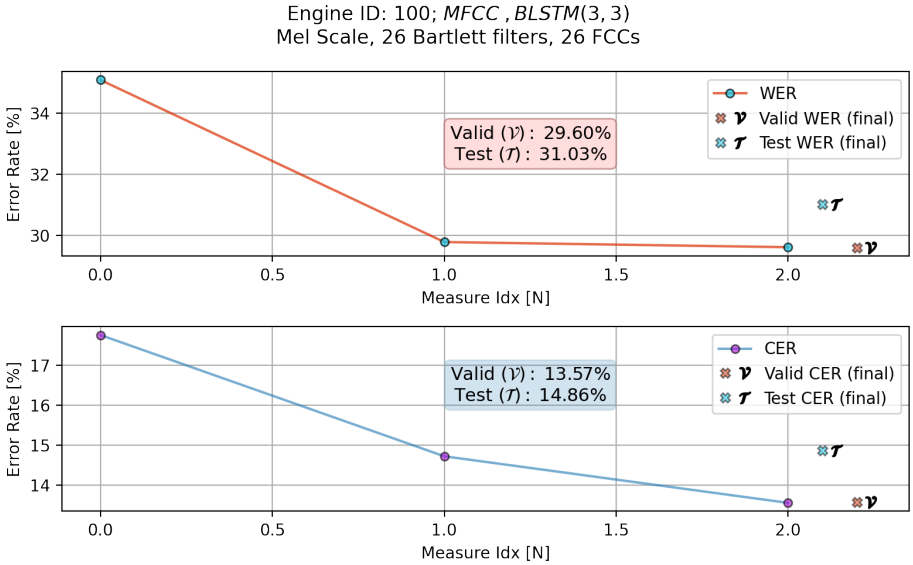
In contrast to the previous engines, engine #9 underwent heavy optimization to fit a target hardware device. In the process, the engine's learnable parameters were quantized to the U16/8 format from the default 64bit floating point.

The training process updated to over 40 epochs with $100K$ randomly selected recordings from the entire CommonVoice dataset. The motivation behind these changes in the training process is to let the model better generalize in a reasonable training time because of the introduction of the additional temporal context features vectors and the quantized architecture.

Figures 7.13a, 7.13b, and 7.14 present the training, evaluation, and the T-F based beamforming results, respectively.



(a)



(b)

Figure 7.13: (a) ASR #9 training accuracy and loss plot; (b) ASR #9 WER, CER evaluation plot.

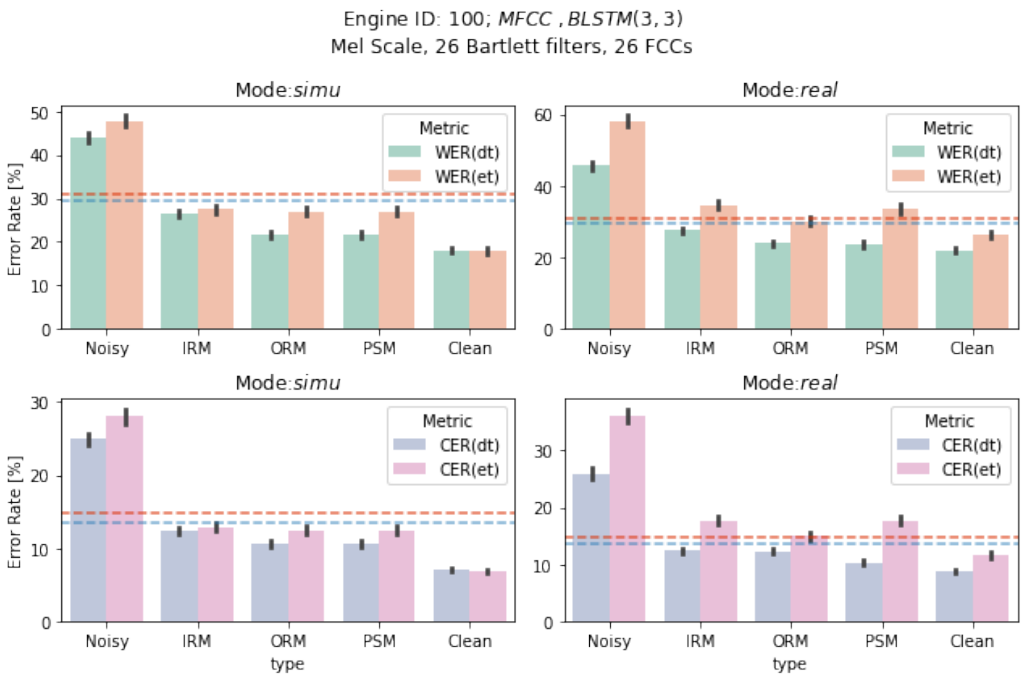
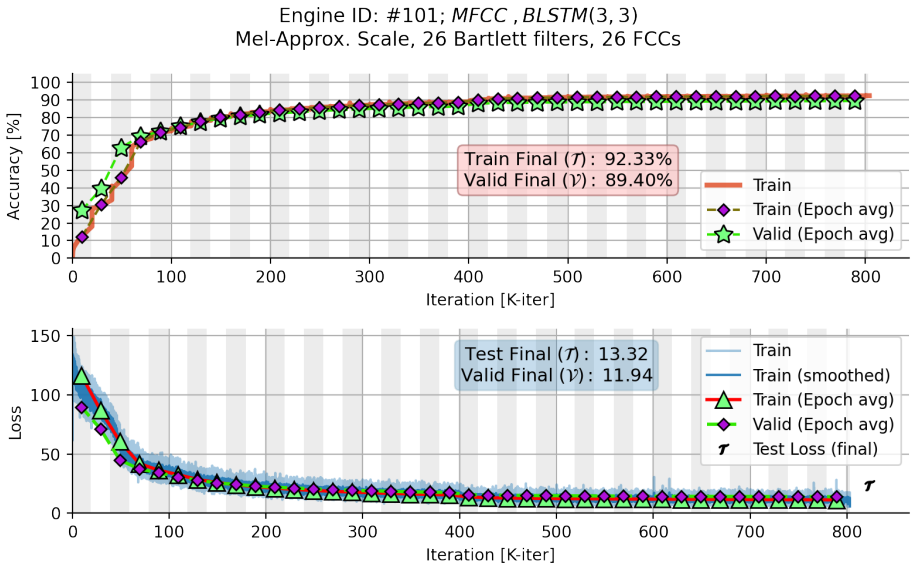


Figure 7.14: ASR #9 WER, CER vs. T-F masks, noisy and clean inputs

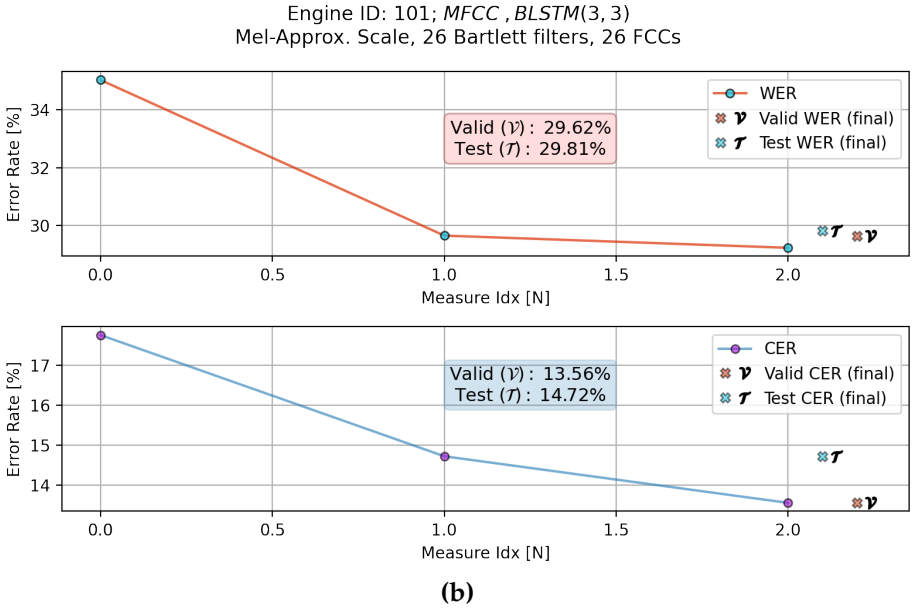
7.2.5.6 ASR Engine #10 – Approx. Mel-FCC, $\Delta, \Delta\Delta$, Context(3, 3)

ASR Engine #10 is identical to ASR Engine #9 except for the actual implementation of the Mel scale. This engine has been trained against the same features utilizing the linear approximation of the Mel scale as described in Chapter 3.

Figures 7.15a, 7.15b, and 7.16 present the training, evaluation, and the T-F based beamforming results, respectively.



(a)



(b)

Figure 7.15: (a) ASR #3 training accuracy and loss plot; (b) ASR #3 WER, CER evaluation plot.

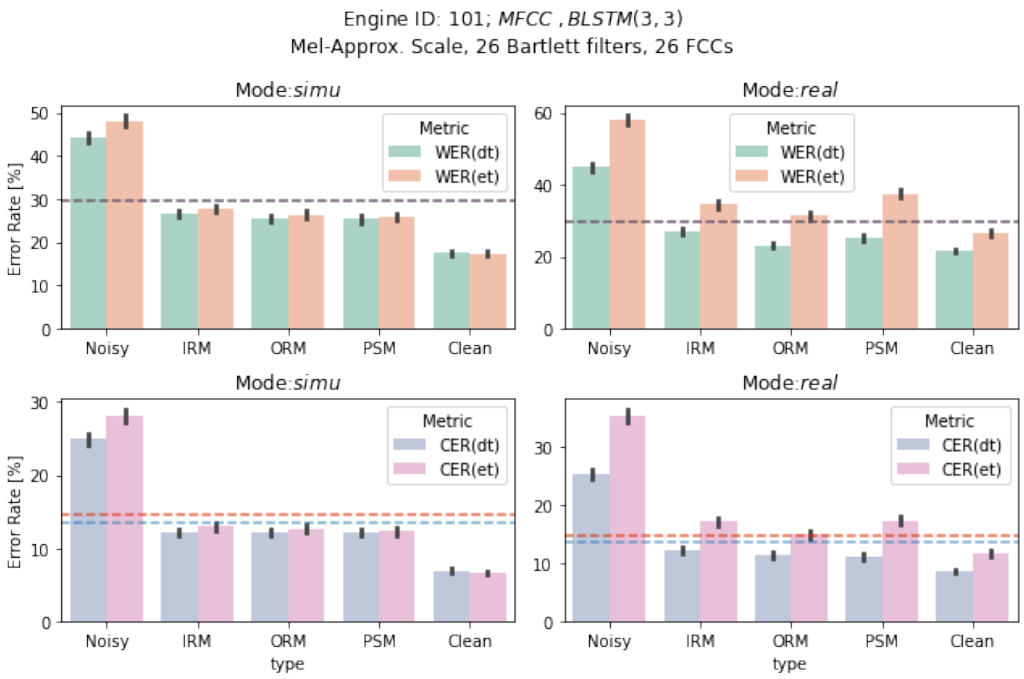


Figure 7.16: ASR #10 WER, CER vs. T-F masks, noisy and clean inputs

7.2.5.7 Other ASR Engines

ASR Engines #4, #5, #7, and #8 were not tested with the enhanced beamformed recordings. The reason is that the performance of these ASR Engines is degraded compared to the previously described ASR Engines. However, for completeness, we present their training results and some WER, CER measures for the training, validation, and test subsets of the CommonVoice dataset.

Figures 7.17 and 7.18 present the training results and the WER and CER metrics measures for ASR Engine #7.

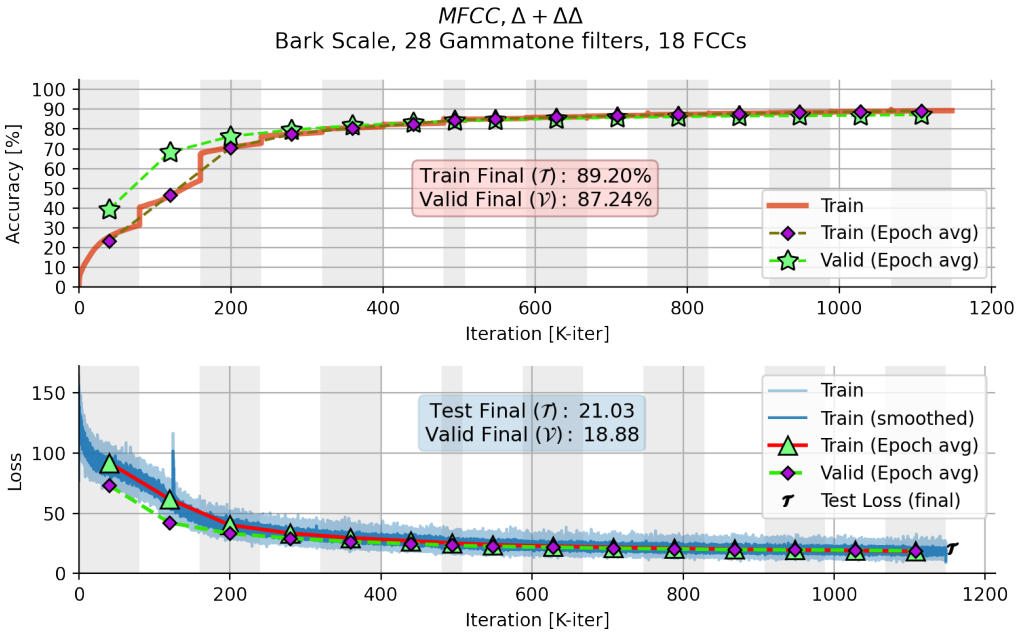


Figure 7.17: ASR #7 training accuracy and loss plot

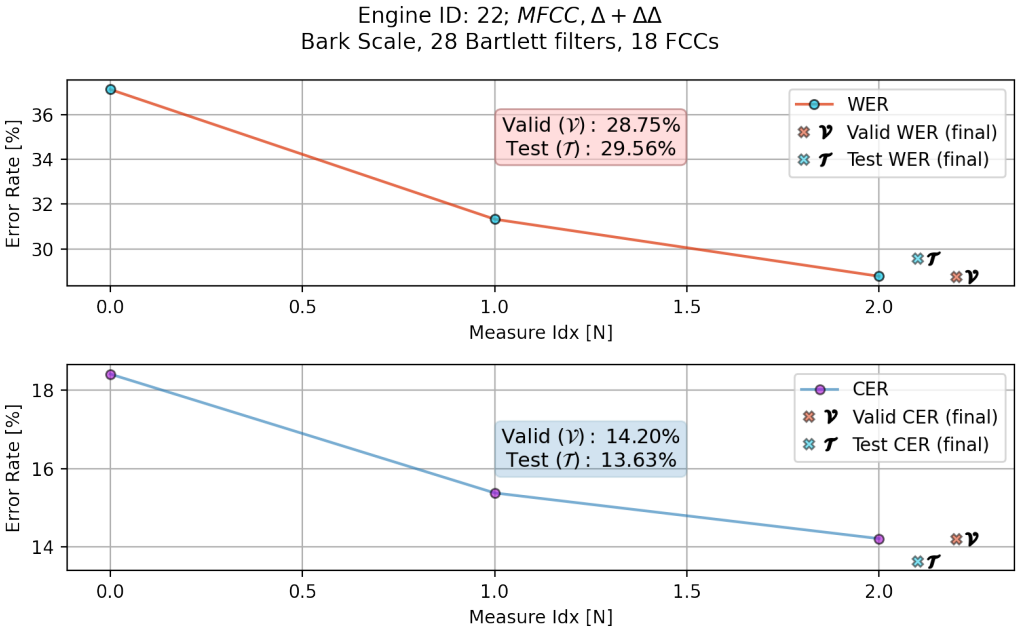


Figure 7.18: ASR #7 WER, CER evaluation plot

Figures 7.19, 7.20 present the training results and the WER and CER metrics measures for ASR Engine #8.

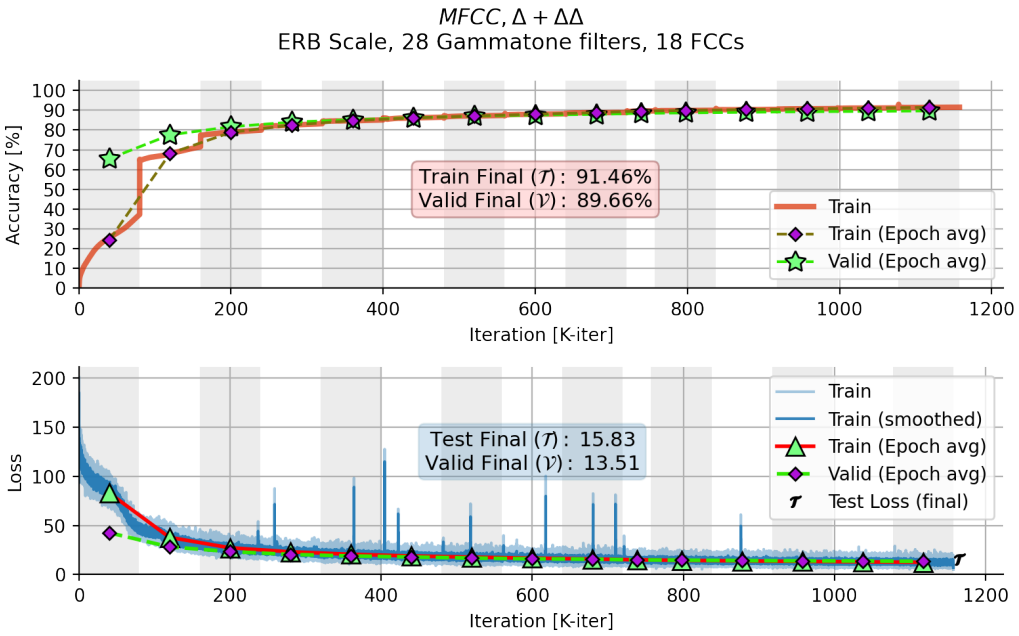


Figure 7.19: ASR #8 training accuracy and loss plot

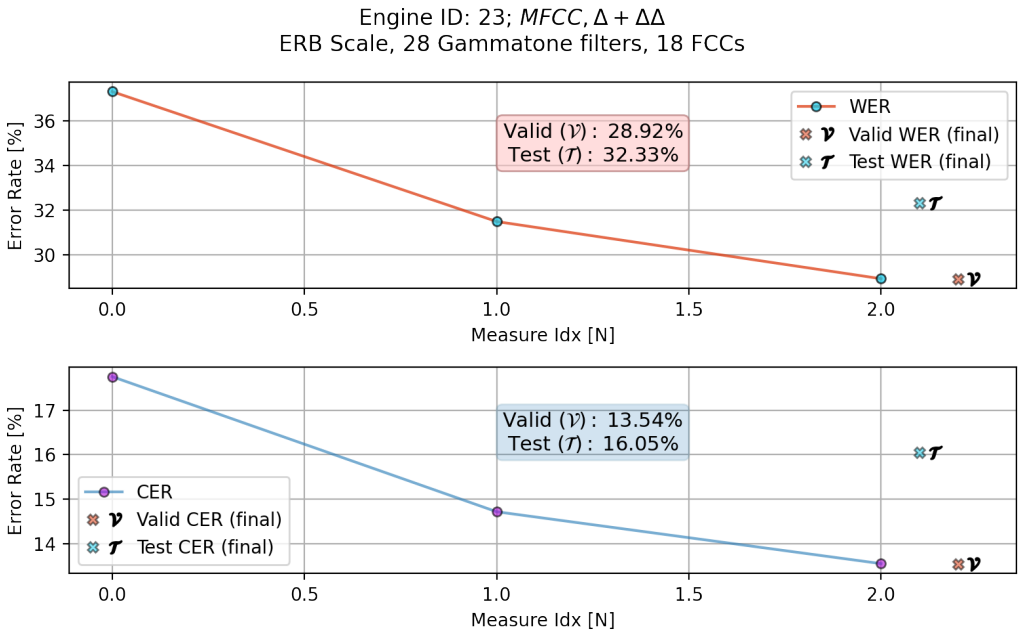


Figure 7.20: ASR #8 WER, CER evaluation plot

Figures 7.21, 7.22 present the training results and the WER and CER metrics measures for ASR Engine #5.

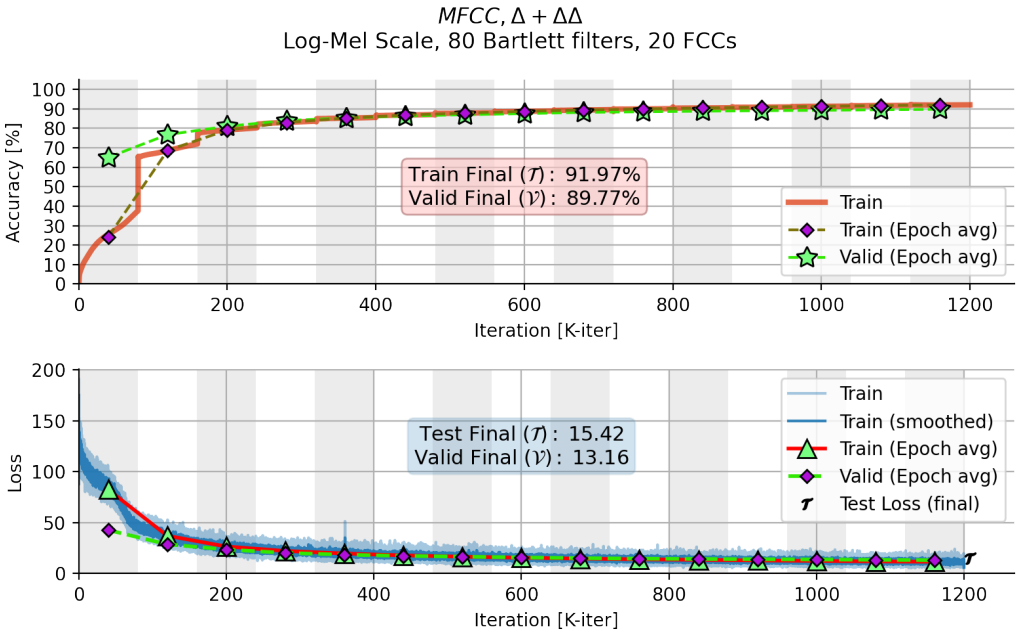


Figure 7.21: ASR #5 training accuracy and loss plot

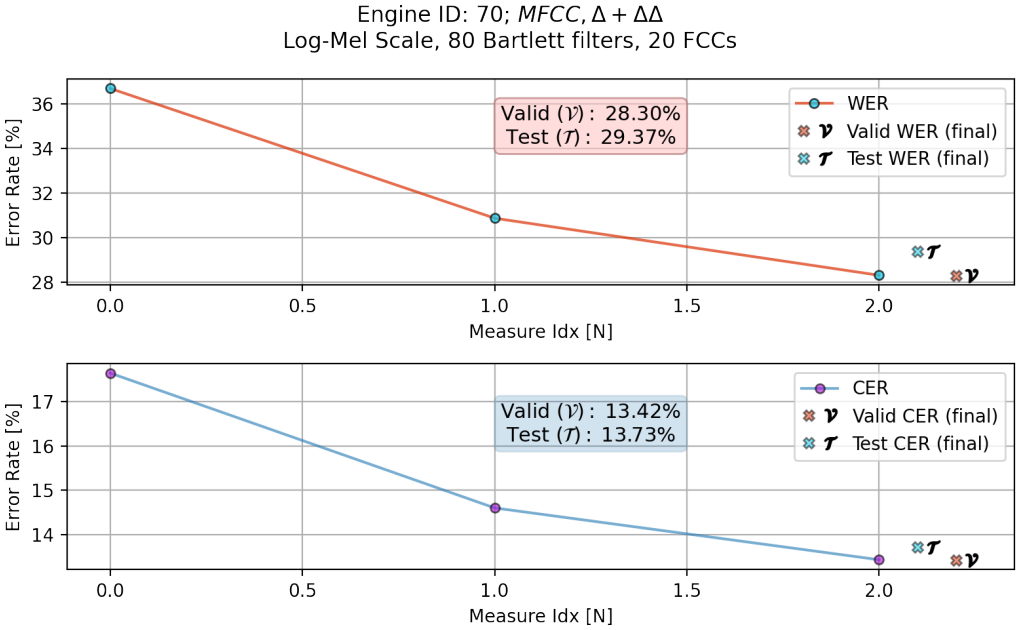


Figure 7.22: ASR #5 WER, CER evaluation plot

The training statistics were not gathered for ASR Engine #4, so only the

WER and CER measures for the validation and test datasets are presented in Figure 7.23.

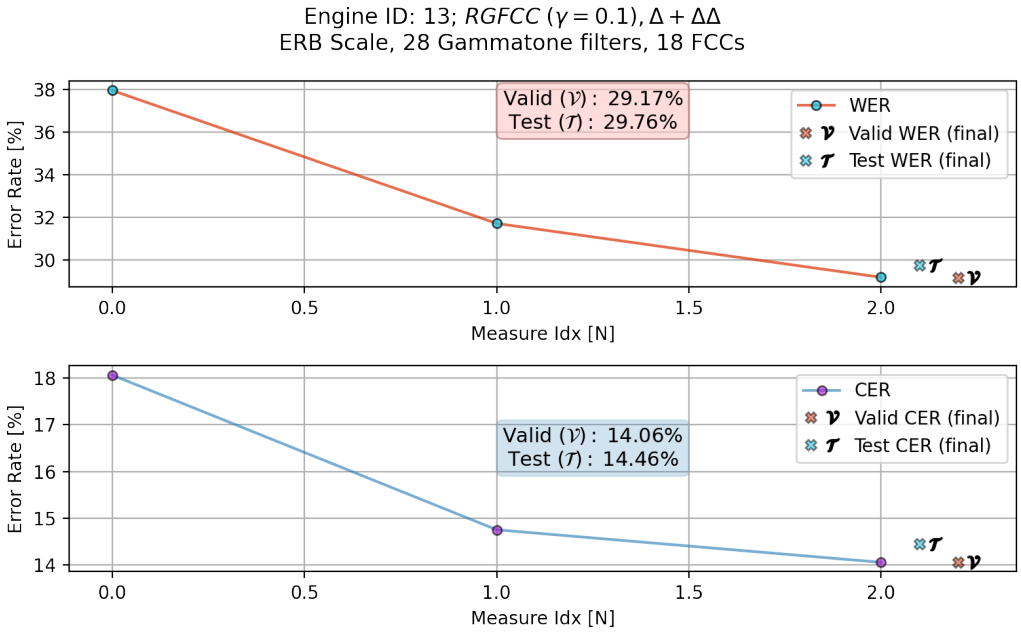


Figure 7.23: ASR #4 WER, CER evaluation plot

7.3 Chapter Summary

In this chapter, we covered the great advantage the E2E ASR system has over the traditional ASR system. Later, we reviewed different algorithms and techniques like the CTC and Seq2Seq that improve ASR system's performance. Following that, we presented the selected architecture for the ASR Engine, which includes CNN networks at the front-end followed by an attention-based encoder-decoder transformer that uses the above-mentioned algorithms.

During experiments, our focus was on the ASR performance according to WER and CER metrics. When training a model using a large dataset, we

observed that the loss function decreases over time. Thus, the engine's model reaches better detection rates and higher accuracy.

In our experiments, we tested multiple engine configurations. However, we consider only ten selected engines as meaningful. These ten engines are described in Table 7.1. The baseline architecture is the same for all the tested engines. The differences between the engines are the selected features for analysis, the scaling method, and mainly the technique of cepstral coefficient extraction.

Using various configurations, we covered a vast spectrum of parameters, indicating each parameter's impact on performance. In doing so, from the measured results, we could deduce that the Mel and Bark scaling methods yielded better results than the ERB scale. Moreover, we discovered that simplifying the cepstral coefficients calculations was made possible by reducing the number of filters or coefficients per T-F bin. A distinct example of this conclusion is the comparison between ASR Engines #5 and #6.

We also examined the change in performance due to root frequency cepstral coefficients. The RFCCs were replaced with the conventional cepstral coefficient extractions, which usually involve complex trigonometric functions such as inverse tangents and logarithms. The RFCCs results matched the conventional methods, and computation-wise, root-square is more efficient than logarithms. However, after introducing the simplifications to the traditional methods, such as the Mel and Bark approximations, we see no advantage in using RFCCs.

In the last trials, we tried to optimize the engine in terms of computation

time, memory footprints, hardware resources, and the derived power consumptions. Our most prominent finding is deduced from comparing ASR Engines #9 and #10 with ASR Engine #6. We simplified the ASR Engine's model by reducing the number of filters and coefficients. We also changed the bit precision for the feature extractions section and slimmed down the model's CNN and transformer layers.

Although detecting degradation in ASR performance has been seen due to these optimizations, we managed to compensate for it by adding an extra feature. With the added temporal context feature, we matched the WER and CER detection ratios we had before the optimization.

In conclusion, since the Mel and Bark approximations show significant improvement in execution time, hardware resources utilization, and therefore less power consumption, we chose them in favor of the logarithm-based calculations. Combining that with the previously mentioned optimizations, we ended with a more capable model. The received model produces the same results in less execution time and consumes less power. In addition, synthesis results point to lower resource utilization ratios, enabling the capability of fitting the model into smaller, cheaper devices. Overall, we successfully reduced the model's size by approximately half, from 146.2 million learnable parameters to 78.8 million.

Figure 7.24 summarizes the results of our ASR experiments comparing the different T-F masking techniques.

Id	Metric	Real (dt, et)					Parameters
		Noisy	IRM	ORM	PSM	Clean	
#1(5)	**WER**	46.30/59.69	27.56/35.55	24.14/34.03	**24.02/35.46**	21.03/25.90	146.2M
	CER	27.23/38.23	13.39/18.91	**10.66/17.94**	11.22/17.40	9.46/12.56	
#2(5)	**WER**	45.75/58.88	27.75/35.93	**25.35/33.42**	25.92/34.47	21.01/25.52	146.2M
	CER	26.20/37.32	13.14/18.59	**10.77/16.54**	11.87/16.39	8.94/11.70	
#3(12)	**WER**	45.16/59.32	27.66/35.58	**26.18/31.83**	26.76/37.61	21.06/26.21	146.2M
	CER	26.18/37.91	12.95/18.50	12.12/15.27	**11.91/18.32**	8.83/12.20	
#6(72)	**WER**	45.05/58.14	26.87/34.69	**24.82/31.36**	26.28/34.26	21.17/26.07	146.0M
	CER	25.78/36.73	12.42/18.01	**10.96/17.18**	11.27/17.79	8.70/12.19	
#9(100)	**WER**	45.60/58.25	27.48/34.58	23.88/30.16	**23.73/33.59**	21.86/26.51	**78.8M**
	CER	25.91/36.05	12.41/17.68	12.38/15.01	**10.33/17.71**	8.94/11.70	
#10(101)	**WER**	44.77/57.82	26.90/34.53	**23.16/31.45**	25.27/37.50	21.62/26.58	**78.8M**
	CER	25.32/35.33	12.22/17.16	11.38/14.95	**11.20/17.37**	8.77/11.71	

Figure 7.24: ASR CER WER Summary vs. T-F Masking

7.4 References

[1] Robert Blust. "Terror from the Sky: Unconventional Linguistic Clues to the Negrito Past". In: *Human biology*. 2013 (cit. on p. 108).

[2] William Chan et al. "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition". In: *ICASSP*. 2016. URL: http://williamchan.ca/papers/wchan-icassp-2016.pdf (cit. on pp. 111, 114).

[3] R. M. W. Dixon. *The Rise and Fall of Languages*. Cambridge University Press, 1997 (cit. on p. 108).

[4] Alex Graves and Navdeep Jaitly. "Towards End-To-End Speech Recognition with Recurrent Neural Networks". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 2014, pp. 1764–1772. URL: https://proceedings.mlr.press/v32/graves14.html (cit. on p. 109).

[5] Andrew Maas et al. "Lexicon-Free Conversational Speech Recognition with Neural Networks". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, 2015, pp. 345–354. DOI: 10.3115/v1/N15-1038. URL: https://aclanthology.org/N15-1038 (cit. on p. 110).

[6] Ruhi Sarikaya and John Hansen. "Analysis of the root-cepstrum for acoustic modeling and fast decoding in speech recognition." In: 2001, pp. 687–690 (cit. on p. 119).

[7] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL] (cit. on pp. 111, 113).

Chapter 8

Datasets

8.1 CHiME 4/5

8.1.1 Overview

The CHiME-4 dataset is provided as part of the CHiME challenge [1, 2].

The challenge is to build an ASR model for multi-microphone devices being used in various noisy environments.

Six microphones are mounted on a tablet device and are recorded with a high-resolution multi-track recorder. Besides those six microphones, an additional close-talking microphone is recorded by another recording device, daisy-chained to the multi-microphone recording device.

The close-talking microphone is referenced as the clean speech channel.

8.1.1.1 CHiME Recording Scenarios

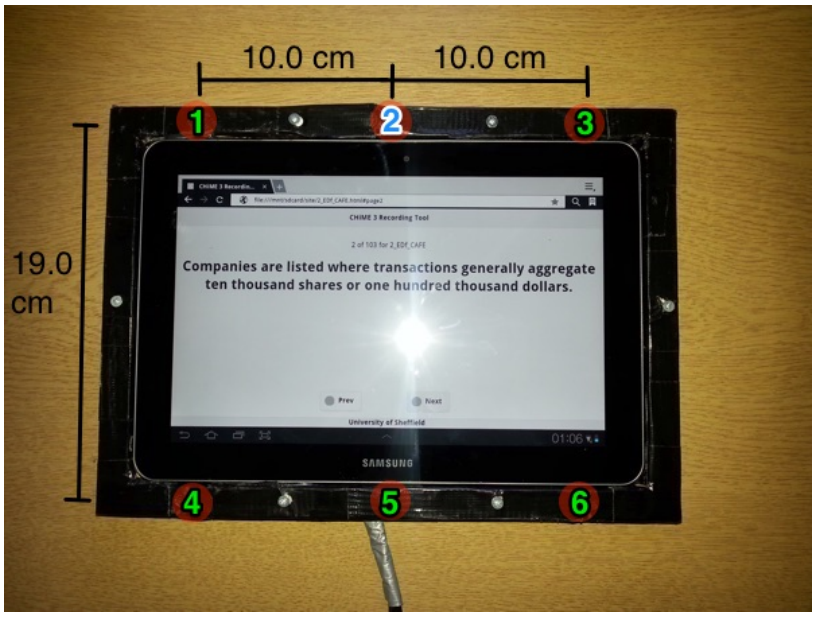


Figure 8.1: CHiME-4 microphone-array

Source: Adapted from [4]

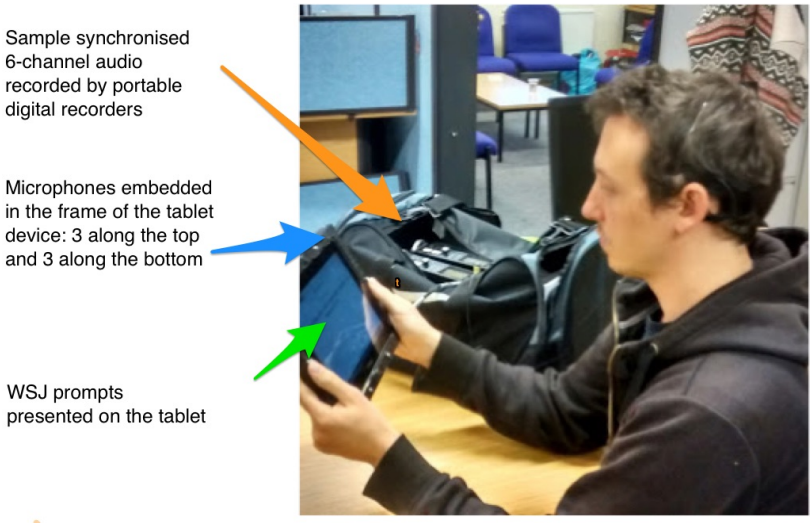


Figure 8.2: CHiME-4 recording setup

Source: Adapted from [4]

Four different noisy environments were selected, a Cafe, a street junction, a bus, and a pedestrian area.

The CHiME-4 dataset stats are shown in Figure 8.3

	Real/Simu	Location	Channels	# speakers	# utterances	hour	# of WAV files
dt05_bth		BTH	0-6	4	410	0.72	2870
dt05_bus_real	REAL	BUS	0-6	4	410	0.68	2870
dt05_bus_simu	SIMU	BUS	1-6	4	410	0.72	2460
dt05_caf_real	REAL	CAF	0-6	4	410	0.69	2870
dt05_caf_simu	SIMU	CAF	1-6	4	410	0.72	2460
dt05_ped_real	REAL	PED	0-6	4	410	0.67	2870
dt05_ped_simu	SIMU	PED	1-6	4	410	0.72	2460
dt05_str_real	REAL	STR	0-6	4	410	0.7	2870
dt05_str_simu	SIMU	STR	1-6	4	410	0.72	2460
tr05_bth		BTH	0-6	4	399	0.75	2793
tr05_bus_real	REAL	BUS	0-6	4	400	0.69	2800
tr05_bus_simu	SIMU	BUS	1-6	83	1728	3.71	10368
tr05_caf_real	REAL	CAF	0-6	4	400	0.76	2800
tr05_caf_simu	SIMU	CAF	1-6	83	1794	3.77	10764
tr05_org			single	83	7138	15.15	7138
tr05_ped_real	REAL	PED	0-6	4	400	0.72	2800
tr05_ped_simu	SIMU	PED	1-6	83	1765	3.75	10590
tr05_str_real	REAL	STR	0-6	4	400	0.73	2800
tr05_str_simu	SIMU	STR	1-6	83	1851	3.92	11106

Figure 8.3: CHiME-4 recording setup

Source: Adapted from [4]

8.2 CommonVoice

The CommonVoice dataset is an open-source, free dataset provided by Mozilla [3].

CommonVoice for the English corpus contains overall 2886 hours of recordings, where 2185 hours of them were validated by the community, and there are 79,398 different voices in the dataset.

Statistics and distributions for the CommonVoice dataset are shown in Table 8.1.

Accent	23% United States English 8% England English 7% India and South Asia 3% Canadian English 3% Australian English 2% Scottish English 1% New-Zealand English 1% Southern African 1% Irish English 51% Undeclared
Age	24% 19-29 13% 30-39 10% 40-49 6% <19 4% 60-69 4% 50-59 1% 70-79 38% Unknown
Gender	45% Male 15% Female 40% Unknown

Table 8.1: CommonVoice dataset statistics

During training, we dropped recordings longer than ten seconds. This is because the dataset contains open-mic recordings, which cause the feature vector to be so large that it cannot fit in either the GPU's dedicated RAM or the server's memory. As a result of omitting recordings longer than the ten

seconds threshold, some recordings, although being valid, are still marked as open-mic and also dropped from the dataset.

Another issue we faced with the CommonVoice dataset is that some recordings were not validated and thus could be corrupted. Filtering out those recordings is very cumbersome and would take long a time to complete. Therefore, the number of non-validated recordings participating in the training, validation, or testing phases was cut in half. We have seen some performance degradation in terms of WER and CER due to non-validated recordings in the test and validation measurements. However, these faulty measures are sparse compared to the validated recordings measures, introducing only a slight impact on the mean value, but can be seen in the variance fluctuations presented in the bar plots in Chapter 7.

Table 8.2 summarizes the CommonVoice dataset subsets and the number of recordings each subset contains.

Set	Utterances [N]
Training	759,546
Dev/Valid	16,264
Test	16,236

Table 8.2: CommonVoice sets utterances distribution

8.3 References

[1] Szu-Jui Chen et al. "Building state-of-the-art distant speech recognition using the CHiME-4 challenge with a setup of speech enhancement baseline". In: *CoRR* abs/1803.10109 (2018). arXiv: 1803.10109. URL: http://arxiv.org/abs/1803.10109 (cit. on p. 140).

[2] Szu-Jui Chen et al. *The 4th CHiME Speech Separation and Recognition Challenge*. 2016. URL: http://spandh.dcs.shef.ac.uk/chime_challenge/CHiME4/overview.html (cit. on p. 140).

[3] Mozilla Corporation. *Mozilla CommonVoice Dataset*. 2022. URL: https://commonvoice.mozilla.org/en (cit. on p. 142).

[4] *Zynq UltraScale+ MPSoC Product Tables and Product Selection Guide*. Tech. rep. University of Sheffield, 2016. URL: https://spandh.dcs.shef.ac.uk/chime_challenge/CHiME4/overview.html (cit. on pp. 141, 142).

Chapter 9

Conclusions

In this research we have investigated some possible explicitly or implicitly direct links between several metrics that are measured in different stages along an E2E ASR system.

We also investigated the impact of speech enhancement where we have seen how a multi-channel microphone array beamformer combined with a T-F masking algorithm can improve an existing E2E ASR system detection rates. This provided insights into a correlation between improvements in SNR, PESQ, and STOI leading to an improvement in WER and CER whom are ASR metrics.

We also presented the magnitude of influence that different scaling methods have on ASR system's performance and requirements, especially when the implementation is targeted for a hardware device rather than a software based solution. In such cases, the requirements for computation efficiency, low power consumption and wise utilization of limited resources are taken into account and affect the ranks of different configuration sets. For example, slim models for the beamformer, masking or the ASR engine would be ranked

higher than heavy and larger models when the target device has limited memory resources. Alternatively, when storage memory is not considered an issue, a better performing algorithm may be achieved using additional memory. Such an algorithm would yield higher detection rates.

To evaluate the End-to-End speech enhancement plus the ASR engine performance, we divided the effort into two sections, where each has been evaluated separately. Later, the two sections were connected together and the evaluation was taken from input to the ASR output.

During experiments, multiple ASR engines were trained against different carefully selected feature sets, each time alternating one parameter, whether it was the features' combination, scaling method, the computation for feature extraction or the neural network's shape and structure. A clear advantage, in terms of ASR performance (WER, CER), has been observed for a combination of the Delta, Deltas-Delta $(\Delta, \Delta\Delta)$ and the companion cepstral coefficients, over the increase in the number of filters and coefficients or the usage of only FilterBanks. Adding temporal context to the mixture, resulted in a slightly better ASR performance. These insights reflected mainly in the effort of simplifying our ASR model and reducing its memory footprints. In accordance, simplifications such as approximations for the scaling methods and the reduction in filters and coefficients were made possible while maintaining equal detection rates and sometimes even better than highly dense FilterBank based ASR systems. On the other hand, root cepstral coefficients extraction did not yield any gain in performance over the natural or log based coefficients, nor the suggested approximations.

Those trained ASR engines were attached to a processing FE (Front-End), that serves as a speech enhancing stage. The speech enhancing is applied by multi-channel multi-microphones array deep beamformers combined with T-F masking. That combination between a GEV beamformer and a T-F masking extraction layer attached to an ASR Engine led to much improved performance compared to the ASR's performance measured as a standalone unit. Deeper analysis of the various T-F masking alogirthms yielded a conclusion that the audio domain metrics such as SNR and STOI have direct impact on the ability of the ASR engine to be more precise. This actually strengthen the intuition that better speech quality or cleaner speech is easier to perceive, and thus, ASR transcripts would be more accurate for it.

Some very important questions arise with regard to adding a FE stage to the ASR engine that is, what would be the cost? What could be traded in favor of performance or higher detection ratios? In this research, we used the same configuration for the feature extractions in both the FE and the ASR engine itself. Doing so, we managed to combine the feature extraction mechanism and save on resources by reusing the same modules. The computation overload for the FE part becomes negligible since it is already implemented for the ASR; Thus, only the extraction of the beamformer coefficients is left for computation. Despite saving on resources, an increase in memory is anticipated to hold the beamformer's and the T-F masking NN model parameters.

In this research we present trade-off options, together with the expected gains in some metrics. The gains or deteriorations in the different metrics are presented side by side to the speech WER and CER metrics which gives a more

comprehensive perspective about the overall performance, from end-to-end. For systems that are not designed with limited resources, like memory and slow operating clock frequency, the more advanced, but complex algorithms for T-F masking as cIRM, PSM, and ORM can be used. Leading to higher perception rates at the ASR engine's output as mentioned before due to higher SNR levels. However, a trade-off can be made between complex T-F masking algorithm and slimmer ASR engines and vice versa. For example, picking a more optimized version of an ASR engine, while utilizing a PSM T-F masking algorithm can yield matched results in terms of WER and CER. According to the results presented in this research, speech-to-text systems can be optimized by wise selection of configurations for certain modules and still achieve the desired quality of performance.

9.1 Future work

In this research we based our ASR engine architecture on the transformer model. Other advanced ASR engine models are available such as CTC based Seq2Seq and the transducer model. Also, modern techniques of introducing additional acoustic information to the ASR training process like the wav2vec outputs are expected to improve the results presented in this research.

Besides the ASR, further optimizations in hardware can be applied with dedicated HW based accelerators, which are now built-in inside new AI capable FPGA devices. These hardware accelerators are tailor made for neural

networks, mainly CNNs. The systolic-array architecture that is very common for such accelerators provide much lesser computing times when compared to conventional piped-line hardware architectures. Furthermore, the computing efficiency measured in tera-operations per energy or power units $[TFLOPS/Joule, TFLOPS/Watt]$ of such accelerators is tens to hundreds times higher.

Going forward, combining both the alternative ASR engine models and more expanded optimizations in hardware, this research can be more comprehensive in terms of the possibilities laid in front of a system designer for taking trade-offs based on the system's requirements.