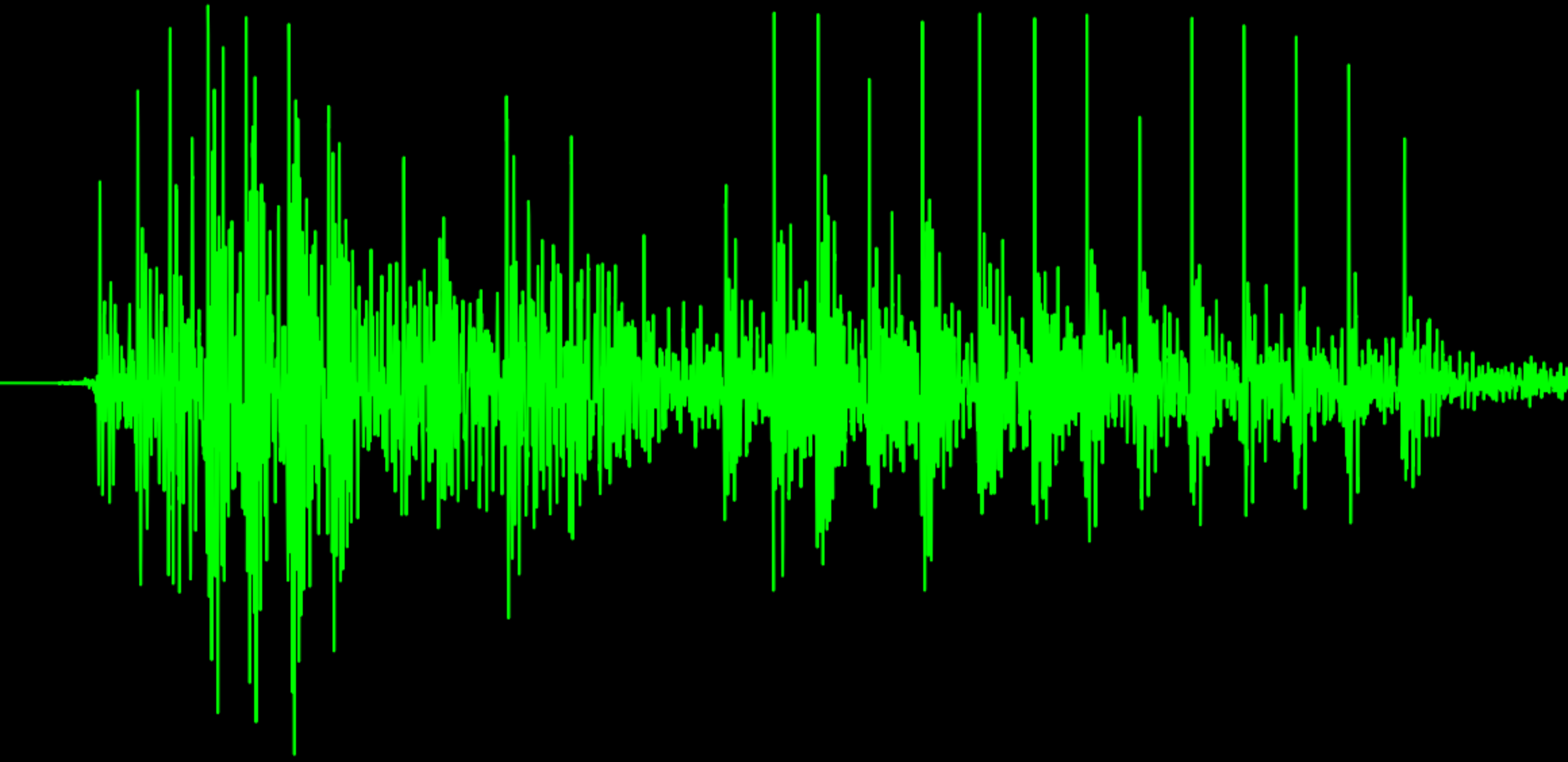
Master Thesis
A Generative Neural Network Model for Speech Enhancement

Husain S. Kapadia

Technische Universiteit Delft



Delft University of Technology

Challenge the future

Master Thesis

A Generative Neural Network Model for Speech Enhancement

by

Husain S. Kapadia

conducted at

GN Hearing A/S, R&D Department, Eindhoven
&
Circuits and Systems Group (CAS), TU Delft

pursuing

Master of Science
in Electrical Engineering - Signals & Systems

at the Delft University of Technology,

Student number:	4707915	
Project duration:	December 1, 2018 – September 20, 2019	
Supervisors:	Prof. dr. W.B. Kleijn,	TU Delft
	dr. ir. Anne Hendrikse,	ReSound GN
Thesis committee:	Prof. dr. W.B. Kleijn,	TU Delft
	Prof. dr. Bert de Vries,	TU Eindhoven
	dr. ir. R.C. Hendriks,	TU Delft

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"A Generative Neural Network Model for Speech Enhancement"** by **Husain S. Kapadia** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 20 September, 2019

Chairman: ______________________
Prof. dr. W.B. Kleijn

Advisors: ______________________
Prof. dr. W.B. Kleijn

Committee Members: ______________________
Prof. dr. Bert de Vries

____
dr. ir. R.C. Hendriks

Abstract

Listening in noise is a challenging problem that affects the hearing capability of not only normal hearing but especially hearing impaired people. Since the last four decades, enhancing the quality and intelligibility of noise corrupted speech by reducing the effect of noise has been addressed using statistical signal processing techniques as well as neural networks. However, the fundamental idea behind implementing these methods is the same, i.e., to achieve the best possible estimate of a single target speech waveform. This thesis explores a different route using generative modeling with deep neural networks where speech is artificially generated by conditioning the model on previously predicted samples and features extracted from noisy speech. The proposed system consists of the U-Net model for enhancing the noisy features and the WaveRNN synthesizer (originally proposed for text-to-speech synthesis) re-designed for synthesizing clean sounding speech from noisy features. Subjective results indicate that speech generated by the proposed system is preferred over listening to noisy speech however, the improvement in intelligibility is not significant.

Acknowledgements

Foremost, I would like to express my deep gratitude to Prof. dr. Bastiaan Kleijn , my direct supervisor and mentor from TU Delft for his continuous guidance, encouragement and useful critiques during my M.Sc. thesis study. His supervision helped me develop a new perspective for tackling problems and also enriched my research skills. In spite of his busy schedule, he provided me with feedback every week through our video sessions and was available for discussions and review on my progress. He provided me with sufficient creative freedom to go explore on my own while at the same time he steered me back on track when I wandered off too far by offering me reality checks at regular intervals of time. It was an absolute pleasure to work under him and a privilege to learn from him.

My research would have been impossible without the support of GN Hearing. I am profoundly grateful to Anne to provide me the opportunity to do my internship followed by a Master thesis with GN Hearing. He was not only my company supervisor during the course of my thesis, but also my source of motivation and inspiration. Anne was there for me whenever I was in need of dire help. His infinite patience and crucial feedback helped me during my experimentation. Moreover, his unconditional faith in my abilities made this experience even more delightful. I thank him for being a part of this journey and walking all the miles with me till the very end. My sincere thanks to Almer van den Berg, Anuj Dev and Erik van der Werf for their constant support, advice and insightful comments.

I acknowledge the contribution of all my peers to appear for the subjective listening test. At this juncture, I would also like to extend my gratitude to Prof. dr. Bert de Vries and dr.ir. Richard Hendriks for taking time off their busy schedules to be a part of my thesis committee.

I am very thankful to Mihir, Nilay, Neel and Sukanya who started this journey at TU Delft with me from India. They were not only present with me during my moments of joy but also at my hardest times. Their encouragement boosted my self-confidence and reassured me of my abilities. I would also like to thank Metin, Jakub and Sebastien for being such amazing project partners, friends and travel buddies. My friends from Signals & Systems, Bishwadeep, Shreyas, Pranav, Sherine, Rajesh, Karishma, Bram, Manolis and Gabrielle have also been a source of inspiration and I have thoroughly enjoyed their company. I am glad to have met and spent my time with all these people during the course of my Masters programme at TU Delft.

I feel very fortunate to have Sakina in my life as she has been a great source of help and motivation. She makes me feel like home even when we are far away from it. Finally, I would like to thank my parents for their unconditional love and concern throughout my life. This work would not have been possible without their constant encouragement and support. I dedicate my work to my parents and Sakina.

Husain S. Kapadia
Delft, September 2019

List of Figures

List of Tables

Contents

1
Introduction

The problem of enhancing speech that is degraded by uncorrelated additive noise has received much attention [1, 2]. In the earlier years $(1980 - 2015)$, this problem was mainly viewed as a statistical signal processing problem [3–6]. The classical speech enhancement approaches analyze the speech signals in the frequency domain, apply suitable gains by designing filters to suppress bands where noise is predominant and then synthesize the spectrum back to a waveform. The calculation of appropriate filter gains was dependent on the noise and speech power spectral density estimates [2]. For stationary noise, this gain calculation was relatively straightforward, but in case of non-stationary noise, statistical estimation techniques were used to actively track the varying noise power [5–7]. Relatively fewer attempts have been made by approaches based on machine learning or deep learning but this number is increasing drastically since 2013 [8–13].

Recently, there has been a boost in using machine and deep learning based methods for solving problems in several domains of science and engineering. Among the reasons why, are that deep Neural Networks can take advantage of large training sets and infer structure autonomously whereas most heuristic and statistical approaches require much more upfront definition by engineers to achieve high performance. Deep learning approaches have achieved state of the art results for various problems in the field of computer vision, language processing and speech recognition [14–16]. In recent years, classical signal processing strategies for the task of speech enhancement have been outflanked by machine learning and deep learning approaches [8, 10, 12]. Most methods introduced, so far, rely on regression based techniques that attempt to map to the waveform or spectral features of speech in the clean dataset from the noisy one [8–10, 13]. However, it is believed that there are limitations in their performance in terms of speech quality due to distortions and artifacts. This aspect is discussed in more detail in section 1.1.

Since the last five years, research in the field of deep learning is tending towards Generative Neural Network models [14, 17–20]. Certain applications, such as, Text to Speech conversion have been said to achieve their pinnacle after the recent introduction of generative models called WaveNet [19] and WaveRNN [21]. The advent of Generative Adversarial Networks (GANs) by Goodfellow *et al.* in 2014 [18] has provided a new perspective of looking at problems of style transfer in images, caption generation for images, video and music generation. The aim of this thesis is to acquire inspiration from existing generative modeling techniques and develop a new generative model for enhancing the quality and intelligibility of noise corrupted speech and also evaluate its performance. The motivation and goals behind employing a generative model for a speech enhancement task will be stated more clearly in 1.2.

In hearing aid devices, resources such as computational power, memory, etc. are restricted by orders of magnitude as compared to the processing units on high-end servers where Deep learning models are actually implemented. Therefore, applying neural networks on a hearing aid device is not straightforward and requires a strong understanding of how the benefits of Neural Networks ensue. This thesis attempts at designing neural network with compact architectures that consume less computational and memory resources. The proposed architecture is not yet ready to be deployed on a hearing aid device but has the capability to be easily deployed on a mobile phone CPU. Now-a-days, a mobile phone is also a part of the hearing aid system, and it operates at a higher power, can perform faster computations and has more memory than a hearing aid device. A comparative analysis on the computation and memory requirements for the proposed network will be conducted, and followed by a discussion on its applicability in current hearing aid systems.

This chapter consists of three sections; first, the gap in research, contributions of this thesis and motivation behind this study are described in section 1.1. This is followed by section 1.2 where the goals of this research are stated explicitly. Thereafter, an overview of the entire thesis is provided in section 1.3, which briefly explains the contents within the subsequent chapters.

1.1. Research Gap & Contributions

Most solutions developed for the problem of speech enhancement have been implemented by analyzing the speech signal in the frequency domain [1, 2, 8, 9]. Traditional signal processing techniques are analytically derived such that the system should produce speech that is the best estimate of the reference in terms of the minimum mean square error [2]. This leads to distortions and artifacts present in the output speech which shows room for further improvement in quality. The shift towards machine/deep learning approaches have lead to certain improvements in results in terms of quality and intelligibility [12, 13, 22]. The advantages of these models over classical methods is that instead of being derived from a set of assumptions, they learn characteristics from data to adapt themselves such that they can generalize to unseen scenarios. Additionally, they apply non-linear transformations which lead to an effective mapping. However, most deep learning methods developed for the task of speech enhancement also minimize either the least squares metric or the mean absolute distance between the estimated and reference spectral features or waveforms [8–13, 22–24].

The aim of majority of the research done so far in the field of speech enhancement has been to either reproduce the target waveform or reproduce target features and then synthesize them to a waveform. This ideology is revealed by the definition of their objective, i.e., minimizing the distance between the system output and the target. This thesis looks at the problem from a different perspective. If one visualizes the speech waveforms or spectra of multiple recordings of the same utterance, it is observed that these waveforms or spectra are not the same. This can be seen in figure 1.1 where four different recordings of the word ”Hello” spoken by the same talker have been displayed. This shows that a particular waveform is not unique to the information conveyed by speech. Hence, trying to map to one particular reference or target seems to limit the performance of such systems and causes output speech to be distorted.

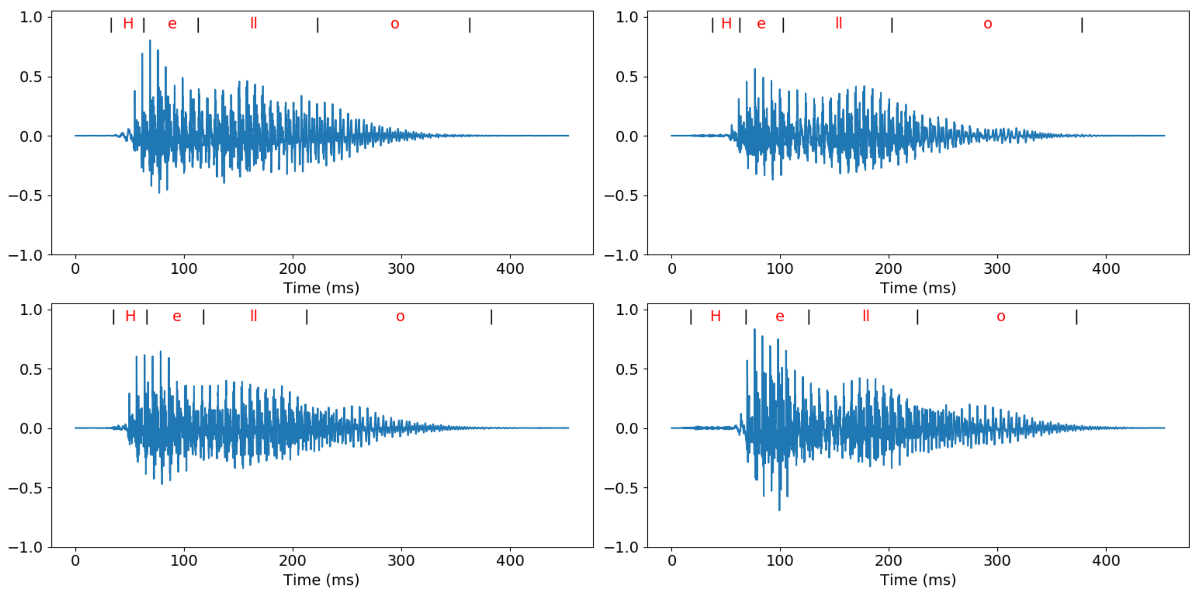


Figure 1.1: Multiple recordings of the word ”Hello” spoken by the same talker

The system proposed in this thesis attempts at artificially generating speech by modeling it as a conditional probability distribution dependent on previous speech samples and the context provided by features extracted from noisy speech. This can be achieved with the help of generative models for speech synthesis such as WaveNet [19] or WaveRNN [21] that have been originally designed for the task of speech synthesis from text. Literature shows an increasing trend towards using generative models for speech enhancement but these methods also partially rely on minimizing the distance metrics between the speech waveform or features [12, 22]. Hence, there does not exist a completely generative model for speech enhancement. The contribution of this thesis is to explore the domain of generative modeling of speech and propose a system that can synthesize clean sounding and understandable speech from context provided by noisy features.

1.2. Research Goals

After highlighting the gap in the research of speech enhancement and proposing a novel route for solving the problem in section 1.1, this section specifies the goals of this research. This thesis report aims to address the following research questions.

How can clean sounding and intelligible speech be generated when the context provided to a generative model is noisy?

What are the functionalities of the key components used in the proposed generative model?

1.3. Overview of the Thesis

This section briefly summarizes the contents within the chapters in this report. The aim of this section is to guide the readers by providing a gist of the narrative and highlight important outcomes from every chapter upfront. The beginning of each chapter has an introduction explaining the topics covered in it. The chapters are organised as follows:

Literature Review

This chapter focuses more on existing solutions for the problem of speech enhancement and also speech generation or synthesis. Literature right from classical signal processing to deep neural networks are explored and critically evaluated. The transition from classical signal processing methods to machine learning approaches is explained. Generative methods outside the scope of speech synthesis have also been studied in order to gain inspiration with the aim to apply them for this research. After weighing the advantages and disadvantages of various generative neural network modeling techniques, the WaveRNN synthesizer is chosen for the desired task.

Feature Engineering

The definition and derivation of features obtained from speech such as spectrograms, Mel spectrograms, Mel frequency cepstra and Spectral Delta are discussed in this chapter. A non-conventional method for normalizing the data using the sigmoid function is proposed. An analysis that compares different noise contaminated features with their clean ones indicate the effect of noise on speech information. Since, the WaveRNN model needs context that delivers appropriate information about speech, this chapter also aims to find the best possible feature representation.

The WaveRNN Synthesizer

This chapter provides a detailed description of the design, training and audio generation procedures for the WaveRNN architecture. Experiments conducted and results obtained are discussed. An analysis of the results, network performance and its components reveal interesting findings about the functionality of the system. It is concluded that synthesizing audio from a WaveRNN model trained directly with noisy features as context does not yield understandable speech as compared to that when trained with clean features. Hence, the noisy features must be enhanced before audio synthesis.

Denoising Features

Based on the conclusions from chapter 4, different denoising networks are proposed in this chapter, namely, the U-Net and the Variance constrained Auto Encoder (VCAE) architectures. Their design, theoretical interpretation, training and inference procedures are described. Furthermore, it contains a discussion on results obtained from training these networks for different configurations. Despite conducting a comparative analysis on the denoised features obtained from both types of architectures and the network performance, it was difficult to finalize on a particular network at this stage.

Fine-tuning the Synthesizer

The principles of transfer learning are highlighted and applied in this chapter to fine-tune a WaveRNN synthesizer with denoised features that was initially trained on clean features. Denoised features from both the U-Net and VCAE networks were utilized for fine-tuning the synthesizer. With the proposed configuration and settings for both feature denoising architectures, results revealed that the combination of the WaveRNN synthesizer with the U-Net architecture yields better quality speech than that with the VCAE network. Finally, the generated speech samples are also compared with existing state-of-the-art speech enhancement models.

Conclusion & Discussion

The final chapter encompasses a discussion section about the advantages and drawbacks of the proposed method for speech enhancement. It also gives an insight into the applicability of this method in existing hearing aid systems. In the end, some remarks on the future extensions of this project are made.

Bibliography & Appendix

The bibliography lists all sources referred to while pursuing this research. There are in total four appendices where appendix A describes fundamental signal processing and machine/deep learning concepts relevant to the subject of the thesis. The topics mentioned in this chapter are discussed from a theoretical perspective. It provides information about basic functioning of different neural network modules which can then be extrapolated to understand the proposed system. Appendix B highlights some extra experiments (besides the ones mentioned in the main text) that contain valuable details. Appendix C mentions the configurations for all the experiments mentioned in the main narrative as well as the ones in appendix B. The last appendix contains a table of specifications for the hardware devices used for experimentation, a mobile phone CPU and a hearing aid processor.

Note

Experimentation performed and analysis conducted were implemented in the 'Python' programming language. The 'PyTorch' deep learning framework developed by Faceboook was used for implementation of the neural networks [25]. The audio files for examples discussed in the experiment sections of chapters 4, 5 and 6 are uploaded on this link. The link contains directories and sub-directories for the chapters and their experiments respectively. Apart from the examples used to discuss the results, other examples are also shared. The readers are advised to listen to the audio files which can help evaluate and understand the results better.

2
Literature Study

This chapter delineates the journey of the development that has occurred in the field of Speech enhancement right from the classical methods to the current trend in deep learning approaches. Some examples also portray the co-existence of both approaches in finding solutions to related problems. Research that has led to significant improvement in speech quality and intelligibility are mentioned and discussed. Certain drawbacks in these systems are also highlighted which led to the development of better systems. Certain distinctive characteristics from all possible methods are kept in mind so that it would be helpful while implementing the desired approach based on generative neural network models.

Recent study in deep learning focuses more on generative neural networks as compared to the regression based approaches. In contrast to the conventional estimation techniques that focus more on increasing signal to noise ratio, the philosophy of artificially generating clean speech from noisy speech or its features is the driving force behind pursuing this direction of research. State-of-the-art generative deep learning methodologies not only designed for speech enhancement but also applied in the realms of computer vision, text-to-speech conversion and speech recognition are investigated. The aim is to shed light on some striking features and unique ideas that have led to a progress. Inspiration is sought from literature that have introduced novel network architectures, training procedures and their interpretation. Exploring different domains help discover opportunities to apply those ideas for the problem of speech enhancement.

2.1. Classical Speech Enhancement Methods

This section highlights one of the classical speech enhancement systems shown in figure 2.1. Most modules from this block diagram still play an active role in telecommunications and hearing aid systems. Speech is analysed by dividing it into short-time segments (usually $20 - 40$ ms) to ensure wide sense stationarity and then transformed into the frequency domain. This helps in obtaining reliable spectral estimates. The processing pipeline is applied in the frequency domain and later the speech waveform is synthesized by using a short-time inverse Fourier transform along with an overlap and add module. The processing pipeline consists of 4 blocks that serve different purposes.

The 'Apply Gain' block applies suitable gains to certain frequency bands at different time segments that are calculated in the 'Target Estimation' block by methods such as spectral subtraction [1] or Wiener filtering [2] (descibed in Appendix A segment A.1.3). The function of this block is to suppress those frequency bands in which noise is predominant so as to increase the Signal to Noise ratio (SNR) for the enhanced output speech. Furthermore, this module faces a difficulty in generalizing to different types of background noises in practice [5]. To circumvent this effect, some additional modifications have to be added to the system such as 'Noise PSD Estimation' which employs statistical models for voice activity detection [4, 26]. Later, they were substituted by superior modules that could actively track non-stationary noise [5–7] and adapt to different types of background noises by minor parameter tweaking. A 'Target PSD Estimation' module is used for recursively estimating the signal to noise ratio [27]. The processing pipeline also relies on prior information either about the speech or noise signals

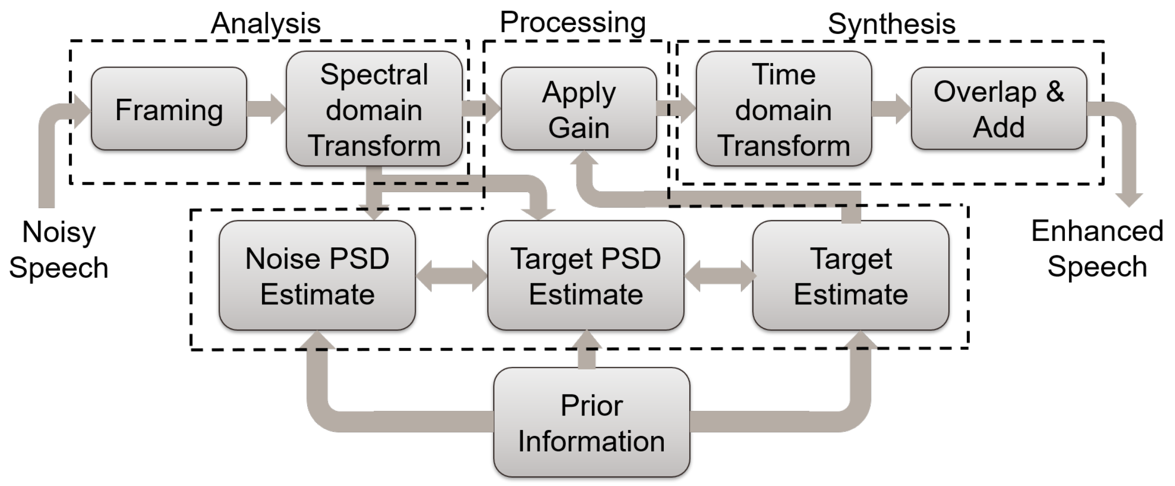


Figure 2.1: Block diagram of a Classical Speech Enhancement System

or the mixing process. This helps in further improving the system performance. Such techniques have led to a notable improvement but minor distortions still exist due to the tracking delay introduced by the modules. These systems could greatly improve the intelligibility but there is still room for significant enhancement in speech quality. In this section, a survey of some state-of-the-art signal processing approaches are presented.

2.1.1. Noise Power Estimation

Noise Power Spectral Density Estimation plays an essential role within a speech enhancement system because it allows the system to adapt its parameters according to the change in noise characteristics over time leading to better enhancement in speech quality. The first step towards solving this problem is to determine speech activity. Hence, a voice activity detection (VAD) module is used to differentiate between speech and non-speech activity in noisy speech conditions. Speech activity is further classified into voiced and unvoiced speech. Detecting unvoiced speech within noise is a challenging problem to solve because their energy levels normally lie within the same range. *Rabiner et al.* proposed thresholding short-time energy levels and zero crossing rate (ZCR) [28] in the time domain to solve this problem. This rudimentary approach yielded a binary mask which was then applied to noisy speech. Despite being sufficiently reliable in capturing some details from unvoiced speech, there exist unprecedented glitches and distortions especially in low SNR situations.

Statistical model based algorithms were introduced which made soft decisions over voice activity instead of hard binary rulings. This was further improved by *Sohn et al.* where the model parameters were estimated first by means of a decision-directed (DD) approach [4] before detecting voice in speech. This led to high detection accuracy as compared to the conventional algorithms. All statistical approaches so far assumed that speech spectra was characterized by Gaussian distributions but a study in 2003 revealed that they could be described more effectively with Gamma and Laplacian distributions [29]. As a result, the distributions defining the hypothesis functions for speech and noise changed which boosted the efficiency in detection [26]. One drawback in VAD based models was that it was difficult to detect sudden rise in noise power and was usually interpreted as speech onset especially for non-stationary noise scenarios [6]. Additionally, noise was not suppressed appropriately in segments where it co-existed with speech. Thus, some fragments in estimated speech sounded silent whereas some noise was still audible in the speech components. This led to an idea of estimating noise statistics in silence periods which can then be used to actively track and minimize its presence during speech activity.

Certain speech enhancement systems contain both VAD models to determine speech activity as well as models to track the noise power spectral density during no speech activity so as to effectively suppress the effect of noise during speech activity [30]. Although, there exists some methods that completely replace the VAD process and directly perform noise tracking [5–7]. R. Martin proposed to do so by tracking the spectral minima in each frequency band irrespective of speech activity or pause [5]. The algorithm recursively smooths out the power spectrum with an optimal smoothing parameter obtained by minimizing a conditional mean square error. In order to track the minima of the periodogram, a collection of large number of frames is required to ensure that speech activity is less. This introduces large tracking delay in the order of $1-2$ seconds resulting in an underestimation of the noise variance [31]. Thus, a bias compensation is applied which gives approximate solutions. The performance is accurate for stationary and slowly varying noise but in case of a sudden rise in noise level it fails to catch up due to the large delay resulting in residual noise.

A different viewpoint for solving this problem was provided by *Hendriks et al.* by using a Bayesian minimum mean square estimation [7] of the noise power given the noisy speech spectrum. Their method significantly reduced the tracking delay by applying an analytically derived bias compensation that can effectively counter noise under-estimation. In cases where the noise level changes abruptly, the recursive parameter updates may stop and lead to stagnation which in turn stops tracking. Hence, it also employs a safety-net by storing past noisy periodogram frames up to 0.8 seconds to overcome this stagnation. This makes the system memory inefficient. Moreover, it cannot compensate well in case of an over-estimation. Later, they proposed an unbiased estimator based on speech presence probability that neglects the bias compensation and the safety net in [7] thus reducing the complexity [6]. A significant improvement in terms of reduced complexity and memory requirement is achieved by replacing the safety-net with a limit on the SPP parameter update. There is slight improvement in tracking delay but residual noise components would still be present due to this delay. However, it provides very reasonable estimates for tracking both stationary and non-stationary noises.

2.1.2. Speech Power Estimation

The estimation of speech power spectral density is a challenging task as speech is even more non-stationary than most noises and so the speech periodogram exhibits large variance. A simple estimate of the speech power spectral density can be obtained by maximizing the likelihood function with respect to the clean speech variance. However, this gives rise to isolated spectral peaks in the estimated speech spectrum due to large fluctuations in the noisy speech periodogram [31]. Upon synthesis, these peaks are perceived as musical noise. The effect of musical noise can be reduced by averaging over successive noisy periodograms by assuming independent successive bins. However, this assumption is violated in practice due to the non-stationary nature of speech thus giving rise to distortions in speech onsets and transients and resulting in poor quality. Hence, *Ephraim & Malah* introduced a soft-decision based approach called the decision directed estimator that recursively updates the clean speech power spectrum based on the estimates in the previous frame [27].

The decision-directed method is a way of applying adaptive smoothing for fast tracking of increasing levels of the speech power. Consequently, fewer speech distortions are introduced at speech onsets and transitions as compared to the averaging. Nonetheless, being sensitive to rising spectral amplitudes, the issue of musical noise is not completely overcome. It not only responds to speech onsets, but also to abrupt shifts in the noise signal. Therefore, outliers in the noise are likely to contribute to the residual noise in the clean-speech estimate which may be perceived as annoying musical tones. In order to control the trade-off between the distortions and the effect of the residual noise, the smoothing parameter and a limit on the a priori signal to noise ratio must be carefully tuned [3].

Summary & Remarks

As seen in this section, a classical speech enhancement system consists of several modules dedicated for handling tasks such as speech estimation, noise tracking, speech power estimation and finally applying suitable gains to the frequency bands in the noisy speech spectra. These systems are derived analytically by studying the speech and noise distributions. To briefly summarize, the goal of this system is to suppress those frequency bands in which noise predominantly exists and enhance the ones in which speech prevails. This improves the intelligibility factor by a considerable degree but the enhancement in quality is limited due to existence of distortions and residual noise components. This occurs because of several reasons such as tracking delay introduced by recursively updating the parameters and suppression of bands in which speech and noise co-exist.

Most of the techniques seen so far assume that Speech DFT coefficients are Gaussian distributed for simplicity in analytically deriving the formulations despite knowing that the coefficients tend to be more super-Gaussian. Non-linear systems based on gamma and laplacian distributions are difficult to develop and analyze. Designing non-linear systems for this task is important to preserve the non-linearity in speech. Moreover, the human auditory system also applies non-linear transforms on speech. Due to the complexity involved in developing non-linear systems analytically, it seems promising to explore neural networks for this task because of their highly non-linear nature and their ability to adapt themselves to the data.

2.2. Speech Enhancement using Deep Neural Networks

Recently, Deep Neural Networks are also applied to the problem of speech enhancement and they seem to outperform most of the traditional techniques [8–11, 13, 23, 32, 33]. In this section, deep neural networks developed for processing speech that are based on a regression approach are highlighted. These techniques are called regression based because they minimize a distance metric between the network estimated speech features and the corresponding clean speech features. The usage of Neural network architectures enable high non-linearity in these systems which leads to an improved estimation of the speech manifold. Different methods are categorized according to the type of architectures used in the following sections.

2.2.1. Fully-connected Neural Networks

Xu, et al. designed a flattened, dense and fully connected feed forward DNN architecture that employs a non linear regression function on spectrograms of noisy and clean speech [8]. Certain strategies such as global variance equalization, noise aware training and dropout were implemented so as to improve the generalization capabilities of the network [8]. Experimental results indicated an improvement over classical methods in terms of objective measures. However, the estimated waveform does contain distortions for examples with mid and low Signal-to-Noise ratios. Distortions occur because this method minimizes the L_2 norm objective function between the network output and reference spectrograms. Thus, the model tries to match high energy (low frequency) signals precisely but fail to do so for low energy signals (high frequency). Moreover, for high dimensional spaces, distance metrics such as L_1 norm tend to perform better than the L_2 norm [34].

Instead of feature estimation, mask approximation methods namely, Ideal Binary Mask (IBM) [32] and Ideal Ratio Mask (IRM) [33] were proposed for estimating the target gain vectors using fully connected networks. The predicted gains were multiplied with the noisy speech spectrum to estimate the reference spectrum. An IBM is defined by applying a unity gain if the Signal-to-Noise Ratio (SNR) is greater than a pre-defined threshold and otherwise the gain is set to zero. A fully-connected network developed to estimate the IBM for speech enhancement [32] led to improvements as compared to some classical approaches. A significant improvement in speech intelligibility was reported. However, this system was based on using prior information not available in real life situations as the noise sequences used during training and testing were similar [35]. Thus, these systems did not reveal any information about performance in real life scenarios. Although, estimating an IBM led to a good performance, a continuous mask might perform better than a binary segmentation that seems too coarse for scenarios where speech and noise are likely to be present at the same time [33]. Additionally, a binary masks would cause more artifacts to be present in the estimated speech due to sudden changes. Thus, an IRM replaced the IBM where an IRM greatly resembles the Wiener filter in frequency domain with a tuning parameter. Note, that the IRM is not statistically optimal and was just motivated heuristically. A DNN trained for estimating the IRM target outperformed the IBM in terms of quality and intelligibility [33]. Though, it could not generalize well for varied types of noises such as babble and cafeteria noise but did considerably well for certain noise types not seen during training [35].

2.2.2. Recurrent Neural Networks

The benefit of using RNNs for this purpose is that it estimates the current set of samples by considering it's relationship with the previously predicted samples. The system proposed by *Osako et al.* was partitioned into three LSTM [36] layers where one each was designed to track noise, track speech and perform filtering [10]. Two different networks were trained, one for the magnitude spectrum and the other for the complex spectrum and concluded that both outperformed spectral subtraction [10]. The network trained for complex spectrograms outperforms the one trained using magnitude spectrograms because noisy phase was reused in the latter whereas it was estimated in the former. It is crucial to estimate the correct phase especially for low SNR examples. However, the improvement was not very significant because the L_2 norm minimization in the complex domain is not analytic and not differentiable, hence it had to be approximated [10]. *Osako et al.* believe that by altering the objective function they minimize, the output could be improved.

Jean-Marc V. developed a hybrid end-to-end speech enhancement system using Signal Processing techniques to extract features and RNNs to estimate the ideal ratio band gains that are applied between pitch harmonics along with pitch comb filters to attenuate the noise [11]. It produces significantly higher quality audio as compared to the traditional MMSE spectral estimator while having a complexity low enough for real-time operation on CPUs [11]. The architecture uses Gated recurrent Units (GRU) [37] as it outperforms LSTMs on this task [11]. *Jean-Marc V.* also reports that the loss function and training data construction plays a more significant role on the final quality of the results as compared to altering the network architecture [11].

2.2.3. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are also successful at enhancing noise corrupted speech signals. CNNs are known to have lesser trainable parameters as compared to Recurrent Neural Networks (RNNs) and Feed-forward Neural Networks (FNNs) which can make them suitable for applications which require less hardware resources [9]. However, CNNs are not suitable for real-time speech processing applications since they operate on blocks of data. *S. Park and J. Lee* introduced a novel network architecture called Redundant Convolutional Encoder Decoder (R-CED) which extracts redundant representations of a noisy spectrogram at the encoder and maps it back to a clean spectrogram at the decoder [9]. The network minimizes an L_2 norm between the clean and estimated denoised spectrograms. There is a slight improvement in speech quality as compared to RNNs and FNNs which is suggested by objective evaluation metrics. However, as discussed in section 2.2.1, the usage of L_2 norm yields a distorted output. Also, instead of extracting time-frequency spectral features, CNNs can also be applied directly on speech waveforms as they are known for temporal signal modeling as well [38].

Rethage et al. proposed a method for speech denoising using the WaveNet architecture. WaveNet is an autoregressive model used for generating speech samples from text inputs which is described in detail in section 2.3.2. However, this denoising method discards the generative modeling aspects and rather adopts a regression based discriminative model [13]. Additionally, their model introduces non-causality based on a facet that real-time applications can afford a few milliseconds of latency. However, this is not valid for real-time operations since future information is not available. *Rethage et al.* introduced an 'Energy-conserving loss' which is a combination of the standard L_1 loss for the estimated denoised signal and the estimated noise with respect to their respective references. The proposed model estimates not just one sample but a set of samples as compared to the original WaveNet which reduces the time-complexity problem in the latter [13]. The model is also conditioned on a binary encoded scalar corresponding to the speaker identity which has shown marginal improvement in results tested for the dataset but this information is unknown in many situations which questions its benefits in real-time application scenarios.

An end-to-end system based on a fully convolutional network architecture using context aggregation was developed for Speech Denoising [23]. This paper achieved state-of-the-art results in 2018. *Germain et al.* designed a system comprising of 2 CNNs, where one performs the denoising on raw speech waveforms (called Denoising Net) and the other performs an audio classification (called Feature Loss Net). First, the Feature Loss Net is trained for an acoustic scene classification task. This pre-trained Feature Loss network is then used to train the denoising network. The denoising network minimizes a loss called 'feature' loss that is defined by computing the weighted L_1 norm on the difference between the feature activations in different layers of the Feature Net corresponding to the reference speech waveform and the noisy speech waveform. This loss function was inspired from a computer vision research for image stylization and synthesis [39].

Summary & Remarks

All the methods discussed so far in the domain of deep learning have focused on speech enhancement based on regression. By incorporating various techniques, hybrid systems, different architectures all the approaches in the end attempted to either match the estimated speech waveform or spectral features (without correctly estimating the phase) with that of the reference. This was done by minimizing distance metrics such as L_1, L_2 norms or the newly introduced 'Feature' loss. The ideology behind this comes from an assumption that there is only one possible target speech waveform. The addition of noise introduces some uncertainty because of which details in the speech information are lost completely. Due to such mixing, the information is not conveyed correctly and it is difficult to retrieve the original target.

In order for the networks to succeed in correctly estimating the clean speech waveform, they require information about that particular waveform in terms of features or other representations as input. However, such information is not available in real application scenarios. As depicted in chapter 1 section 1.1, it is very unlikely that multiple recordings of the same utterance to have the same waveform. Thus, there is a possibility for a different objective apart from matching the waveform or its features. Following such a route has indeed displayed considerable improvement in listening but it has its limitations in terms of quality. As a result, the following sections in this chapter explore, generative neural network models as they can generate clean sounding speech samples by not constraining the system to preserve a particular waveform.

2.3. Generative Neural Network Models

As inferred by the concluding paragraphs of the previous section 2.2, a scope to explore generative neural network models was induced. Because generative models learn an effective mapping that can imitate the training data distribution, it can generate novel samples belonging to such a distribution. In the following segments, different generative models related to speech enhancement [12, 22, 24, 40] and speech synthesis [19, 21, 41] are discussed.

2.3.1. Generative Adversarial Networks

Very recently introduced, Generative Adversarial Networks (GANs) were also used for speech enhancement [12]. Most of the techniques discussed previously were designed to be used in the frequency domain whereas this method uses the raw audio waveform directly thus preserving information regarding phase. Having being trained for different speakers and noise types, the system incorporates generalization through shared parametrization [12]. GANs have shown good results when it comes to images but this is the first attempt at using GANs for audio [12]. Here, an encoding-decoding convolutional network with residual skip connections is used as the Generator network which takes the noisy speech signals as inputs and attempts to recreate clean speech signals with the help of the Discriminator network. One highlight of the generator is that it is an end-to-end structure. This system shows improvements in results as compared to the Wiener Filter in terms of subjective and objective measures. The objective function optimized by this network was inspired by Least-squares GAN [42]. After preliminary experiments, *Pascal et al.* found it convenient to add an L_1 metric to the generator objective function so as to minimize the distance between the generated waveform and and its corresponding clean samples. Moreover, experiments in [43] reveal that the SE-GAN method also relied more heavily on the L_1 norm minimization as compared to the GAN objective. The generative aspect came into effect once the model was trained well enough to be able to replicate the speech waveform. Thus, the true potential of GANs has not been harnessed. The system runs into a instability issues due to mode collapse if the L_1 criterion is discarded.

Training GANs is a tedious task and requires patience as they can be very unstable. Certain measures need to be undertaken in order to train them efficiently and encourage convergence as mentioned in [44]. The performance of GANs was further improved in terms of stability by using the Wasserstein loss criterion [45]. These networks are called Wasserstein Generative Adversarial Networks (WGANs). However, in certain situations these models also faced problems while training. To solve these problems certain strategies were suggested in the paper [46] which have proven to improve the performance of WGANs. One of the striking modifications was to penalize the norm of the gradients in the WGAN objective function and *Gulrajani et al.* named this technique Wassersteins GAN with Gradient Penalty (WGAN-GP). Experiments in [43] show that implementing these changes over the architecture employed in [12] enforces stability, however, the generated speech samples contained high frequency leaks. This occured due to implementation issues mentioned in [47] and can be rectified by hyperparameter tuning.

Further literature study revealed that *Shan et al.* had developed a similar speech enhancement system using 'WGAN-GP' [40]. However, the generator objective function also incorporated a weighted sum of both L_1 and L_2 metrics between the generated speech waveform and its reference. The procedure to set weights controlling the contribution of these metrics to the overall loss function is done through trial and error [40]. This change fails to add much value and also improvements in the results are very minor. Thus, these examples of GANs for speech enhancement exhibit the need for superior network architectures and improvements in finding better regularization metrics that not only stabilize the training but also yield better quality results.

In 2018, a group of researchers from NVIDIA corporation developed a novel generator architecture which yields state-of-the-art results and generated images with high-quality resolution [14]. The GAN is designed for the application of style transfer to morph the style of an image with another image which results in generating a new image that is a blend of both the input images. The proposed study introduces a way of reducing the entanglement in the feature space which leads to drastic improvements in image quality. Similarly, the idea can be extended to noisy speech being morphed to clean speech. *Takeru et al.* introduced a new study that aims to improve the instability in GAN training by proposing a regularization metric based on the spectral normalization of weights of the discriminator network [48]. Spectral Normalization is a method in which the weight of every layer in the discriminator is regularized by estimating the largest singular value for that layer using iterative techniques. Additionally, the method is free from intensive hyper-parameter tuning and does not even incur additional computation cost. claim that this regularization technique improves the quality of generated samples as compared to many previous techniques. The method seems promising to be implemented for improving the performance of the SE-GAN architecture. By constaining the weights of the network, the flow of information can be controlled which could lower the amount of high frequency leaks in the generated speech samples.

2.3.2. Autoregressive Temporal Convolutional Networks

van den Oord et al. developed a state-of-the-art deep neural network model that produces natural sounding speech from textual features. WaveNet is a fully probabilistic and autoregressive model that predicts the distribution for each audio sample conditioned on previous ones [19]. In order to capture, the short term and long term temporal dependencies in modeling audio, the network was designed using dilated causal convolutions [49] [50] with large receptive fields. The network outputs a categorical distribution over the next sample value with a softmax layer [19] and is optimized to maximize the log-likelihood of the data with respect to the parameters. Furthermore, to make the computations more tractable, μ-law companding transformation [51] was used to quantize the audio sample to 8-bit values instead of a simple linear quantization scheme. The model can also produce different kinds of voices by conditioning the data on speaker identity [19]. Subjective MOS test results indicate that the quality of generated speech is significantly better than all existing systems. One of the trade-offs of using WaveNets is that it has a massive architecture and requires more computational resources thus having a larger training time. Despite of taking many steps at reducing the time complexity by incorporating dilated convolutions and residual skip connections [19] it still requires large resources. A faster implementation of WaveNet that preserves the quality was launched very recently [52].

Another attempt addressing the problem of speech enhancement was made using the WaveNet architecture. This time its autoregressive and generative modeling properties were utilized. *Qian et al.* name it the Bayesian Speech Enhancement WaveNet framework as it uses three WaveNet models, one known as the prior model which learns the prior distribution of clean speech so as to regularize the output to be in the speech space and the other two form the likelihood model that learn to model the noise given the noisy raw audio and the estimated clean audio as input [24]. A combination of both these distributions is used to define the posterior expectation of the speech sample to be predicted. The likelihood model is very complex to train due to its humongous architecture and in addition to that it is non-causal in nature which it not very suitable for real-time applications. Moreover, the model is also speaker dependent and this information is not known in most real case scenarios. Moreover, due to increased complexity of the likelihood model, it is trained only on a small size of noisy training data thus making it less aware about the noisy conditions and more reliant on the prior model.

2.3.3. Autoregressive Recurrent Neural Networks

The SampleRNN model consists of a hierarchical structure combining memory-less modules named autoregressive multilayer perceptrons (MLPs) and stateful modules such as LSTM [36] or GRU [37] modules to capture the variation in temporal sequences in order to generate audio samples one at a time [41]. This model is unconditional, i.e., it does not require any hand crafted features describing the properties of the generated audio waveform. The modules within this system operate at different clock rates (in contrast to WaveNets) such that a module dedicated for audio sample generation operates at the sampling frequency and another module that maps slowly varying long term dependencies (phonemes) operates at a slower rate [41]. Hence, it provides flexibility in allocating computational resources in modeling different levels of abstraction. Since, SampleRNN is an unconditional model, it cannot be used for modeling speech as the output would simply contain mumbling sounds with varying voice and no sensible information. In order to generate speech, additional information should be provided such as pitch, Mel frequency coefficients or even text [21]. However, this literature provides a novel network architecture that can be used for generating high quality audio. On conditioning the input data with features that contain information about speech, this network can be made suitable for generating speech.

Thereafter, *Kalchbrenner et al.* proposed a sequential generative model with a focus on text-to-speech synthesis using single-layer RNN with a dual softmax layer that generates human-like sounding speech samples. Despite having a compact network architecture, the quality of generated speech samples match the state-of-the-art and at a rate much faster than existing text-to-speech models [21]. The main goal of this paper was to reduce the time complexity in generating high-fidelity audio samples from text in order to make it real-time. Moreover, by applying a weight pruning technique the number of weights in WaveRNN are reduced considerably thus making it a sparse network. The single-layer RNN architecture along with a weight pruning method reduced the number of parameters and hence the computations to a great extent [21]. This also makes it possible to run this network on a mobile CPU. Later, an even faster version was proposed by adding a Linear Predictive Coding (LPC) scheme to the WaveRNN network called the LPC-Net [53]. WaveRNN uses text input as features (although not mentioned in the literature about how it is used) conditioned with the previously generated speech samples to generate new samples. For the speech enhancement application, features from the noisy speech can be either handcrafted or extracted with the help of another neural network. These features would represent information related to speech and hence can be used for generating clean sounding speech.

2.4. Digital Signal Processing vs. Deep Learning

This section highlights some striking differences between Signal Processing and Deep Learning approaches. This helps in understanding certain advantages and disadvantages of both methods and justifying the shift to neural networks from signal processing.

1. **Linearity vs. Non-Linearity** - For simplicity, most signal processing methods used for noise removal apply linear transformations which restricts their capability of effectively mapping the target space. There exist non-linear signal processing methods as well but deriving them analytically and implementing them are difficult tasks. On the other hand, deep learning approaches are known to be highly non-linear processes and are thus suitable to capture non-linear aspects in speech as well. The human auditory system also involves many non-linear processes and taking inspiration from it to apply neural networks for processing speech is highly attractive. Moreover, given the fast-paced development in this field it is also relatively straightforward to implement.
2. **Assumption based vs. Data Dependency** - The shift from statistical signal processing based approaches to machine learning and deep learning methods are due to the fact that the former methods are based on certain assumptions whereas deep learning methods infer structure from the data provided which makes them better in certain conditions. Moreover, with the abundance of data available today, it becomes very practical to solve complicated problems for which analytically derived solutions can sometimes be impractical. However, when data is scarcely available, the performance of deep learning systems drops as well. This is where signal/data processing or machine learning methods can help.

3. **Scalability** - Deep learning solutions are a bit more scalable than signal processing methods. The reason being that deep learning methods are capable of handling various types of input features and large data size. By simply tweaking hyper-parameters, a deep learning based system can accept input of any type and dimension and learn its mapping to a category. Whereas, signal processing methods will have to be derived again and re-designed for it to work for a different set of features or data. However, deep learning solutions are also limited on their scalability because they have to be re-trained everytime a hyper-parameter is changed. This can be very time consuming depending on the complexity of the system.
4. **Complexity** - Signal processing solutions are less complex than deep learning solutions in terms of time, computational efforts, hardware resources and power consumption. From the trend in research, it is seen that the complexity of deep learning systems is on the rise as deeper networks are encouraged in order to improve model performance.

Summary & Remarks

Both CNNs and RNNs have shown their prowess in modeling sequential data streams. To proceed further, a decision has to be made for choosing a generative model to be implemented. Based on the existing generative models, a summary of the comparative analysis is made which helps in making this decision. Table 2.1 lists some characteristics of the generative systems which helps us compare the techniques.

Table 2.1: Comparative Analysis between several Generative Models for raw Audio

Characteristics	Autoregressive Temporal Convoluional Networks	Autoregressive Recurrent Networks	Generative Adversarial Networks
Examples for Speech / Audio Applications	WaveNet, Parallel WaveNet, ClariNet, Fast WaveNet	WaveRNN, SampleRNN, Tachotron	SE-GAN, SE-WGAN
Input Handling	Has a flexible receptive field due to dilated convolutions which makes these architectures capable of handling a larger number of input samples. Thus they are able to capture long term and short term dependencies in audio.	Long term dependencies can be difficult to capture as an RNN cannot look as far as a CNN in the input space. The inputs need to be fed serially. They can capture short-term dependencies very well.	The generator being made of CNNs gives them a flexible receptive field also making them capable of handling a large number of input samples thus enabling to capture long term and short term dependencies in audio
Training Time	Convolutional Architectures normally train faster on GPUs due to their parallel nature and shared parameterization. However, given the network size, number of parameters of the models listed in the examples, these networks take longer to train.	Generally, Recurrent Architectures being sequential in nature train slower on GPUs. However, the models listed in the examples have lower computation and overhead times which makes them equally fast or faster.	Having to train the Generator and Discriminator architectures simultaneously in one cycle, increases the number of computations, thus making them very long to train.
Training Stability	The models use Residual blocks of convolutional layers which leads to stable gradients ensuring stability in training.	Recurrent layers generally suffer from vanishing / exploding gradients due to backpropagation through time. However, by using LSTM or GRU units which the models in the examples use, these problems do not occur.	Generative Adversarial Networks are known for instability in training due to vanishing / exploding gradients and suffer from problems such as mode collapse making them extremely difficult to train.
Inference / Generation Time	Despite requiring one sample at a time for inference, audio generation is quite slow due to a complex network architecture. Parallelizing WaveNets have led to faster inference but longer training. Thus,not suitable for real-time applications.	Because Recurrent Networks maintain a single hidden state and have lesser number of parameters, they are capable of generating samples much faster.	They are moderately fast because only the generator is used. They are not capable of generating audio one sample at a time recursively as compared to the other 2 methods and require blocks of audio for generation. This makes it unsuitable for real-time applications because of latency issues.
Memory Retention	It has a larger memory retention due to large receptive fields and shared filters.	Relatively lower in practice due to a single hidden state. It can be improved by stacking more recurrent layers, however it would increase training and inference time.	It also has a larger memory retention due to shared filters in convolutional architectures.
Hardware Requirements	Hardware requirements are large due to heavy computations.	Hardware requirements are extremely small due to the less complex architecture.	Hardware requirements are moderate as only the generator is used.
Performance	Quality of generated speech is extremely good. Achieving state-of-the-art MOS scores.	Quality of generated speech is below TCNs but by a very small margin.	Quality of generated audio, is not upto the mark and suffers from distortions.

Amongst all the Generative modeling techniques seen so far, autoregressive TCNs have shown better performance in terms of output speech quality. However, they come with a highly expensive computational power. Autoregressive RNNs, being less complex and not compuitonally hungry, have also proven to perform well enough. Weighing the strengths and trade-offs of all the methods, WaveRNNs seem to be a practical approach to begin experimenting with.

3 Feature Engineering

The field of machine learning and pattern recognition rely on features that are characteristics of the data or measurable entities obtained from the data under observation. It is crucial to choose features that are informative, discriminative and independent in nature. This facilitates a better representation of the input space and so analyzing features plays an important role in order to make an appropriate choice. Deep learning models are known to extract their own learned feature representation from data. Input features to neural network models may or may not be independent.

This chapter is structured as follows where first characteristics of the dataset used for implementing the speech enhancement is mentioned. This is followed by segments explaining different features that can be extracted from speech and used as input for neural network models. These features include the speech waveform itself (section 3.2) and different time-frequency spectral features of speech such as spectrograms (section 3.3), Mel spectrograms (section 3.4) and Mel Frequency Cepstrum Coefficients (MFCCs) (section 3.5) and Delta features (section 3.6). These sections discuss their advantages and disadvantages for being used as context for the speech synthesis system. A modification is proposed to the Delta features in section 3.6 where a parameter is introduced to control the difference between time segments. Feature normalization is mentioned in section 3.7 where practical implementation problems with the min-max normalization is discussed and an alternative of using sigmoidal normalization is proposed. With deep learning, neural networks could also be used to map the input space to some intermediate latent representation as described in section 3.8. Later, a comparative analysis is conducted on the clean and noisy versions of these features and reported in section 3.9 which shows the effect of the addition of noise on different features.

For the chosen generative deep learning model as described in chapter 2, features act as context or conditioning information that represents the content within speech. Hence, with a feature analysis, one can discern what features are least affected by the addition of varied types of noises at different signal to noise ratios. This in turn helps to determine what noisy features would provide the most suitable context to the generative neural network model for speech synthesis.

3.1. The VCTK Dataset

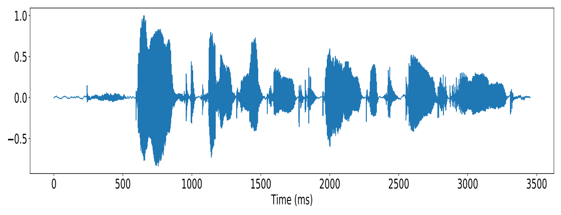
The speech dataset used for experimentation and analysis is called the VCTK dataset [54]. This dataset is available publicly and it encourages comparisons with many existing speech enhancement techniques. The dataset includes clean and noisy audio recorded at 96kHz sampling rate and downsampled to 48kHz. For many previously conducted studies and hence also for this thesis, the speech files are downsampled to 16kHz for training and testing [12, 13, 22] . The clean speech files are recordings of sentences sourced from various text passages, uttered by 30 English-speakers, both male and female, with various accents. 28 speakers constitute the training set and 2 are reserved for the test set [54]. Training samples are synthetically mixed at one of the following four signal-to-noise ratios (SNRs): 0, 5, 10 and 15dB, with one of the 10 noise types. From the 10 noise types, 2 are generated artificially and 8 are sourced from the DEMAND database [55]. This results in $11,572$ training samples from 28 speakers under 40 different noise conditions with approximately 10 different sentences in each condition per training speaker. In total, the training set consists of 10 hours of speech. Test samples are also mixed synthetically at one of the following four different SNRs: 2.5, 7.5, 12.5 and $17.5dB$, with one of the 5 noise types. This results in 20 noise conditions for 2 speakers. Thus, the test set features 824 samples from unseen speakers and noise conditions.

3.2. Audio Waveform

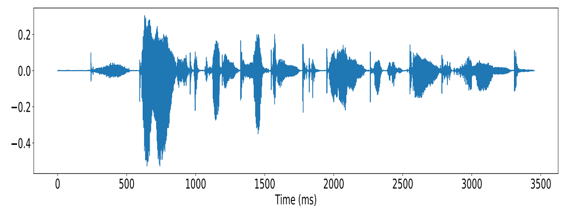
The dataset introduced in section 3.1 consists of audio waveforms for clean speech and its corresponding noise corrupted version. In end-to-end speech enhancement systems such as [12, 13, 22] speech waveforms were directly used as input features. However, if noisy speech waveform is used as context for generating clean speech, the information about the phase and alignment of samples is completely lost because of the addition of noise. The voiced speech components which have relatively higher energy are less affected in contrast to unvoiced speech that is strongly affected by noise. As a result, a technique known as pre-emphasis is applied (as a pre-processing stage) on the speech waveforms especially to raise the energy of unvoiced speech segments. The usage of pre-emphasis is inspired from [12] as it was used to increase the energy of high frequency speech components. Pre-emphasis is applied as shown in equation 3.1

$$y(t) = x(t) - \alpha x(t-1), \quad \forall\ t = 1, 2, ..., T-1, \tag{3.1}$$

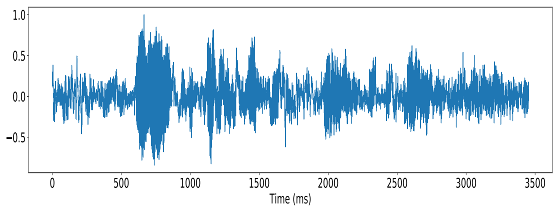
where, y is the pre-emphasized version of the original waveform x governed by a factor α. α normally varies between $0.9 - 1$. t stands for the time indices for the entire length of the audio samples T. The pre-emphasis operation can be inverted to obtain the raw waveform and is called de-emphasis. Figure 3.1 shows the raw clean and its corresponding noisy speech waveforms with their pre-emphasized versions.



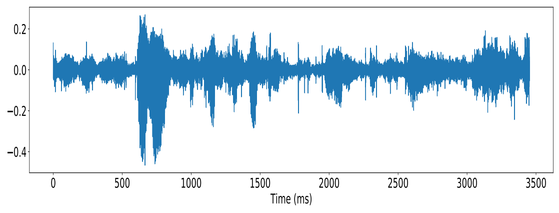
(a) Raw Clean speech waveform



(b) Clean speech after applying pre-emphasis α=0.97



(c) Corresponding speech waveform corrupted with babble noise



(d) Noisy speech after applying pre-emphasis α=0.97

Figure 3.1: Clean and Noisy speech waveforms with their pre-emphasized versions for an example in the dataset

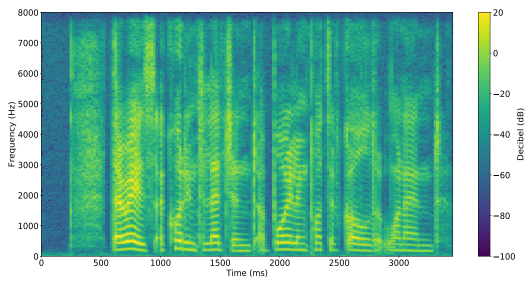
A single channel waveform contains information about the energy, phase and alignment of the samples. Despite this, it is difficult to comprehend any meaningful linguistic information on a sample level. Ideally, deep learning does not and should not involve any feature analysis as the network (with increasing depth) is capable of extracting hidden features within the data. However, this highly increases the model complexity, would require large amounts of data to learn and might result in other computational issues. Thus, simply using the noisy waveform as context to synthesize clean sounding speech does not seem to be a very practical approach. Hence, extracting spectral features from speech by applying waveform decomposition techniques, namely the short-time Fourier transform (described in chapter A, section A.1.1) can be utilized.

3.3. Spectrograms

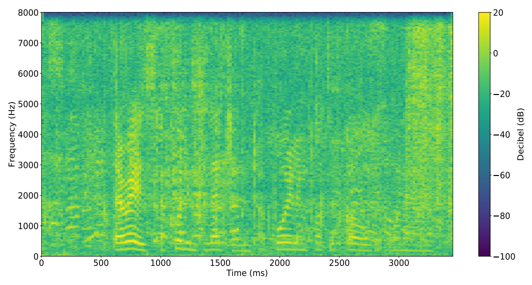
The power spectrum of a speech waveform calculated over short time frames describes the distribution of power into frequency components composing the waveform. A spectrogram is a visual representation of the power spectrum as it varies with time. Hence, the spectrogram is a time-frequency visualization of audio and is calculated as follows:

$$\begin{aligned} Y_k(t) &= \sum_{n=0}^{N-1} y(n+tD)w(n)e^{-\frac{j2\pi k}{N}n}, \quad \forall\, t = 0, 1, ..., \left\lfloor 1 + \frac{T-N}{D} \right\rfloor \\ S_k(t) &= |Y_k(t)|^2, \\ S_k^{dB}(t) &= 10\log_{10}(\max(10^{-10}, S_k(t))) \end{aligned} \tag{3.2}$$

where, y is the speech waveform (can be raw or pre-emphasized) which is multiplied by the window function w and Y is the time-frequency representation of the signal obtained by applying a short time Fourier transform $stft$ (described in chapter A, section A.1.1). The frequency bins are represented by k and t stands for the number of short time segments of length N overlapped by D samples (typically while obtaining the spectrogram frame lengths of $20-40$ ms with a 50% overlap is used). The spectrogram S is obtained by taking the squared magnitude of Y. The intensity can also be displayed on the decibel scale S_{dB} as such a logarithmic rescaling aligns the amplitude axis with human perception of loudness. Figure 3.2 shows the pre-emphasized clean and its corresponding noisy spectrogram. The spectrogram in figure 3.2 was calculated over a 40 ms speech frame with 50% overlap and a 1024-point DFT. A Hanning window function was applied to minimize spectral leaks. The frequency bins are represented in the continuous domain (Hz) by applying a suitable scaling. The effect of pre-emphasis is visible in figure 3.2 as the high frequency components also have higher energy on the decibel scale.



(a) An example of a pre-emphasized clean spectrogram



(b) Corresponding noise corrupted spectrogram of the same example in (a)

Figure 3.2: Clean and their corresponding Noisy spectrograms obtained from pre-emphasized speech for an examples in the dataset

Spectrograms are a 2D representation of speech that convey information about the energy of speech and/or noise components present in different frequency bands of clean/noisy speech and the variation of this energy across time. The temporal axis of a spectrogram is orders of magnitude more compact than that of a waveform, meaning dependencies that span tens of thousands of time-steps in waveforms only span hundreds of time-steps in spectrograms. However, they lack the phase and alignment information completely as shown in equation 3.2. Hence, synthesizing a waveform only from spectrograms is difficult when phase information is missing. On addition of noise to clean speech, the energies of speech and noise components co-exist during speech activity which leads to loss in speech information. This loss in information can lead to incorrect context representation. Moreover, the dimensionality of spectrograms is quite large which can increase the model complexity and resources.

3.4. Mel Spectrograms

Mel spectrograms are derived by applying the Mel filter bank on spectrograms. Mel filter bank is a set of triangular filters that peak (having a response of 1) on their centre frequencies and decrease linearly towards 0 till it reaches the center frequencies of the two adjacent filters. Mel-filter banks are displayed in figure 3.3. These centre frequencies are linearly placed on the Mel-scale. The Mel-scale aims to mimic the non-linear human ear perception of sound by being more discriminative at lower frequencies and less discriminative at higher frequencies. The reason for using the Mel space is inspired by the human auditory system, in particular the cochlea, as it cannot discern the difference between two closely spaced frequencies. Hence, the Mel transform is known to align the frequency axis with the human perception of pitch. The spectrogram estimate still contains a lot of information which may not be required in the context. In essence, The Mel spectrogram takes the weighted average of the energy present at different frequency bands in the spectrograms with its neighbouring bands. The Mel-scale is a transform derived from the Hertz-scale and is represented by equation 3.3. This mapping from Hertz to Mel is interchangeable.

$$\begin{aligned} m &= 2595\log_{10}\left(1+\frac{f}{700}\right), \\ f &= 700(10^{\frac{m}{2595}}-1), \end{aligned} \tag{3.3}$$

where, m and f represent the frequencies on the Mel and Hertz scales respectively. It can be inferred that if centre frequencies are placed linearly on the Mel-scale then they are placed exponentially on the Hertz-scale. This is also seen in figure 3.3 where the lower centre frequencies are placed close by and the higher ones are widely separated. The first filter is narrow and would indicate the energy present in the band around the minimum selected centre frequency (100Hz in the figure). As the frequencies get higher, the filter gets wider and would be less concerned about variations in the spectrum. The triangular filters around these centre frequencies are modeled as follows:

$$H_m(k) = \begin{cases} 0, & k < L(m-1), \\ \dfrac{k-L(m-1)}{L(m)-L(m-1)}, & L(m-1) \le k < L(m), \\ 1, & k = L(m), \\ \dfrac{L(m+1)-k}{L(m+1)-L(m)}, & L(m) < k \le L(m+1), \\ 0, & k > L(m+1), \end{cases} \tag{3.4}$$

where, H denotes the triangular Mel filter bank filter with m centre frequencies and k frequency bins present within the spectrogram. And, L is the list of the number of mel-spaced centre frequencies.

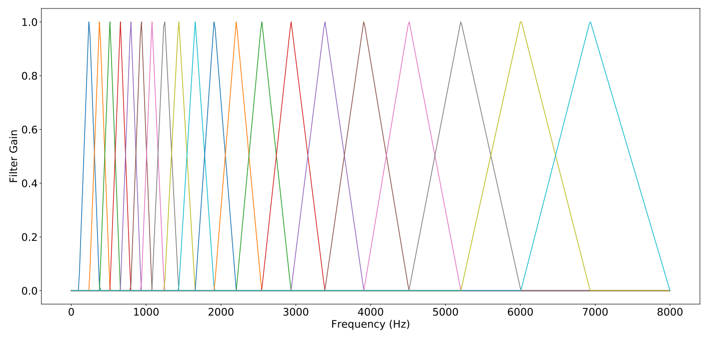


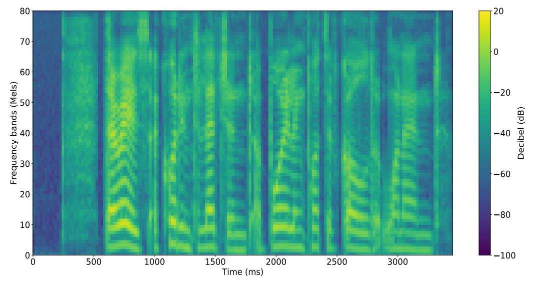
Figure 3.3: Mel Filter Bank for 20 filters

After designing the Mel filter bank, it is applied to the spectrogram as follows:

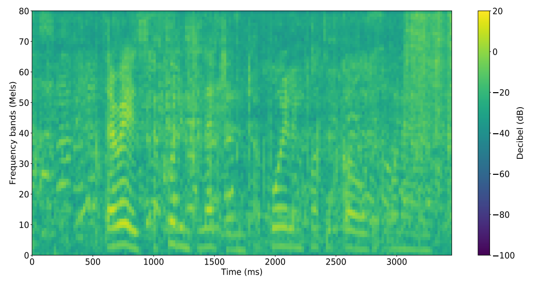
$$MS_m(t) = H_m(k) \circ S_k(t), \tag{3.5}$$

where, MS stands for the Mel spectrogram at the m^{th} centre frequency, H represents the Mel filter bank and S denotes the Spectrogram with k frequency bins and t time segments.

Figure 3.4 shows an examples of the clean Mel spectrograms and its corresponding noise corrupted version with 80 Mel bands extracted from the frequency bins in the spectrogram shown in figure 3.2. The harmonics during speech activity are well preserved for clean speech examples, however, with the addition of noise, these harmonics are a bit distorted (depending on the noise power and its characteristics).



(a) An example of a pre-emphasized clean Mel spectrogram



(b) Corresponding noise corrupted Mel spectrogram of the same example in (a)

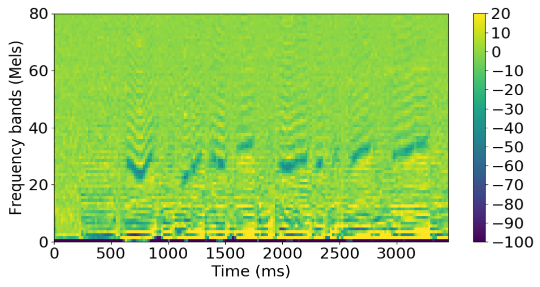
Figure 3.4: Clean and their corresponding Noisy Mel spectrograms obtained from pre-emphasized speech for an example in the dataset

3.5. Mel Frequency Cepstrum Coefficients

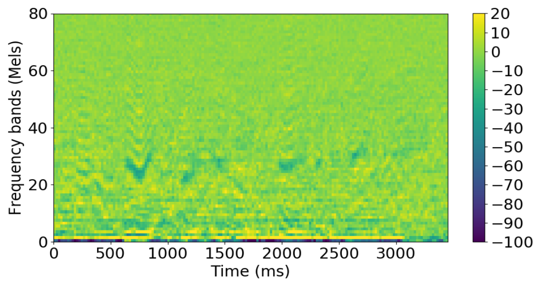
Mel Frequency Cepstra are obtained by applying a Discrete Cosine Transform (DCT) on the Mel Spectrogram. Mel Frequency Cepstrum Coefficients (MFCCs) are known to contain linguistic information in speech and hence are extensively used in the application of Automatic Speech Recognition (ASR) [56, 57] . In many ASR literature, the higher MFCC bands (typically above 13) are discarded as they represent very fast changes and tend to degrade the performance of speech recognition systems [58]. However, for a speech enhancement application it is suspected that using lesser bands might lead to a loss of information (especially high frequency sounds or consonants). MFCCs are obtained by applying a DCT as follows:

$$mfc_m(t) = \sum_{m=0}^{M-1} MS_m(t) \cos\left[\frac{\pi}{M}\left(m + \frac{1}{2}\right)k\right], \quad \forall\, k = 0, 1, ..., M-1, \tag{3.6}$$

where, mfc indicates the mel frequency cepstrum coefficient and MS stands for the Mel spectrogram at the m^{th} center frequency for all M centre frequencies and t time segments. Figure 3.5 shows the Mel frequency cepstra for clean speech and its corresponding noise corrupted version. The values of the clean MFCC features vary within the range $[-500, 200]$ and that of the noisy MFCC features vary within $[-300, 100]$. However, in order to increase the contrast, this range is reduced to $[-100, 20]$ while plotting. This shows the variations along the Mel bands more prominently as compared to a plot with the original range.



(a) An example of clean MFCC features



(b) Corresponding noise corrupted MFCCs of the same example in (a)

Figure 3.5: Clean and their corresponding Noisy MFCCs obtained from pre-emphasized speech for an example in the dataset

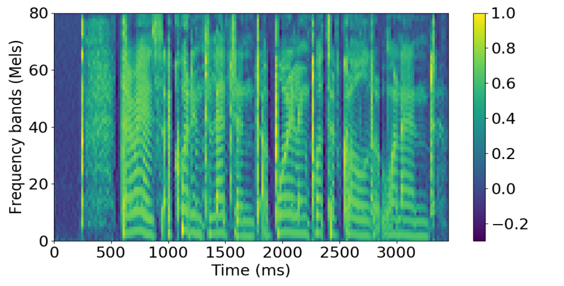
One of the outcomes of applying a DCT on the Mel spectrograms is that the energies in neighbouring bands become decorrelated. Since, the Mel filter bank has overlapping filters, the energies of the bands in the Mel spectrogram are quite correlated with each other. This decorrelation of features turns out to be fruitful for many machine learning based approaches. It is because of this decorrelation property exhibited by the DCT, MFCCs were so attractive to machine learning models. However, when it comes to deep neural networks, the structure of the data is modelled implicitly without any explicit assumptions about the data distribution. Hence, there is no added advantage of favouring MFCCs over Mel spectrograms just because they are decorrelated [58]. Besides, a study also reveals that MFCC features are not very robust to noise, however, normalization can make then partially robust to noise [58].

3.6. Delta Features

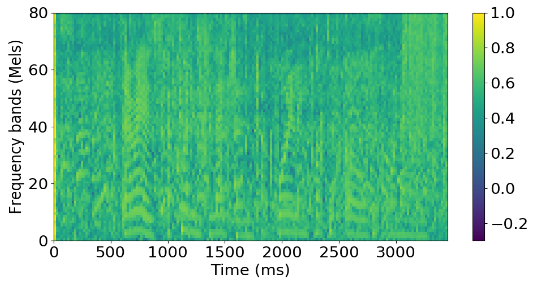
Delta features are also used in addition with Mel features for many speech recognition applications [58]. Delta features are also known as differential coefficients since they represent the change in the Mel features along time. The Mel feature vector describes the power spectral envelope of a frame, but speech also has dynamic information, i.e., the trajectories of the Mel feature coefficients over time. Calculating these trajectories and concatenating them with the original Mel feature vector increases performance of ASR models significantly [58]. Delta features can be obtained for both types of Mel features, i.e., Mel spectrograms and MFCCs. Ideally, Delta features are obtained by simply taking the difference between a Mel feature vector at the current time index and a feature vector in the past. Here, an additional variable is used to control this difference. This idea was inspired by the concept of pre-emphasis explained in section 3.2.

$$D_i = M_i - \alpha * M_{i-n}, \quad i = 0, 1, .., N - n. \tag{3.7}$$

where, D represents the Delta feature vector and M stands for the Mel feature vector at time index i. n stands for the shift in time index. A typical value for $n = 1 or 2$. α controls the difference and should vary within the range $0 - 1$. If the value of α is closer to 1, then only the onsets of speech or high variation in noise will be present in the Delta features. Whereas, for a low value of α, the Delta features and Mel features would be alike. Thus, a mid value in between $0.4 - 0.6$ is ideal since it relatively amplifies the onset of speech and also preserves the harmonic structure.



(a) An example of clean Delta features with $\alpha = 0.4$



(b) Corresponding noise corrupted Delta features with $\alpha = 0.4$ of the same example in (a)

Figure 3.6: Clean and their corresponding Noisy Delta features obtained from pre-emphasized speech for an example in the dataset

Figure 3.6 shows the Delta features obtained from Mel spectrograms by setting the α parameter to 0.4. Note that these features are calculated after normalizing the Mel features as described in section 3.7. Thus, the onsets of speech are amplified which were initially not so clear in the Mel spectrogram or MFCC features. The speech onset contains information about consonants and unvoiced speech sounds which are generally missing in the presence of noise. Thus, these intricate details are now visible to the network and they can also be mapped effectively. Delta features used in the past had to be concatenated with the Mel features and this increased the dimensionality of the input features which in turn increases model complexity. With the introduction of the α parameter, the Delta features not only contain information about the speech consonants but also harmonics in voiced speech. Thus, there is no need to concatenate Delta features proposed in this segment with the Mel features.

3.7. Normalization

Before the features are fed into the neural network it is important to normalize the features in order to avoid computational problems and biased results. Features have to be scaled down from the Decibel (dB) scale to a standardized range of $[0, 1]$. One of the ways to normalize these features is by applying a min-max normalization as follows:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.8}$$

where, x represents the features in the decibel scale and x' represents the data in the normalized scale. However, deciding on how to obtain the minimum or maximum values for the normalization is crucial here as some practical problems were faced during implementation. The existing ways of selecting the minimum and maximum for a min-max normalization are as follows:

1. Fixed range: The minimum and maximum decibel (dB) values are fixed for the features for all examples, i.e., clean and noisy speech. By observing a general trend and visualizing data, let's assume a mapping from $(-100, 20)$ dB $\rightarrow [0, 1]$. This mapping means any value above 20 dB would be clipped to 1, any value below -100 dB would be clipped to 0 and the values within the decibel range would be scaled by the formula mentioned in 3.8. The latter case would not matter much, as -100 dB is an extremely low power value and inefficiently mapping low values would not severely affect the estimation performance. However, clipping off higher values can lead to loss of spectral peaks which correspond to the formants in speech spectrum. Formants in speech spectrum provide important information about speech characteristics. Additionally, while retrieving values back to the decibel (dB) scale for analysis of estimated features could lead to faults due to approximations because of the clipping. Choosing a maximum value depends on the Signal to Noise ratio (SNR) which is unknown in general. One simple solution to this problem is to choose a maximum value that is high enough for all data samples. But this can compress the range very drastically. Another way could be to subtract a reference decibel level before normalizing in order to avoid peak clipping but it is still not a fully guaranteed method. Motivating the choice of such a reference level is also difficult.
2. Dataset dependent: The maximum and minimum values can be obtained from the dataset used for training the network. This will at least prevent the clipping of peak values within the training set, however, it is still not reliable in terms of unseen data (for example, the test set). Retrieving estimated features for further analysis would also be inaccurate as the decided minimum and maximum value depends on the dataset and is not universally true.
3. Data dependent: In order to avoid the peak clipping, the minimum and maximum values can be obtained separately for different examples from the dataset. However, the normalization would not remain standardized anymore. For example, if a clean speech example within the dataset varies from $[-100, 20]$ dB and its corresponding noisy version varies from $[-80, 40]$ dB then after normalization, 1 represents 20 dB for the clean example and also 40 dB for noisy speech. Moreover, retrieving estimated features back to the decibel scale from the normalized range is not possible since the minimum and maximum decibel values are not fixed. Hence, an analysis on estimated features cannot be performed.

The approximations involved in these methods could not only lead to some missing details in the normalized representation but also somewhat faulty representations while retrieval. As a result, a different normalization method is proposed which avoids the hard clipping of peak values to 1 irrespective of how high the maximum can get and also be completely retrieved back to the original decibel scale. This makes it a completely standardized and invertible normalization approach. The normalization technique can be expressed as follows:

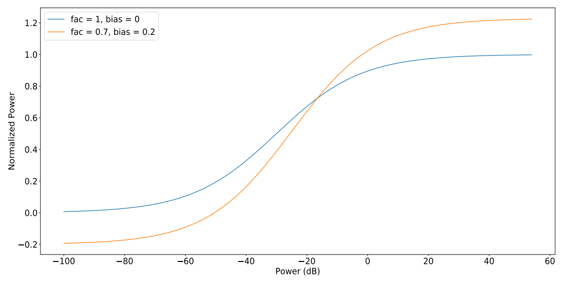
$$\begin{aligned} x_s &= \frac{x - \text{shift}}{\text{scale}}, \\ x' &= \frac{1}{\text{fac} + e^{-x_s}} - \text{bias}, \end{aligned} \tag{3.9}$$

where, x' is the normalized representation of the input data x. The expression for deriving x' is inspired from the sigmoid function mentioned in chapter A section A.2.1. Sigmoid acts as a soft clipping function as it varies extremely slow at higher input values tending towards 1 at input ∞ and at lower input values saturating at 0 for $-\infty$. The control variables fac and bias are used to control the normalization range, i.e., any other value instead of $[0, 1]$. Setting values for fac and bias do not require much effort and depends totally on the decided normalization range.

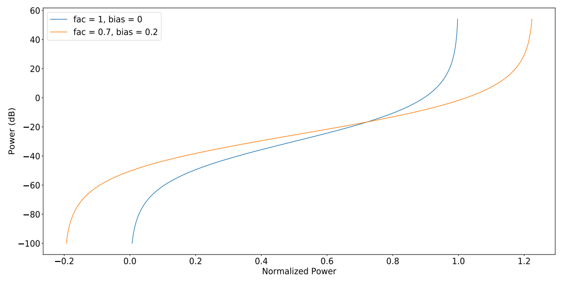
The input to the sigmoid-like function is the scaled down representation denoted by x_s. Since, the original decibel scale is a skewed, the shift term compensates for the skewness and scale is used to scale down from the shifted decibel range to $[-5, 5]$. The values for shift and scale can be easily calculated by solving two simultaneous equations. This range of $[-5, 5]$ is chosen because the sigmoid function almost saturates at those extremities. An advantage of using this normalization is that even if a fixed range of input decibel values is decided, an outlier input value will never be hard-clipped to 1 and will remain distinguishable from other input values that fall within the fixed range. An additional benefit of using this normalization method is invertibility back to the decibel range. However, the inverse transform is sensitive to outlier input values. If the input value is is very close to the extremes, i.e., 0 or 1, it could be mapped to a very large value on the decibel scale which is not desirable. In case of such extreme situations, the output decibel scale must be clipped to -100 dB and 50dB. The inverse transform of the sigmoidal normalization can be expressed as:

$$\begin{aligned} x_s &= \log\left(\frac{x' + \mathrm{bias}}{(1 - \mathrm{fac} * \mathrm{bias}) - \mathrm{fac} * x'}\right), \\ x &= x_s * \mathrm{scale} + \mathrm{shift}. \end{aligned} \tag{3.10}$$

Figure 3.7 shows the modified sigmoidal curve that maps from the decibel scale to normalized scale and its inverse transform. Before, applying the sigmoidal transform, the decibel range of $[-100, 30]$ dB is mapped to $[-5, 5]$ by setting the variables $\mathrm{shift} = -40$ and $\mathrm{scale} = 14$. As shown in the figure, the control variables fac and bias control the normalization range, i.e., when $\mathrm{fac} = 0.7$ and $\mathrm{bias} = 0.2$, then the normalized outputs vary from $[-0.2, 1.2]$ and for $\mathrm{fac} = 1$ and $\mathrm{bias} = 0$, the normalized outputs vary from $[0, 1]$. On comparing the two mappings, the former can be a bit advantageous as the slope of the normalized values is slightly larger than 0 at the extremities. Hence, it does not clip faster and can preserve the spectral peaks better.



(a) The sigmoidal normalization transform



(b) Inverse transform of the sigmoidal normalization

Figure 3.7: Variation in the normalization range by changing the variables fac and bias

3.8. Latent Features

Latent features are intermediate representations that are obtained by mapping the input features such as the raw waveform or their spectral features using a neural network. Thus, neural networks could themselves determine the most striking or distinguishing characteristics from the input space for the task they are trained for. One such example is shown in chapter 5 section 5.6.3.1. In this thesis, these features have not been used for training the WaveRNN synthesizer but this could be an interesting route to explore in the future. An advantage of doing so would be that the network now receives an input that would be the most meaningful representation since it was learned by the network itself. An idea of implementing such a system is discussed in chapter 7 section 7.3.

3.9. Feature Analysis

After an illustration of the different features that can be used to represent audio and its context to the WaveRNN synthesizer, this section reports an analysis conducted on the noisy and clean versions of these features. This study can help choose appropriate features that represent the speech content in noisy speech well and that are also not severely affected by the addition of noise. The Pearson's correlation coefficient is used to determine the effect of noise on different Mel features. Figure 3.8 shows the diagonal elements of the the correlation matrix (between the Mel bands of the clean and noisy features. The noisy versions of the Mel spectrograms, Mel Frequency Cepstral Coefficients and Delta Features are compared with their respective corresponding clean versions. These features are a time-frequency representation of speech, thus the correlation coefficient is calculated by keeping the Mel bands as variables and the time-segments as observations. This analysis is done for examples of all 4 Signal-to-noise Ratios, i.e., 2.5dB, 7.5dB, 12.5 dB and 17.5 dB and for 4 different noise types namely, 'Cafe', 'Public Square', 'Bus' and 'Living' present in the test dataset.

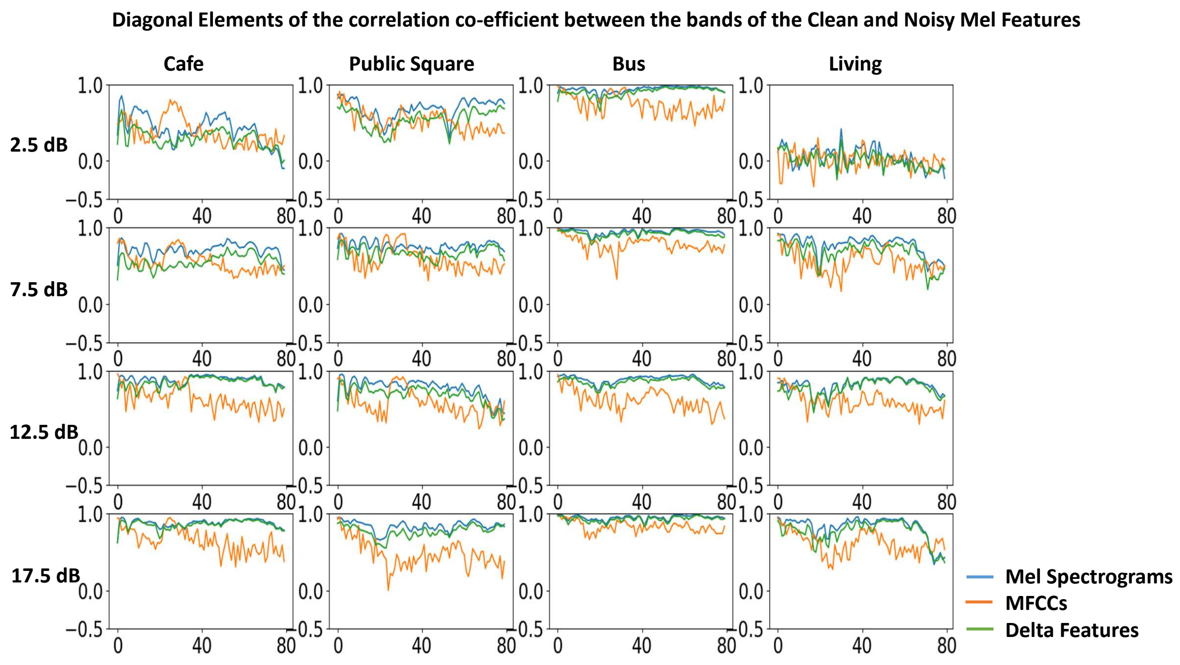


Figure 3.8: Diagonal Elements of the Pearson's Correlation Coefficient between the 80 Mel bands of the Clean and Noisy Mel features. The vertical axis in each subplot represents the correlation coefficient. The horizontal axis stands for the number of Mel bands. The matrix of subplots consists of examples with different noise types and Signal-to-Noise ratios shown by the left and top labels in the figure respectively.

For higher noise power, the correlation between the clean and noisy versions of all features is poor except the noisy type 'Bus'. This noise type does not seem to affect the speech severely since the correlation in almost all bands is close to 1 at all SNRs. As compared to the Mel spectrograms and Delta Features, the noisy MFCCs show a slightly lower correlation with clean MFCCs and decreases even further in the higher Mel bands. This indicates that the MFCC features are affected by the presence of the 'Bus' noise type. Such a characteristic is seen in many such plots for different SNRs and noise types, where the noisy MFFC features show a relatively lower correlation with their clean versions as compared to other features. Hence, on an average it can be inferred that MFFCs are affected by the presence of noise more severely than other Mel features. However, there are two examples for 'Cafe' noise and 'Living' noise at 2.5 dB where the correlation coefficients between the noisy and clean versions of all the features are equally affected showing a strong influence of noise.

Concluding Remarks

This chapter introduced various features that can be extracted from a speech waveform and can be used as informative context for training neural network systems. This section summarizes the key takeaways from the topics discussed. Conclusions from this chapter form a baseline for the subsequent chapters. The following conclusions are drawn from this chapter:

- Using the raw wavefrom of speech as input to the WaveRNN synthesizer does not seem to be the most appropriate choice since the raw waveform does not contain any linguistic information on the sample level and not to forget the presence of noise in the input for this application. Indeed, deep learning models can extract meaningful context from the raw waveform but this would cost more in terms of time, complexity and resources to capture long term dependencies in the time domain.

- Providing features in the time-frequency domain seems to be like a viable option since the temporal axis of a spectrogram is order of magnitude more compact than a waveform. However, spectrograms still have a high resolution which not only make the model complex but also allow noise to be presented to the model as equally as speech information since the input in this case is corrupted by noise.
- Features in the Mel domain are an attractive choice since it aligns more with human perception. Due to the Mel transform, the resolution of the features reduces and it preserves speech information at the same time reducing the effect of noise to a certain extent.
- Applying a DCT over Mel spectrograms to obtain Mel frequency cepstral coefficients (MFCC) does not seem to pose any added advantage. Moreover, applying a linear transform could discard some highly non-linear information in speech. Research in the past has also revealed that MFCCs are susceptible to noise which make them less attractive.
- The Delta features proposed in this chapter would be interesting to experiment with since they not only amplify the power for consonants and unvoiced speech segments but also preserve the speech harmonics in the voiced components of speech. Thus, there is no need to append them with Mel spectrograms or MFCCs as was done in previous studies. This also reduces the dimensionality of the input making the system less complicated.
- An analysis on different Mel features shows that MFCCs are affected by noise more severely than others. Hence, MFCCs must not be used for speech synthesis when they are noise corrupted.
- The sigmoidal normalization technique for speech features proposed in this chapter seems to be a plausible solution for avoiding the clipping problem that arises with the min-max normalization.

4 The WaveRNN Synthesizer

Originally the WaveRNN architecture [21] was introduced to synthesize natural human-like sounding speech from text. In this thesis, the possibility of synthesizing natural sounding speech from context provided by features extracted from noisy speech is explored. The features used for this purpose are discussed in chapter 3.

This chapter delineates the design of a WaveRNN synthesis system in section 4.1, its auto-regressive generation process and training procedure in sections 4.2 and 4.3 respectively and hyper-parameter definitions along with the system setup in section 4.4. Furthermore, different experiments were performed with clean and noisy context and the results obtained are discussed in section 4.5. Some additional experiments are also mentioned in Appendix B. The outcome of these experiments reveal that the WaveRNN can generate clean sounding speech but it suffers from low intelligibility when noisy features are provided as conditioning input. Hence, features must be enhanced externally before synthesizing speech. Section 4.6 describes the analysis that was conducted on the network performance. The analysis also includes visualization of components such as the hidden state of the GRU, output probability distributions, distributions of the network parameters to gain an insight into their functionality. In the end section 4.6.3.4 summarizes the key findings and provides concluding remarks.

4.1. System Design

A rudimentary WaveRNN model was constructed for experimentation. To begin with, some characteristics inspired from the original WaveRNN model [21] are adopted, whereas certain other aspects (mentioned later) have not been included for simplicity. However, this architecture is considerably modified as compared to the original WaveRNN. Figure 4.1 illustrates the architecture design used for training a Speech Enhancement WaveRNN system. The training procedure and speech generation process are described in detail in sections 4.3 and 4.2 respectively. Some details about the inputs to the network and its design are mentioned in the following paragraphs.

The WaveRNN Architecture displayed in figure 4.1 primarily consists of a Gated Recurrent Unit (GRU) layer [37] followed by two fully connected feed forward layers. Being a compact network with less layers allows this architecture to be trained faster and generate speech with few operations, thereby less computations are involved in each step. Moreover, having a hidden state to maintain the aggregated and compressed representation of the context makes RNNs quite suitable for this purpose. Additionally, within a single transformation, RNNs are able to combine the hidden state with the input context. GRU units being free from the problems of exploding or vanishing gradients ensure training stability as mentioned in Appendix A section A.2.6. Thereafter, a fully connected layer with ReLU non-linearity is used. The last layer is also a fully connected layer but with a softmax non-linearity. This yields a categorical distribution for predicting the value of a speech sample given the context provided by the previous sample and the noisy features for that sample. The network details such as its configurations and hyper-parameter settings are described in section 4.4.

The WaveRNN synthesizer relies on normalized features in the Mel domain that are extracted from speech such as Mel Spectrograms, Mel Frequency Cepstra or Delta features as described in chapter

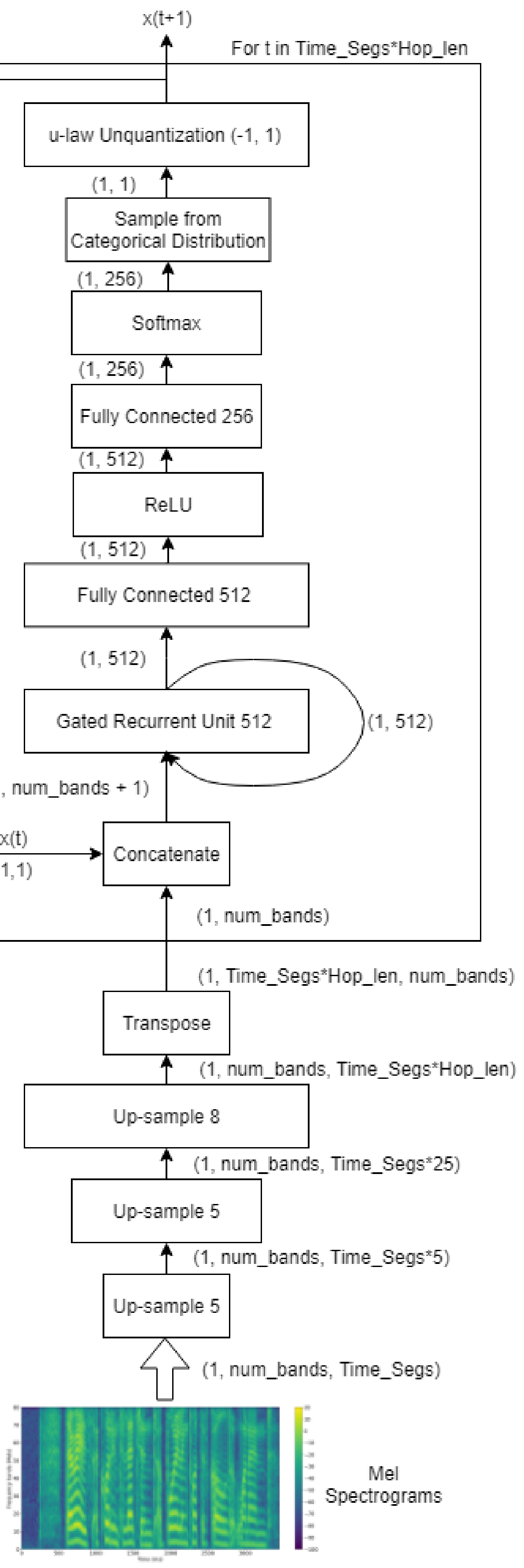


Figure 4.1: A fully trained WaveRNN Architecture used for synthesizing speech from noisy features

3 sections 3.4, 3.5 and 3.6 respectively. However, before inputting these features to the WaveRNN model, they are passed through an upsampling network which is explained in detail in the following paragraph. Additionally, speech samples from the previous time-step are also supplied as input data to the WaveRNN and concatenated with the upsampled Mel features as seen in figure 4.1. While training this network, a segment of clean speech (that corresponds with the time length of the Mel features) is provided as input data so as to enable the auto-regressive nature during generation phase. Hence, the network learns a relationships between the clean speech waveform conditioned with its corresponding upsampled feature space to effectively model speech.

The upsampling network consists of three layers as shown in the bottom of the figures 4.1 and 4.2. Noisy Mel Spectrograms are fed to this network and the main purpose of the upsampling network is to present the features to the network at the sampling frequency. Each upsampling module consists of stretch and convolution operations along the time dimension of the features, thus preserving the number of Mel bands. In the figure each upsampling module has a number, N, written on it which stands for the stretch operation, i.e., the number of repetitions of every time segment in the input features. Convolution by a learnable kernel (size $[1, 2 * N + 1]$) leads to smoothing out the energy across time (analysis mentioned in segment 4.6.3.4 reveals this characteristic of the upsampling layers in this network). Note that there is only 1 channel in all these convolution layers. One may argue that there is no real purpose of dividing the upsampling network into several layers as there is no non-linear operation such as ReLU or sigmoid being applied after convolution layer. However, dividing them into different layers gives some insight into feature interpretations. Furthermore, fewer parameters are used by the upsampling network when it is divided into multiple layers instead of using just 1 layer.

4.2. Generating Speech

Before describing the training procedure for the architecture, it would be beneficial to gain an understanding of how this synthesizer generates speech in an auto-regressive manner. Thus, this section is based on the assumption of a fully trained WaveRNN architecture. Given noisy features in the Mel domain as conditioning input, the model generates one sample of speech at a time and re-uses the previously generated sample to predict the next, thus modeling speech as a conditional distribution. To understand the idea behind using neural networks for generative modeling, refer to Appendix A section A.2.7. Mel features extracted from noisy speech (after being passed through the upsampling network) act as conditioning features to the previously generated sample. After these features were scaled and matched to the sampling frequency of audio, all feature bands at a particular time index were used to predict a sample. The overall operations involved in the proposed architecture are as follows (biases omitted for brevity)

$$\begin{aligned} f_t &= [x_{t-1}, M_t] \\ u_t &= \sigma(R_u h_{t-1} + W_u f_t) \\ r_t &= \sigma(R_r h_{t-1} + W_r f_t) \\ e_t &= \tau(r_t \circ (R_e h_{t-1}) + W_e f_t) \\ h_t &= u_t \circ h_{t-1} + (1 - u_t) \circ e_t \\ P(x_t) &= \mathrm{Softmax}(O_2 \mathrm{ReLU}(O_1 h_t)) \end{aligned} \tag{4.1}$$

where f_t is the concatenation of the previously predicted speech sample denoted by x_{t-1} and the upsampled Mel features denoted by M_t at the current time-step t. u_t, r_t, e_t and h_t are the 3 gates and hidden state of the GRU layer respectively. The weight matrices R and W are formed from matrices R_u, R_r and R_e and W_u, W_r and W_e which are computed as a single vector product to produce the contributions of all 3 gates u_t, r_t and e_t. σ and τ are the sigmoid and tanh non-linearities. O_1 and O_2 represent the weights of the 2 fully connected layers with their respective ReLU and Softmax non-linearities to obtain the posterior distribution $P(x_t)$.

The first input speech sample is initialized to $'0'$ as clean speech generally has no utterance in the first few milliseconds. The initial hidden state vector for the GRU unit is also initialized to $'0'$. After passing the Mel features through the upsampling network, it's dimensions along the time axis are now approximately the same as the length of the audio file [1]. After this, every time index in the upsampled

[1]The length is perhaps shorter than the original size because the network simply multiplies the time dimension in the features

Mel features is concatenated with the previous sample predicted by the network. This input stream is fed to the GRU layer which is then passed on to two fully connected layers, one with ReLU non-linearity and the other with Softmax to obtain a categorical posterior distribution. The output from the GRU layer, besides being propagated forward to the subsequent layers, is also reused to update the hidden state of the GRU as described in Appendix A section A.2.6. The next speech sample is obtained by sampling from the categorical posterior distribution of the 256 values it can take for 8-bit speech encoding. The predicted sample is in the quantized state, i.e., discrete or in the integer domain, $\mathbb{Z}$, and so in the end, an un-quantization step is applied to map the speech sample back to a continuous domain, $\mathbb{R}$. After this step, the predicted sample is fed back as input again and the same cycle continues for all the time frames in the speech. The predicted probability distribution for a speech sample is visualized and analyzed in segment 4.6.3.2.

This method of generating speech samples is yet not very fast. The reason and calculations are explained in section 4.6.1. The original WaveRNN incorporates a batched sampling with sub-scaling method for generating new samples. This method is not implemented in the proposed system.

4.3. Training

One of the main parts of training is defining how the input data and/or conditioning features are presented to the WaveRNN synthesizer. This is done by defining a 'collate' function which gathers all the information, i.e., noisy Mel features, input and output speech (ground truth) that are required for training this model. As mentioned before, the WaveRNN network is provided with clean speech as input so that it learns how to model speech alongside Mel features which provide information about the linguistic context in terms of the energy contribution within the Mel frequency bands. A look back period known as the 'sequence length' is defined which determines how many samples of clean speech are presented to the network during training. Generally, more the samples the better the network can learn as it can capture long term dependencies in speech. However, the number of computations and training time can increase to a great extent. Furthermore, within the collate function, a window length, w is calculated based on the sequence length S_L and the hop length h of the audio (the definitions and configurations of these terms are mentioned in section 4.4) for determining the number of time segments of the noisy Mel features that are provided to the upsampling network. A random time frame of Mel window length, w from every noisy feature in the batch and its corresponding clean speech segment of sequence length are then selected for training. The output speech is an 8-bit μ-law encoded version of the same clean speech as the input but shifted by 1 sample. The network tries to predict this μ-law encoded version of the clean speech for every sample. Note that, the sample is assigned an integer label value within the range $(0, 255)$ instead of a real number representation. The original WaveRNN model proposes a 16-bit encoding and so the clean speech is split into coarse 8 bits and fine 8 bits. This part has not been considered in the proposed system.

Note that, there is one architectural difference while training the network, i.e., a log softmax operation is used in the final layer instead of a softmax which is used while generation. Both serve the same purpose but using a log softmax over a softmax function is advantageous. Apart from practical reasons like improved numerical performance and gradient optimization that boost training performance, the log softmax function is known for heavily penalizing the model when it fails to predict the correct sample value. Moreover, using a log softmax over a softmax function means yielding log probabilities over probabilities thus leading to information theoretic interpretations.

Network training is performed on different batches of the data. For every batch, the model is loaded and its parameters are updated by optimizing a negative log likelihood (NLL) loss function as described in Appendix A segment A.2.2.2. Since, the output layer of the WaveRNN network while training is a log softmax as shown in figure 4.2, the loss function turns out to be a cross entropy loss. This is expressed as follows:

$$L(O, X_q) = -\frac{1}{n}\sum_{i=1}^{n} log\left(\frac{e^{O[X_q]}}{\sum_{j=0}^{255} e^{O[j]}}\right) = \frac{1}{n}\sum_{i=1}^{n}\left[-O[X_q] + log\left(\sum_{j=0}^{255} e^{O[j]}\right)\right] \tag{4.2}$$

where, $L(p, X_q)$ stands for the cross entropy loss, O is the output of the last fully connected layer of the WaveRNN network and X_q represents the vector of sequence length containing 8-bit quantized speech

by the hop length without adding the first frame length. The number of frequency bands is preserved in this operation

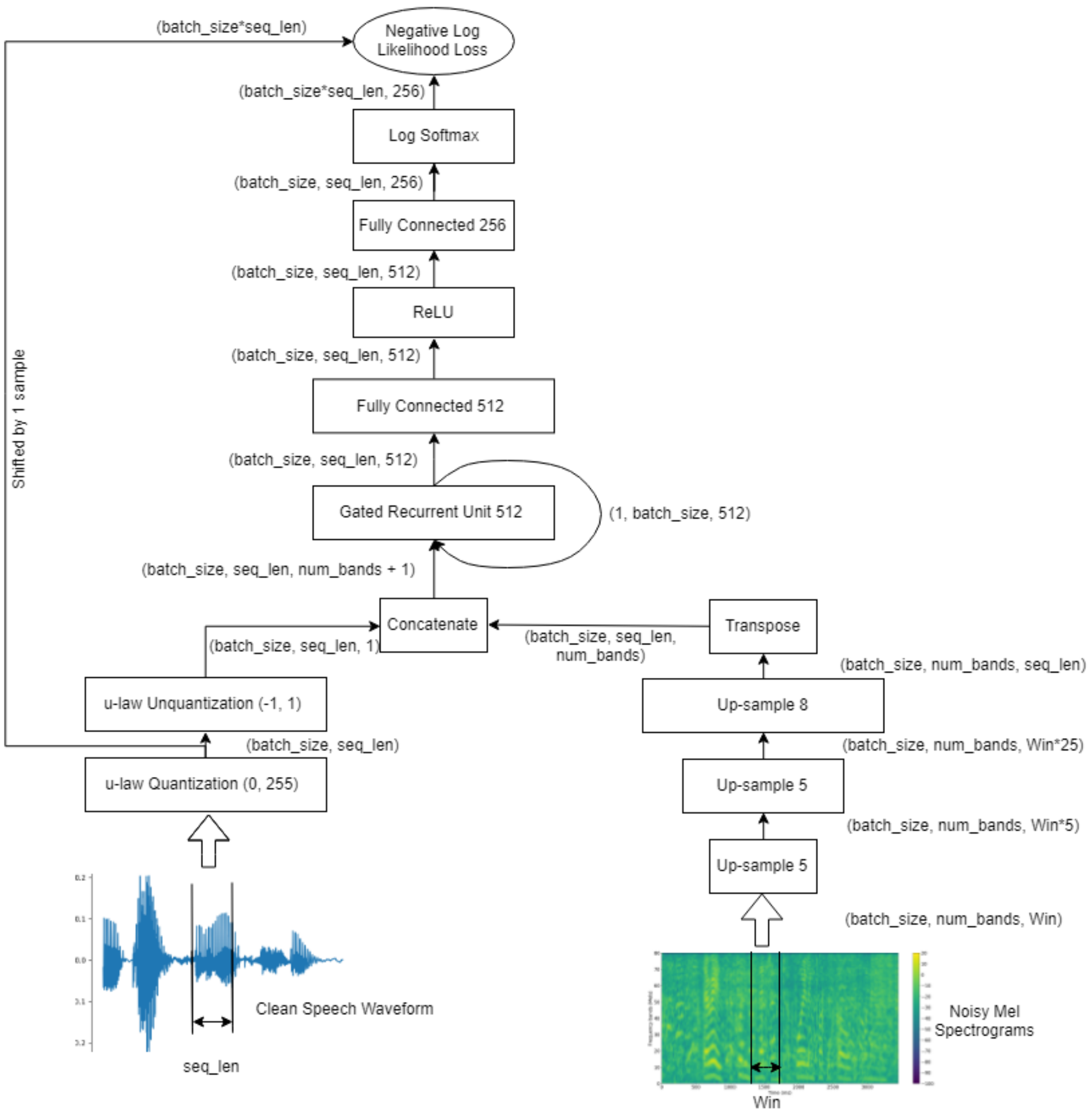

Negative Log Likelihood Loss
(batch_size*seq_len)
(batch_size*seq_len, 256)
Log Softmax
(batch_size, seq_len, 256)
Fully Connected 256
(batch_size, seq_len, 512)
ReLU
(batch_size, seq_len, 512)
Fully Connected 512
(batch_size, seq_len, 512)
Gated Recurrent Unit 512
(1, batch_size, 512)
(batch_size, seq_len, num_bands + 1)
Shifted by 1 sample
Concatenate
(batch_size, seq_len, 1)
(batch_size, seq_len, num_bands)
Transpose
(batch_size, num_bands, seq_len)
u-law Unquantization (-1, 1)
Up-sample 8
(batch_size, seq_len)
(batch_size, num_bands, Win*25)
u-law Quantization (0, 255)
Up-sample 5
(batch_size, num_bands, Win*5)
Up-sample 5
(batch_size, num_bands, Win)
Clean Speech Waveform
seq_len
Noisy Mel Spectrograms
Win

Figure 4.2: Block diagram for training a WaveRNN synthesizer to produce clean speech given noisy features

samples. Equation 4.2 depicts that the output at the last layer of the network, O, is the logarithm of a discrete distribution (softmax) over 256 values for each sample averaged over the mini-batch of size n. This distribution determines the probability of occurrence of a sample value given the previous samples and the context provided by Mel features for that sample. On a sample level, this network can be viewed as a multi-class classifier that classifies based on the 8-bit quantized representation of clean speech that varies from $(0, 255)$. Thus, the value of a particular sample in the entire speech segment is determined. An Adam optimizer [59] is used to minimize the NLL loss and the parameters are updated by backpropagating the gradients through the network.

4.4. Setup

Network hyper-parameter definitions and configurations are mentioned in this section. The following table 4.1 enlists the parameters used in the network, their setup and some remarks.

Table 4.1: Setup of the WaveRNN Synthesizer

	Parameter Name	Format	Remarks
Network Hyper-parameters			
	GRU units	R	Number of Gated Recurrent Unit Nodes
	Fully Connected layer 1 units	F	Number of nodes in the first fully connected hidden layer
	Fully Connected layer 2 units	P	Number of nodes in the last fully connected layer. Should be equal to the unquantized representation of audio varying from [0, 255]. Based on 8-bit μ law encoding.
	Upsample factors	$[U_1, U_2, ..., U_n]$	Factors by which the time dimensions in the features are scaled. Note: Product of all factors should be equal to the Hop Length.
	Sequence Length	S_L	Look back period of audio samples while training. Scalar multiple of the hop length. Scalar value chosen arbitrarily.
	Batch size	b	Batch size of data loaded into the model.
	Epochs	e	Training cycles.
	Learning rate	lr	Learning rate for the optimizer.
Feature configurations			
	Feature		Type of features i.e, Mel spectrograms or MFCCs
	Mel bands	M	Number of Mel bands.
	Mel window	w	Number of time segments considered in Noisy Mel specrogram while training. Calculated as S_L / $h + 2 * pad$ where $pad = 2$.
	Sampling Frequency	sr	
	Window Length		Number of samples in a short time frame of audio.
	Hop Length	h	Number of overlapping samples. 50% overlap considered.
	FFT size	N	N point DFT is computed on every short time frame.

4.5. Experiments & Results

Experiments were performed by adjusting certain configurations described in section 4.4. The motivation and aim for each experiment is stated in their respective segments. Results obtained from the respective experiments are exhibited. There were no major changes made in the WaveRNN Synthesizer architecture from the description mentioned in section 4.1. The inference and training procedure also remains the same as described in section 4.2 and 4.3 respectively. In this section, only a few experiments that align well with the narrative in this chapter are mentioned. However, some extra experiments that also led to important findings are mentioned in Appendix B.

4.5.1. Train with noisy Mel spectrograms as context

The goal of this experiment is to evaluate the performance of the system when noisy Mel spectrograms are used as context for the synthesis of clean speech. Before conducting this experiment, there were a few initial attempts at training the WaveRNN model with Mel Frequency Cepstrum Coefficients (MFCCs). Some of these experiments are mentioned in Appendix B. Results obtained from the experiments described in segments B.1.1 and B.1.2 lead to an urge for analyzing different spectral features that can constitute appropriate context for the synthesizer. Since, there was no considerable improvement in intelligibility observed after making the respective changes, it was essential to view the features from a critical perspective. This led to an analysis described in chapter 3 section 3.9 where it was discovered that MFCC features are severely affected by the addition of noise. On the other hand, Mel spectrograms are also being affected by noise, but this effect is not as intense as that on MFCC features as depicted in figure 3.8.

After training is completed for the configurations mentioned in table C.3, Mel spectrograms extracted from the noisy speech present in the test set are provided as context to the fully trained WaveRNN synthesizer. Based on this context, the synthesizer generates audio samples in an auto-regressive fashion as described in section 4.2. The results obtained are illustrated in figures 4.3 and 4.4 where, the waveforms of the clean, noisy and generated speech along with their respective spectrograms are displayed. The generated speech waveform looks completely different from the clean speech waveform as expected from a generative model. On listening to the generated audio file from context provided by noisy Mel spectrograms, the following observations are made:

1. The generated speech is free from noise irrespective of SNR as no noise characteristics are heard.
2. No glitches and artifacts are present.

3. Blabbering and mumbling sounds occur especially in extreme noise scenarios. Consonants and unvoiced sounds are unclear. This makes it difficult to understand the speech and hence intelligibility is poor. However, there is a slight improvement over the results obtained from experiments B.1.1 and B.1.2.
4. Pitch and intonation of the speaker are different from the original clean speech example. In case of extreme noise in input features, the speaker voice and prosody keeps changing for different segments within the speech. However, this happened more frequently in results obtained from experiment B.1.1.

From previous experiments described in B.1.1, B.1.2 and this experiment, it was clear that mumbled speech and blabbering sounds occur due to the presence of noise and simply reconfiguring the network hyper-parameters would not alleviate this problem. Since, Mel spectrograms are not as severely affected by noise as MFCC features, the effect of noise was not as pronounced and so there was a slight improvement in intelligibility. The generated speech still does not sound very clear and this clarity is far from that of the reference clean speech.

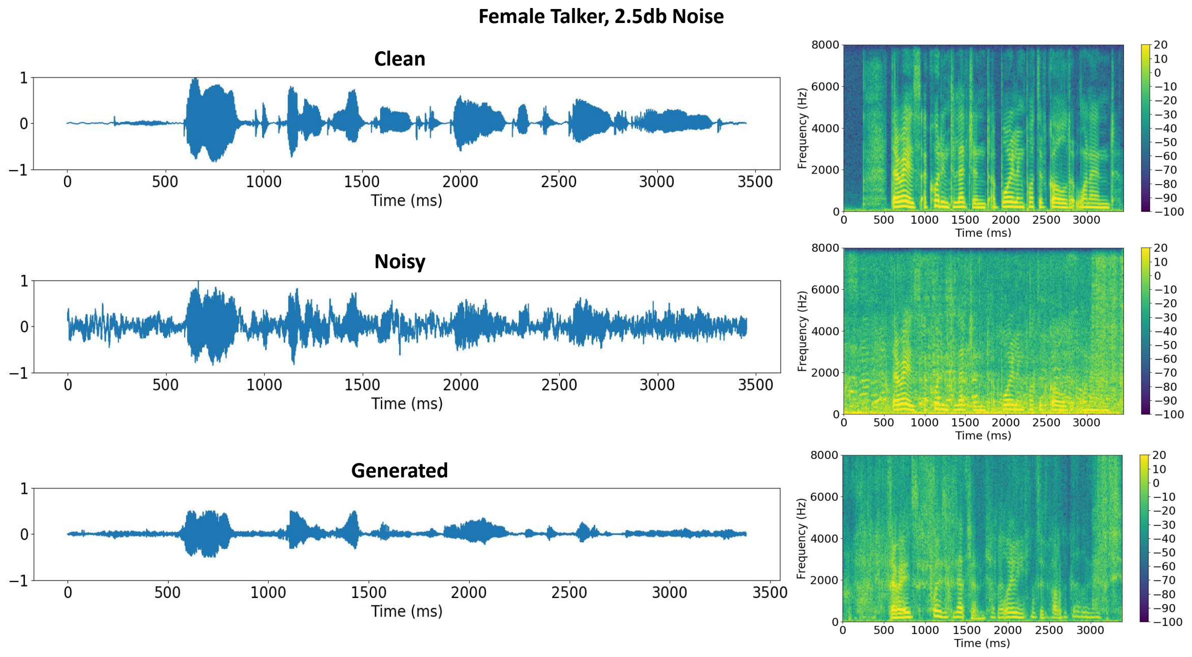


Figure 4.3: **Top**: Clean speech waveform and its spectrogram for a female talker. **Middle**: Babble noise added at 2.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from noisy Mel spectrogram as context, and its spectrogram.

Some characteristics heard in generated speech are also observed in the spectrogram of generated speech as shown in figures 4.3 and 4.4. Spectrograms of generated speech in both examples, show that the waveform is free from distortions and artifacts. In the spectrogram of generated speech, the spacing in between the harmonics is different from that in the spectrogram of clean speech. Thus, generated speech sounds different from the clean speech. This spacing does not change very often as observed in the figures B.1 and B.2 which shows that voice change within the same speech example is lesser than before. The intonation of the speaker is also different from the reference clean speech and this is visible by comparing the spectrograms of generated speech with that of the clean reference. The variation in the harmonics along the time axis is different for almost all speech segments in the spectrogram of generated speech from the spectrogram of clean speech. Despite the effect of noise being less severe on Mel spectrograms, the correct pitch and intonation was not captured by the network. Thus, it would be beneficial to verify whether the synthesizer is capable of reproducing similar sounding speech when only clean features are provided.

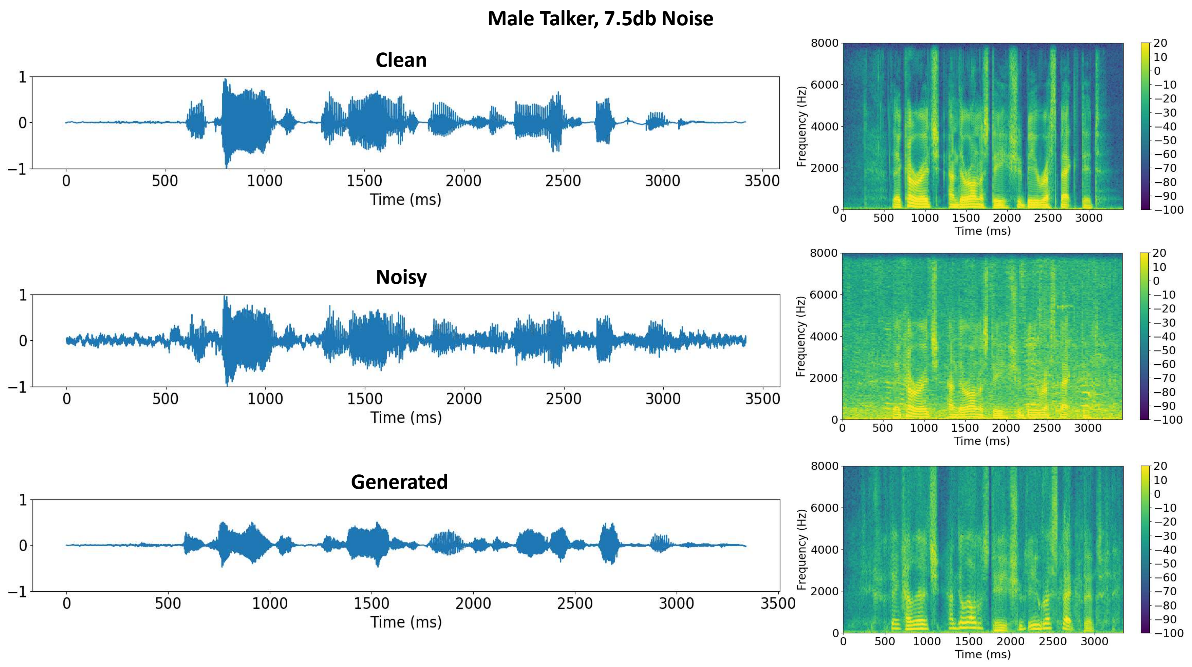


Figure 4.4: **Top**: Clean speech waveform and its spectrogram for a male talker. **Middle**: Babble noise added at 7.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from noisy Mel spectrogram as context, and its spectrogram.

4.5.2. Train with clean Mel features as context

The aim of this experiment is to test whether the WaveRNN network is capable of generating clean speech if trained for a longer time and given clean context. From the results of the previous experiments described in segments B.1.1, B.1.2 and 4.5.1, it was clear that the presence of noise in the features was indeed causing a severe loss of information such as pitch, intonation and consonant sounds (in extreme noise cases). Thus, the generated speech synthesized by the WaveRNN model was not intelligible even though the speech sounded free of noise, artifacts and distortions. The measures taken in the previous experiments did lead to minor improvements in the intelligibility, however, generated speech was still not completely understandable and there is a significant scope of improvement. Thus, it felt imperative to verify whether, the WaveRNN synthesizer can generate speech of at least the same quality as the reference clean speech when context provided was clean? Moreover, in the previous experiments it was suspected that the system was not trained long enough. WaveNet based generative models are trained for at least a million steps (close to $1-2$ weeks) or even more while, this system was trained only up to 365k steps (approximately three days). Hence, the number of training cycles was also increased in this experiment.

After training is completed for the configurations mentioned in table C.4, Mel features extracted from the clean speech present in the test set are provided as context to the fully trained WaveRNN synthesizer. Note that, since Mel features are extracted from clean speech for training, the number of Mel Bands was again set to 80 so that details of speech can be completely captured. The synthesizer was trained for two cases separately, once with clean Mel spectrograms and then with clean Delta features. Based on the context provided by these features, the synthesizer generates audio samples in an auto-regressive fashion as described in section 4.2. The results obtained for training with Mel spectrograms are illustrated in figures 4.5 and 4.6. While the results obtained for training the synthesizer with Delta features are shown in figures 4.7 and 4.8. The waveform of the clean and generated speech along with their respective spectrograms are shown for both the cases. After listening to the generated speech, it is difficult to distinguish it from the reference speech since they sound so alike. Despite this, the waveform of generated speech looks completely different from the clean reference even though the clean features were used. On listening to the audio files obtained from both the cases, the following observations are made:

1. No distortions or artifacts are present in the generated speech.
2. The voice and tone of the speaker are correctly reproduced.
3. No information is missing and speech is free from mumble sounds and blabbering.
4. Speaker sounds hoarse as his/her voice quavers when trained for the same number of epochs as in experiments 4.5.1 B.1.1 and B.1.2. The quality improves by large margins when the network is trained longer and it sounds as good as the reference clean speech.

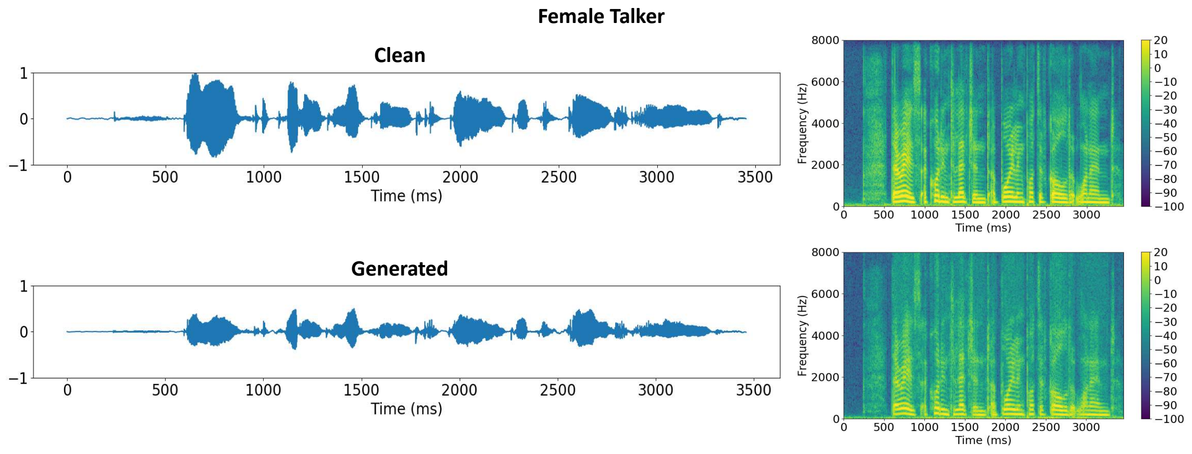


Figure 4.5: **Top**: Clean speech waveform and its spectrogram for a female talker.
Bottom: Speech generated from clean Mel spectrograms, and its spectrogram.

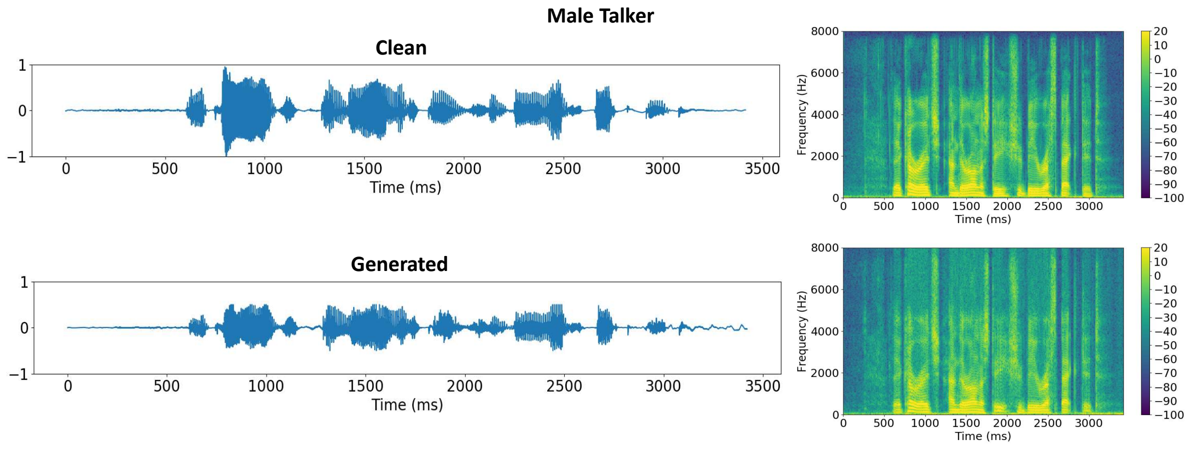


Figure 4.6: **Top**: Clean speech waveform and its spectrogram for a male talker.
Bottom: Speech generated from clean Mel spectrograms, and its spectrogram.

An analysis on the spectrograms of speech generated from Mel spectrograms shown in figures 4.5 and 4.6 reveals that the harmonics in the clean and generated spectrograms are approximately equally spaced which indicates that the pitch or the speaker voice identity is preserved. Similar characteristics were observed in spectrograms of speech generated from Delta features as shown in figures 4.7 and 4.8. For most cases, the generated waveform did not exhibit any abrupt discontinuities thus indicating that the generated speech is free from distortions and artifacts. However, in figure 4.7 there is an artifact present in the time segment between $1800 - 1950$ ms. Such occurrences are attributed to instability in the GRU layer and is displayed in segment 4.6.3.1. An analysis on the hidden state of the GRU layer for that example shows that such regions contain large variations in magnitude and do not

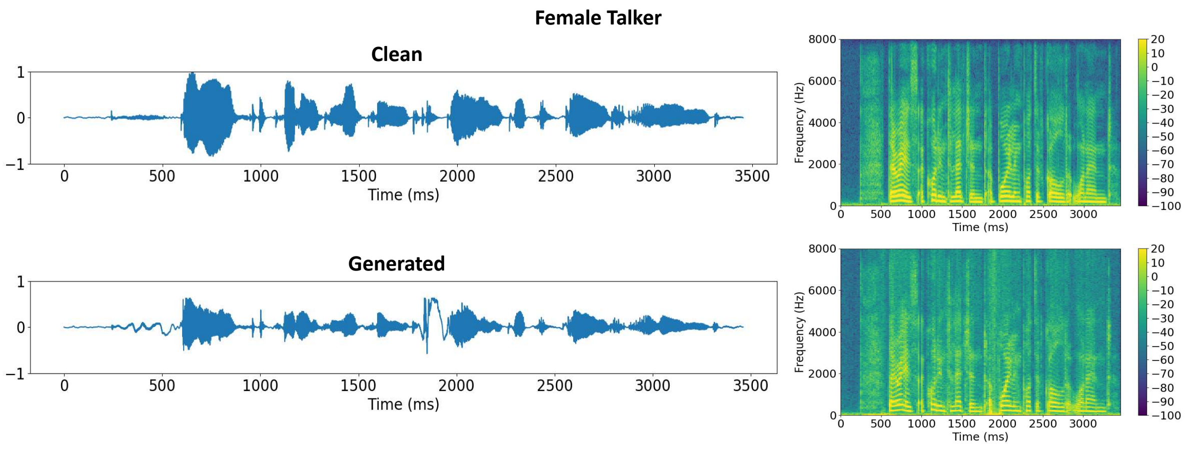


Figure 4.7: **Top**: Clean speech waveform and its spectrogram for a female talker.
Bottom: Speech generated from clean Delta features, and its spectrogram.

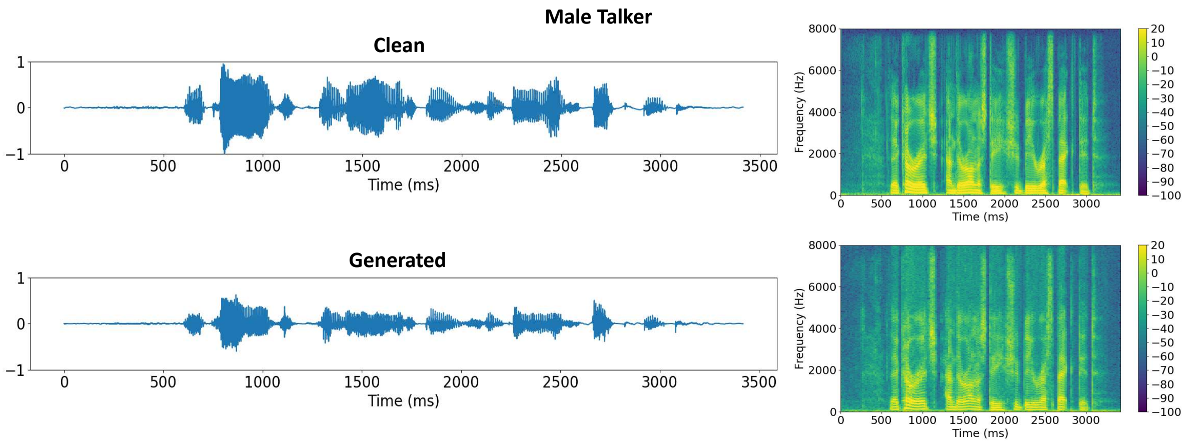


Figure 4.8: **Top**: Clean speech waveform and its spectrogram for a male talker.
Bottom: Speech generated from clean Delta features, and its spectrogram.

appear to be as uniform as other silence regions as shown in figure 4.10. This instability in the GRU layer can be either due to insufficient training or over-training with a large learning rate [60]. Also note that, the existence of such artifacts is very random and do not always occur. If the same example is generated once again with the same conditions, the waveform may or may not contain such artifacts.

Thus, the hypothesis was confirmed that when clean features are provided to the WaveRNN synthesizer, it generates clean sounding and intelligible speech. Moreover, the WaveRNN synthesizer is capable of generating clean sounding speech from any type of feature as far as the features are clean. From this experiment, it was inferred that the WaveRNN architecture is not capable of denoising features. However, for the speech enhancement problem only noisy features are available as input. Thus, there can be two possible routes from here, where

1. The architecture can be changed by either adding more layers or introducing a regularization on the waveform like other methods such as [12, 13, 24, 40].
2. Denoise the noisy features separately and then use them to fine-tune the WaveRNN synthesizer that is pre-trained on clean features. This is inspired by a phenomenon called 'Transfer Learning'.

In the first approach, adding more components to the network can make it more complicated and does not guarantee any improvement in results. Moreover, if a regularization term for matching the waveform is used, then the technique does not remain solely generative anymore. In the second approach, another neural network architecture can be designed specially for denoising the features. Thereafter, the WaveRNN synthesizer that is pre-trained on clean features can be fine-tuned by training it with denoised features. However, one may wonder, why not train the synthesizer from scratch on denoised features instead of fine-tuning a synthesizer pre-trained on clean features? One of the reasons is that these denoised features may still contain some spikes or traces of noise and training a randomly initialized network on such features may also not produce intelligible speech. On the other hand, initializing the weights of the synthesizer with pre-trained weights can be beneficial as the network already has learned to model speech. Moreover, it can also be claimed that the synthesizer is a completely generative model. Hence, the second approach is selected. The feature denoising and fine-tuning processes are discussed in more detail in chapters 5 and 6 respectively.

When the network is trained for 6000 epochs on the GeForce GTX 2080Ti GPU (specifications mentioned in Appendix D), the training time for this network is close to four days. Hence, in order to obtain better quality speech, the network must be trained for a significantly longer period of time. However, this is still moderate as compared to other methods such as Speech Enhancement GAN and WaveNet [12, 19] which take more than one week of training. In order to decrease training time, the sampling rate of audio was reduced to 8 kHz and this reduced the training time thus allowing experiments to be conducted faster. Details of this experiment are described in Appendix B.1.3.

4.6. Analysis

A network analysis is conducted to get some insights into its functioning and performance. This section is divided into three segments namely, computational analysis, network performance and visualizations. In the first segment, the amount of computations and memory resources required for the operations involved in synthesizing the speech samples are estimated. Secondly, the performance of the WaveRNN model is evaluated during training and testing with the Negative Log Likelihood score. Lastly, the layer components and states within the network are visualized to gain an insight into their functionality.

4.6.1. Computational Analysis

The number of operations involved in the WaveRNN are $N = 5$ matrix-vector products. This can be inferred from equation 4.1. The computational resources required for generating speech are measured in 'flops' abbreviation for the number of floating point operations. While calculating flops, the operations such as addition of the bias parameter, application of the activation function, etc. are not considered. These operations get overshadowed by the matrix-vector multiplications of the features with the network weights because they are not as intensive as the latter. The dot product between the network weights with the feature vectors are basically multiply-accumulate operations where elements in the row of the weight matrix are multiplied and summed together. So, if a weight matrix of k rows and n columns is dot multiplied with a feature vector of length n then the operations involve n multiplications and $n-1$ additions, k times. Thus, in total there are $(2n-1) \times k$ flops.

The GRU layer in the WaveRNN consists of three matrix-vector multiplications for the update gate u_t, reset gate r_t and obtaining the current hidden state e_t. As per the configurations of the network in table C.4 and equation 4.1 [2], the feature vector M_t consists of 80 Mel bands which is concatenated with a speech sample x_t thus making it a total of 81 elements in f_t. The hidden state of the GRU consists of 512 elements and thus the length of the update and reset gate vectors will also be the same. The weight matrices applied on the feature vector for the current sample f_t and the previous hidden state h_{t-1} can be concatenated and expressed as a single matrix vector product as well. Hence, the total flops for the GRU layer are $(2 \times (81 + 512 - 1)) \times 512 \times 3 \approx 1.82$ Mflops. [3] [4] Besides, the GRU layer, the WaveRNN model also consists of two fully connected layers and each contain a single matrix-vector product. From the network configurations, these operations contain 512 and 256 nodes respectively which lead to $(2 \times 512 - 1) \times 512 + (2 \times 512 - 1) \times 256 \approx 0.79$ Mflops. Hence, the total number of computations involved in generating one speech sample are approximately 2.61 Mflops. While training,

[2]Configurations from the experiment on training with clean Mel features are considered for these calculations.
[3]Mflops stands for million or Mega-flops.
[4]The estimated flops are rounded off to a higher number in the third significant digit because the Hadamard products included in the calculation of the current and final hidden states, e_t and h_t respectively, are omitted for simplicity.

the sequence length of the RNN is 1920 samples and so the number of operations in the forward calculations are approximately 5 Gflops [5] per batch.

Along with the number of computations, the memory requirements for the network weights is also an important factor. The number of parameters in the WaveRNN network for the configuration mentioned in table C.4 are 1.308 million. Since, each element is a 32 bit floating point number, the size of the weight file is $1.308M \times 4\text{bytes} \approx 5.23$ MB. As compared to other speech synthesis networks such as WaveNet [19], the WaveRNN architecture is significantly smaller. The significance of these numbers with respect to different hardware considerations is described in chapter 7.

The WaveRNN synthesizer generates speech samples at an average rate of 1.6 kHz on the GPU. Ideally, the generation rate should match with the sampling frequency of audio, i.e., 16 kHz. Thus the generation speed is slow. This can be improved by either implementing the batched sampling method and/or by pruning the network weights proposed in the original WaveRNN research [21].

4.6.2. Network Performance

The performance of the network while training is determined by the negative log likelihood (NLL) loss function (described in Appendix A section A.2.2.2) with respect to the clean reference audio sample. Figure 4.9 depicts the training loss curve with the loss expressed in bits on the vertical axis and the number of training cycles or epochs on the horizontal axis. This curve is obtained for different experiments and it is observed that irrespective of the type of feature used whether clean or noisy; they all converge close to the same point, i.e., roughly around 2.5 bits. This means that a generated sample can be represented by 2.5 bits/sample on an average.

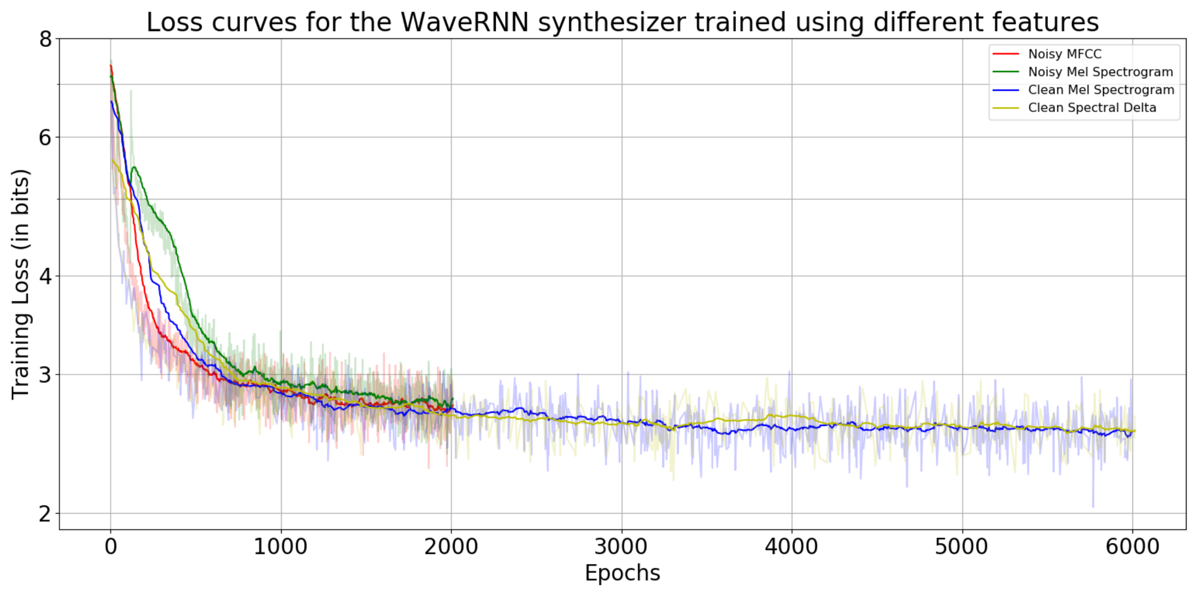


Figure 4.9: Loss curves for the experiments described in B.1.1, 4.5.1 and 4.5.2.

4.6.3. Visualizations

In order to gain some perspective about how the network functions and what are the purpose of different layers, certain components of the network are visualized.

RNN Hidden State Visualization

On generating speech from context provided by clean Mel spectrograms, the hidden state of the Gated Recurrent Unit (GRU) layer is recorded as shown in the top half of the figure 4.10. On the vertical axis lie the number of nodes in the GRU layer and the horizontal axis shows the length of the audio file in milliseconds. The bottom half of the figure displays a waveform of the reference clean speech

[5]Gflops stands for billion or Gigaflops.

whose corresponding Mel spectrogram features were used as context. It is observed that the GRU hidden state exhibits a distinct pattern during speech activity. Whereas during no speech activity, the transition across time is fairly smooth. If the variation in the GRU hidden state with time is analogized with water in a pool, then, this behaviour is analogous to ripples over still water, where the ripples indicate speech activity and the still area represents no speech activity.

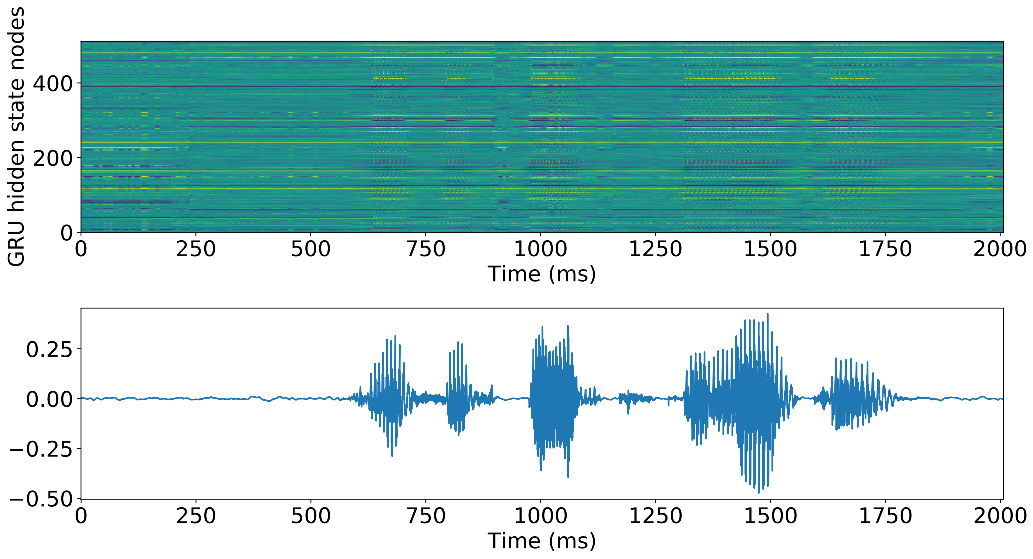


Figure 4.10: **Top**: Hidden State of the GRU obtained from clean Mel spectrogram features. **Bottom**: The clean speech waveform from which the Mel spectrogram was obtained

On zooming into a time segment with speech activity, as depicted in figure 4.11, different patterns repeating periodically are observed as highlighted in red boxes. The significance of the shape of these patterns is not known yet, however the frequency at which they repeat seems to be particularly interesting. In figure 4.11, several patterns are highlighted and the thing common between all these patterns is the frequency of their re-occurrence. A simple calculation reveals that the frequency of this repeating pattern lies within the range of the human voice pitch frequency. In the figure, all patterns repeat roughly 12 times over a segment of length 120 ms. Thus, each pattern replicates at the rate of 100 Hz which lies within the range of the pitch for a male speaker. Moreover, the example chosen contains speech narrated by a male speaker. The peaks in the speech waveform that represent the pitch frequency of a speaker are aligned with the repeating patterns in the GRU hidden state. A similar trend is also observed for female speakers.

The GRU layer can sometimes become unstable while generating speech as discussed in the results of experiment 4.5.2. The hidden state of the GRU exhibits unusual patterns in such cases as shown in figure 4.12. Here it is seen that the variation of the hidden state across the nodes for a particular time slice is much higher than the case of a stable GRU. This problem can be solved partially by lowering the learning rate after the training loss curve has converged. It is shown in experiment 6.6.3 that after this change such situations do not arise, although, it is believed that a simple hyper-parameter tweak cannot guarantee stability. Another possible solution that is inspired from [61] could alleviate such scenarios. An extra penalty function can be added to the objective function while training the network such that it enforces a restriction on the total variance of the hidden state with a user-defined variance. This solution is not implemented in this thesis, but it could be interesting to apply.

Predicted Probability Distribution & Information Rate

The softmax operation applied after the last layer of the WaveRNN network predicts a probability distribution of a sample value as shown in figure 4.1. This is a conditional probability distribution that depends on the previously predicted sample along with Mel features as context at the current index as described in section 4.2.[6] The network maps this information to a μ-law encoded 8 bit quantized

[6]This analysis was conducted for clean features as context.

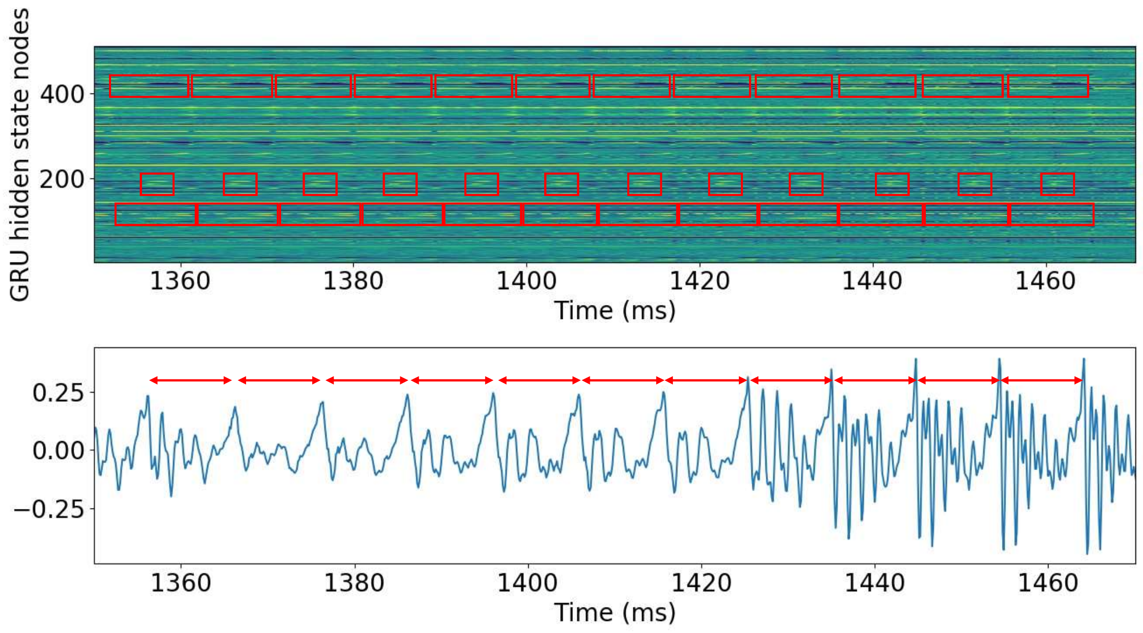


Figure 4.11: **Top**: Hidden State of the GRU for a 120 ms segment of the same example in figure 4.10. **Bottom**: Clean speech waveform for the same segment.

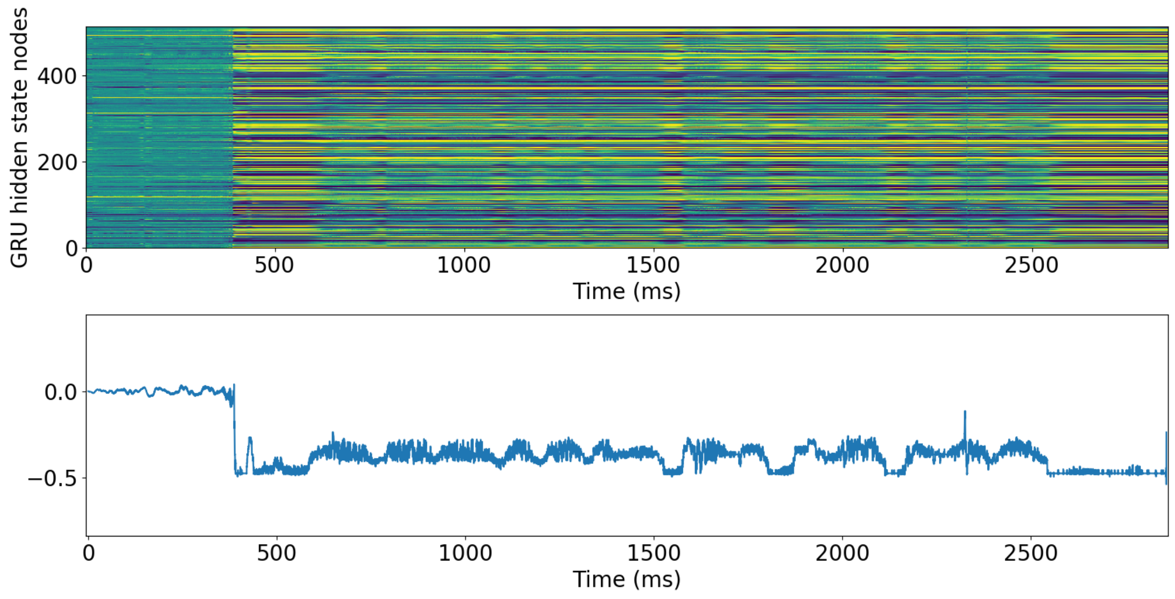


Figure 4.12: **Top**: An unstable Hidden State of the GRU layer. **Bottom**: Generated speech waveform due to instability in the GRU layer.

representation of a speech sample. The top half of figure 4.13 depicts the probability distribution of the generated sample values at every time index. The vertical axis represents the 8 bit quantized sample values ranging from $0 - 255$ and the horizontal axis stands for the time index of a sample. The value for a sample is predicted by randomly drawing a sample from the predicted probability distribution. This is how speech is generated by the WaveRNN model as shown in the bottom half of figure 4.13. The probability distribution in figure 4.13 closely represents a waveform in the log domain because of the application of μ-law encoding.

Inspired by [62], an analysis is conducted from an information theoretic point of view to understand the usage of the WaveRNN architecture as a generic generative model for speech coding. The WaveRNN

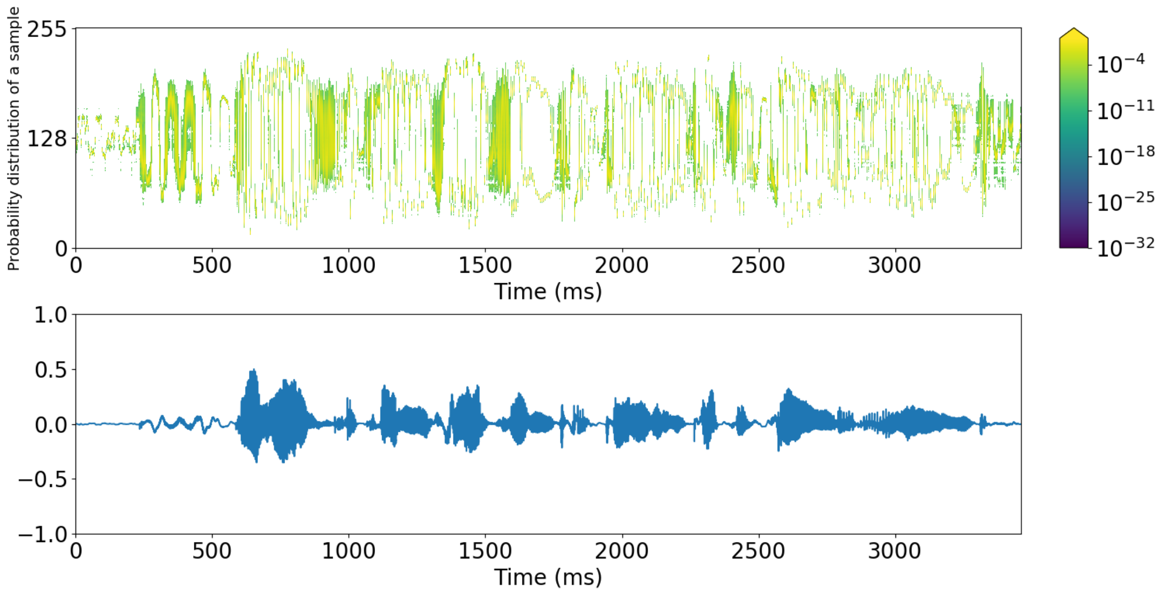


Figure 4.13: **Top**: Probability distribution of a sample value while generating speech. **Bottom**: Generated speech waveform.

encodes the conditioning variables, i.e., the Mel features and the observed waveform samples obtained from the conditional distribution. The WaveRNN can be characterized by two rates, namely, the average entropy rate and an upper bound on the average entropy rate. The average entropy rate of the estimated conditional probability distribution is given by:

$$\widetilde{H} = -\frac{1}{|\mathbb{A}_{\mathbb{O}}|}\sum_{i\in\mathbb{A}_{\mathbb{O}}}\sum_{n\in\mathbb{N}} q_n^{(i)}\log_2 q_n^{(i)}, \tag{4.3}$$

where, $\widetilde{H}$ represents the average number of bits needed to encode a sample. $\mathbb{A}_{\mathbb{O}}$ stands for the audio sequence and its length is denoted by $|\mathbb{A}_{\mathbb{O}}|$. $\mathbb{N}$ is the space to which the audio signal is mapped after applying the μ-law quantization. The conditional probability distribution at the output of the WaveRNN for sample i is denoted by $q_n^{(i)}$. The upper bound on the average entropy rate, denoted by R, is also the lower bound on the real-world rate that is produced by the WaveRNN on the observed sequence. If R and $\widetilde{H}$ are close, then it can be claimed confidently that the true average entropy rate is similar.

$$R = -\frac{1}{|\mathbb{A}_{\mathbb{O}}|}\sum_{i\in\mathbb{A}_{\mathbb{O}}} \log_2 q_{n_i}^{(i)}. \tag{4.4}$$

The average entropy rate or the uncertainty in predicting a sample value is approximately 2.5 bits/sample as per equation 4.3. This means that the network can process information at an approximate rate of 40 kbits/sec on average at a 16 kHz sampling rate. Also, the upper bound for waveform coding as per equation 4.4 is approximately 2.49 bits/sample, or 40 kbits/sec on average at a 16 kHz sampling rate. Since, the rates are similar the true average entropy rates are also similar.

Figure 4.14 displays a speech segment of 200 ms and the corresponding instantaneous information rates for the WaveRNN. The earlier part is a voiced speech segment where the required rate is low, whereas, the latter part is an unvoiced speech segment which is relatively unstructured thus requiring a higher rate. The samples constituting of voiced speech have a very sharp probability distribution thus requiring very few bits to encode (roughly 2 bits), on the other hand, samples belonging to unvoiced speech have a rather flat distribution requiring more bits to encode (roughly 6 bits). This is also visible in figure 4.13 where the probability distribution for the sample values in unvoiced speech segments is stretched more than that for voiced speech segments. Moreover, since unvoiced segments occur less frequently than voiced segments in speech the average entropy rate is lower. The rate also varies with the pitch cycle as repetitive peaks are observed in the instantaneous rate that align with the pitch frequency of the waveform.

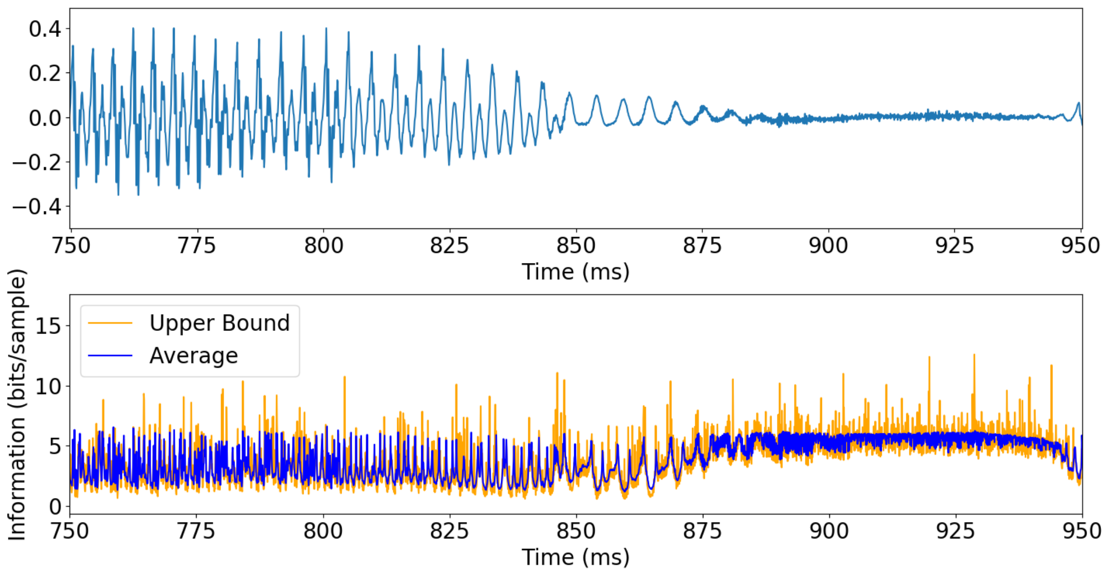


Figure 4.14: **Top**: Generated speech waveform. **Bottom**: Instantaneous information computed in terms of the average entropy rate given by equation 4.3 (Blue) and the upper bound on the rate given by equation 4.4 (Orange).

Network Weight distributions

The distribution of the learned parameters or the weights of the network are examined in this segment. Figure 4.15 shows the histogram of the weights at different layers of the WaveRNN. These histograms are plotted on the log scale along the vertical axis so that the elements with a lower occurrence in the weight matrix are also visible clearly. It is observed that the distribution of these learned weights are super-gaussian in nature. Studies have proved that speech is distributed as a super-gaussian [63] and the network has learned such a distribution on its own. In contrast to the classical methods that are designed based on the assumption of a gaussian distribution of speech, this method learns the true nature or distribution of speech.

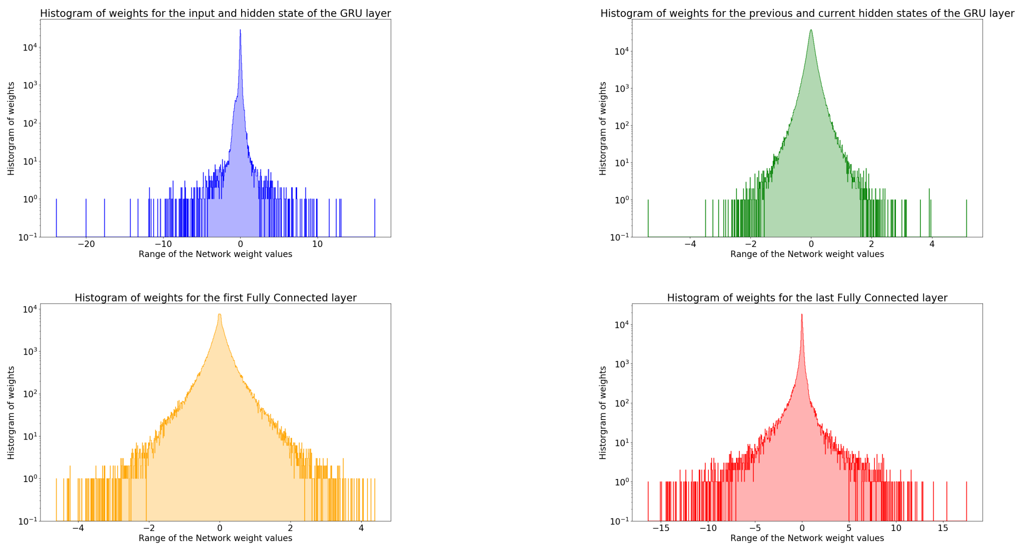


Figure 4.15: Histogram of the weights at different layers of the WaveRNN model

Another noticeable aspect is that the histogram for the weight matrices at all the layers peak at 0 which indicates that a lot of weights are either 0 or very close to 0. This means that these weights are sparse in nature. Thus the size of the weight matrices can be compressed by a method known as pruning [64]. Pruning methods are not a part of this thesis and can be considered as future work. It is important to reduce the size of the architecture and the number of parameters so that a neural network can be used on smaller and mobile hardware. Such analysis gives insights into possibilities of such measures.

Upsampling Network Kernels

As described in 4.1, the up-sampling network provides the input features to the WaveRNN synthesizer at the sampling frequency of audio. It does so by stretching the features along the time dimension, i.e., by repeating the time segments N times as indicated on the layers of the up-sampling network in the block diagram of the system in figure 4.2. Thereafter, a convolution operation is applied on these stretched features where learnable kernels of length $2 \times N + 1$ are convolved with the Mel features along the time dimension. Every layer of the up-sampling network applies the stretching and convolution operation on the Mel features which are then provided to the WaveRNN synthesizer. An analysis is performed on the up-sampling network's convolutional kernels in order to understand it's role in the system.

The kernel weights are initialized uniformly and after training is complete the learned kernels take the shape as shown in figure 4.16. The figure displays weights of the learned kernels at every layer of the up-sampling network along with their magnitude and phase responses in the frequency domain. The magnitude response of the kernel weights in all the layers indicate that the kernels act as low pass filters where they smooth out the stretched Mel features along the time axis. The phase response of all the kernels are linear which indicates that the filter simply induces a group delay without any phase distortions.

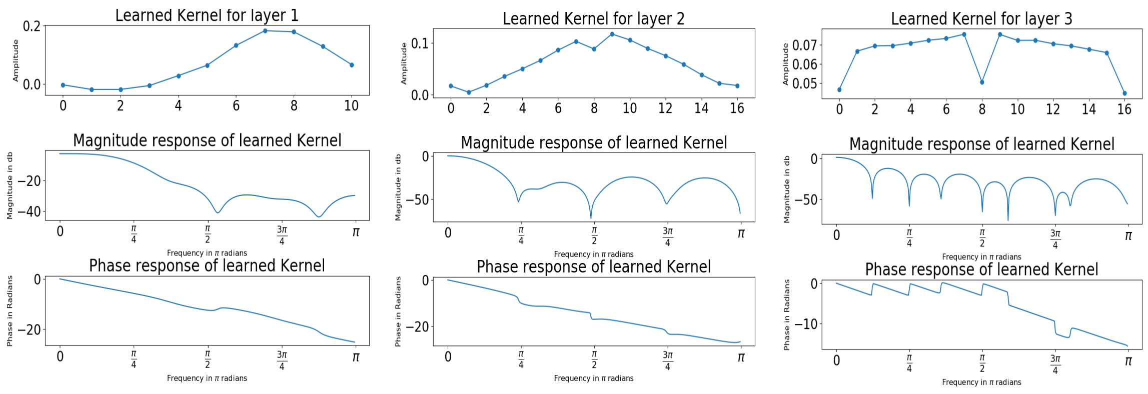


Figure 4.16: Magnitude and Frequency responses of the learnable kernels from the Up-sampling network

The up-sampling network causes delay in providing context to the synthesizer due to the convolution operation. At the moment, the up-sampling network needs to be provided with at least 3 time segments which is equivalent to 80 ms of speech as per the experiment configurations described in Appendix C table C.4. Thus, in order to provide the WaveRNN synthesizer with Mel features at the sampling rate, the up-sampling network needs future information. Note that, this does not mean that the WaveRNN needs future information for generating speech since it only depends on the context provided at the current time step and previously predicted sample. As a result, due to the up-sampling network, it is not yet possible to implement this system for real-time scenarios. One of the ways to reduce this delay would be to decrease the hop length, i.e., the amount of overlap between two short time frames for analyzing speech. Moreover, this analysis of kernel weights can help in designing FIR filters with similar magnitude and phase responses such that the delay in providing context to the synthesizer at the sampling frequency is reduced.

Concluding Remarks

The WaveRNN model, a neural network based speech synthesizer was introduced and explained in detail. This chapter discussed its training and generation procedures, experiments and analysis conducted. This section summarizes the key takeaways from these topics. Conclusions from this chapter form the baseline for the subsequent chapters. The system has certain noteworthy advantages and also has some shortcomings which are presented here.

The following inferences are drawn from this chapter:

- WaveRNN is a fully generative network that models audio as a conditional probability distribution depending on the previous audio samples and the context provided to it.
- As far as the context is clean, the quality and intelligibility of generated speech is extremely high. It is almost indistinguishable from the reference speech. This is irrespective of the type of features used as context.
- The presence of noise in the context does not affect the quality of the synthesized speech as it sounds very natural. However, the generated speech lacks intelligibility due to mumbled words and difficulty in reproducing consonant sounds.
- Simply tuning network hyper-parameters is not the key to improve the model performance and may not alleviate the problem when noisy context is provided. A good understanding of input features is required.
- Separately denoising or enhancing the features before they are provided to the WaveRNN synthesizer seems to be a more practical solution. Moreover, instead of training the synthesizer ground up on denoised features as context, it would be beneficial to fine-tune the WaveRNN that is pre-trained on clean context. This is inspired from a principle called 'Transfer Learning'.
- Hidden state in the Gated Recurrent Unit is capable of capturing the voice identity of a speaker since repetitive patterns that are synchronous with the pitch frequency of the speaker are observed.
- The weights of the network follow a super Gaussian distribution which is similar to the distribution of speech. Hence, the network learns the natural distribution of speech unlike classical approaches that rely on a Gaussian assumption to analytically derive their models.

The advantages of the WaveRNN synthesizer are as follows:

- Auto-regressive model that is capable of processing streaming audio instead of processing blocks of samples.
- For the superior quality of generated speech (from clean context), the size of this model and the number of computations involved is extremely small as compared to other neural network based speech synthesis models such as WaveNet [19] and SampleRNN [41].
- So far no analytically derived or statistical model can synthesize speech from Mel features. This neural network model learns the transformation from the Mel domain to speech samples.

The disadvantages of the WaveRNN model are as follows:

- Generated speech can contain artifacts sometimes due to instability in GRU layer. The occurrence of such artifacts is unpredictable.
- At the moment, the audio generation time is slow. It is 10 times lesser than the sampling frequency of audio. This can be improved by pruning the network weights which in turn reduces the processing time, but it is not within the scope of this research and can be pursued in the future.

5
Denoising Features

This chapter outlines the process of denoising the noisy features before they are provided to the WaveRNN for synthesizing the waveform. From chapter 4, it was inferred that noisy features in general do not provide informative context for generation of clean speech. Speech corrupted by different types of noises makes the spectral structure in speech disordered hence rendering features ineffectual. It is difficult for the WaveRNN architecture to learn the manifold of speech when the context provided itself is corrupted with noise of varied characteristics. As a result there is a need to first remove the noise from these features before they can be synthesized to a waveform.

The organization of this chapter is as follows, where, first the design of two denoising architectures, namely, the U-Net and Variance Constrained Auto-Encoder are described in section 5.1.1 and 5.1.2 respectively. This is followed by the training and feature denoising processes in sections 5.2 and 5.3 respectively. The configuration and hyper-parameter definitions of both architectures are highlighted in section 5.4. Later, experiments performed with different architectural changes and results obtained are discussed in section 5.5. Thereafter, an analysis reported in section 5.6 contains a study on the network's performance with respect to objective metrics. It also contains visualizations of intermediate feature representations which provide an understanding about the network's operations. In the end, concluding remarks are mentioned to elucidate the key findings in this chapter.

5.1. System Design

For the denoising task two different architectures are proposed. The following subsections describe the architecture design in detail and the motivation behind those choices. It will also throw light on reasoning out the usage of specific components while designing the system.

5.1.1. U-Net

The first choice for the denoising architecture was inspired from a U-net [65]. Originally, the U-net architecture was proposed for solving the biomedical image segmentation problem. Later, this U-net architecture was also adopted in the Generator module of the Speech Enhancement GAN with some modifications [12]. The main characteristics of the U-Net architecture that makes it attractive are the Encoder and Decoder modules which maps the input to an intermediate space and then back again to the original space. This intermediate space is compressed in terms of resolution of the spatial dimensions but may not be an overall compressed representation. This is achieved with a fully convolutional neural network as decribed in chapter A segment A.2.5. The network also consists of skip connections from the encoder layers to the decoder layers. This prevents deep networks from the vanishing gradients problem by providing an alternate path for the gradients to flow during backpropagation. However, the use of skip connections is also arguable as the hidden units can also tend to get pruned because of the enforced linearity due to an identity mapping. The proposed architecture captures these key aspects from the U-Net along with some modifications described in the following paragraph.

The proposed U-Net architecture illustrated in the left half of figure 5.1 consists of a fully convolutional encoder-decoder structure with skip connections. The encoder is created by seqeuntially

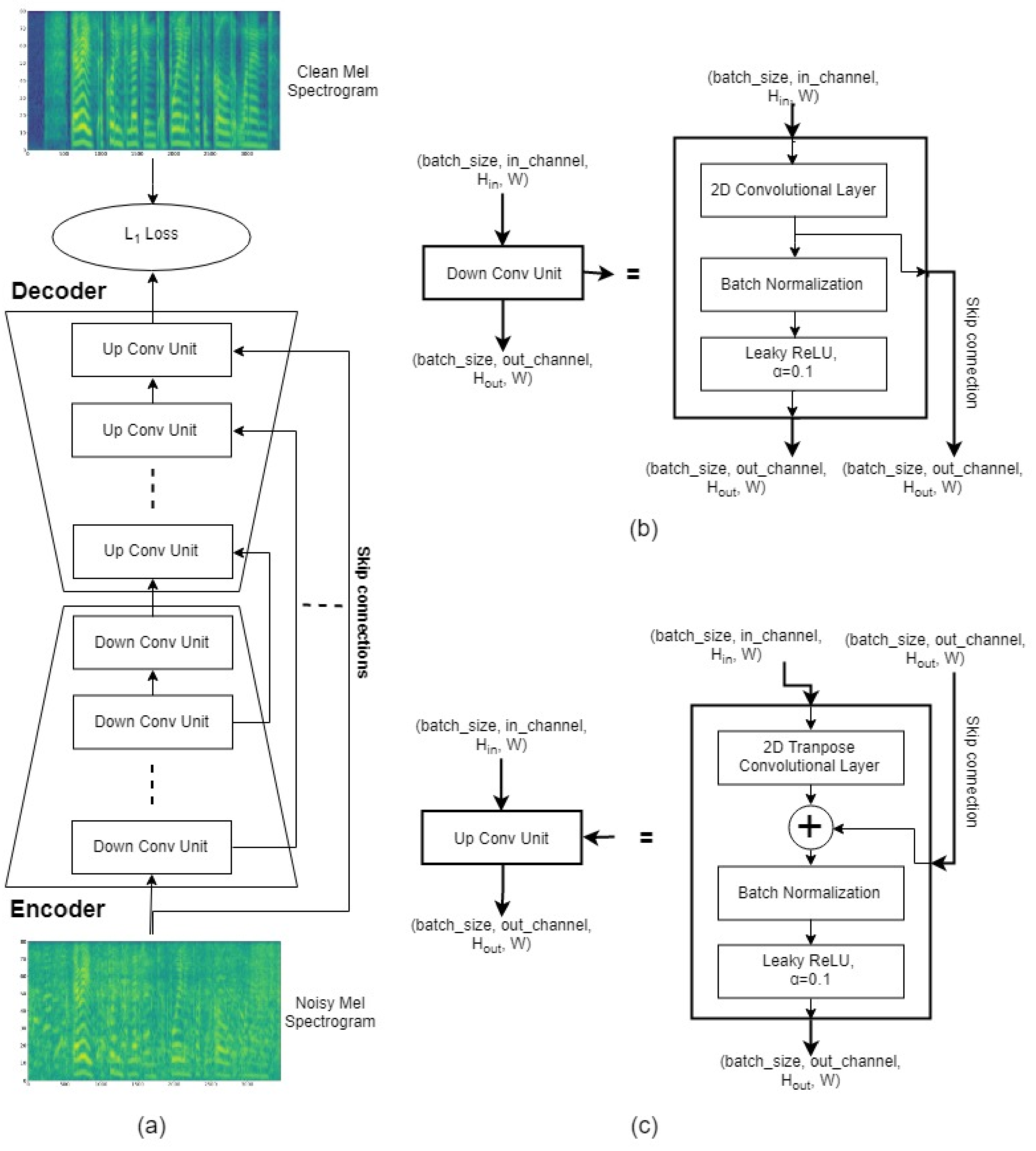


Figure 5.1: Denoising Architecture based on the U-Net design

stacking the sub-module named 'Down Conv Unit' that consists of a 2D convolutional layer and a batch normalization layer followed by a leaky ReLU non-linearity as shown in the top right of figure 5.1. Since, the input to the denoising architecture are Mel features, the learnable kernel in the convolutional layers are designed in such a manner that they span more samples along the Mel direction and less along the time dimension. Furthermore, by tweaking hyper-parameters such as the stride, padding and kernel size, the convolution operation and downsampling of the output size can be controlled. Similar to the encoder structure, the decoder is also designed by sequentially stacking the sub-module named 'Up conv unit' that consist of a 2D transposed convolutional layer and a batch normalization layer followed by a leaky ReLU non-linearity as shown in the bottom right of figure 5.1. As the name suggests, this sub-module is responsible for upsampling the input features to a higher resolution space. Skip connections are drawn from the input to the encoder and from every 'Down Conv Unit' (except the last unit) in the encoder to every 'Up Conv Unit' in the decoder as shown in the left half of figure 5.1. Because of this, the hyper-parameter settings such as stride, padding and kernel size of the transposed convolution in the decoding stage have to be the same as that of the convolution in the encoding stage. As a result of which the output size of the 'Up Conv Unit' is the same as the output size of the 'Down Conv Unit' but their inputs have different dimensions. In the 'Down Conv Unit', these skip connections are added to the upsampled features (like in ResNets) instead of being concatenated as depicted in SE-GAN [12] and U-Net [65]. The presence of batch normalization layers in both sub-modules makes the network training stable as it normalizes the output features at every layer. Additionally, they provide regularization effects preventing overfitting, ReLU type non-linearities are used because they are known to perform well for speech processing [66] as compared to other non-linearities. Moreover, Leaky ReLU is preferred over ReLU as it also allows the flow of negative values unlike ReLU which could pose a hindrance for regression problems.

5.1.2. Variance Constrained Auto-Encoder

The idea behind the second choice for the denoising architecture is inspired from Auto-encoders. Auto-encoder are widely used for denoising applications [67]. The main difference between the U-Net and Auto-Encoder architectures is that Auto-Encoders have a bottleneck structure at the end of the encoder, i.e., they squeeze the input space to a compressed representation and the decoder maps it back to the original space. Since the last few years, a lot of research was conducted on Variational Auto-Encoders (VAE) [17]. They have also been applied for the task of speech enhancement in the frequency domain [68, 69] . VAE's seem to be a natural choice because of its encoder-decoder structure that maps the noisy input features of speech to a latent space which is then used to generate clean speech. However, recent study has also shown that VAE does not necessarily learn a meaningful latent representation [61, 70] , i.e., the input features are independent of the latent representation. This is because, the output of a VAE is obtained by sampling from a Gaussian distribution whose parameters are governed by the latent space. This distribution is of a fixed shape and not completely representative of the input and hence, it may fail to correctly capture the speech manifold of the input to the desired output. This in turn, means that the desired speech content cannot be controlled. To counter this, the Variance Constrained Auto-Encoder (VCAE) was proposed because this network does not enforce a pre-defined prior on the distribution over latent encoding thus allowing the structure of the latent features to represent data better.

The architecture of this network is simple and similar to the U-Net in many aspects as shown in the left half of figure 5.2. For instance, the sub-module's in the encoder and decoder structures are quite similar to that in the U-Net architecture (except the skip connections) and depicted in the right half of figure 5.2. The encoder and decoder for the Variance constrained Auto-Encoder are also structured alike, i.e., by sequentially stacking their respective sub-modules. However, there are some stark differences as well that must be kept in mind. To start with, the existence of skip connections in auto-encoders is contradictory to the fundamental principle of using a bottleneck, so skip connections are omitted here. The bottleneck structure can be created by either using dense fully connected layers or using convolutional layers (by limiting the number of channels). The total number of nodes in the bottleneck layer are lesser than the input nodes whereas this is not the case with the U-Net model. This ensures that the encoder network captures the manifold of speech related features in the latent space and projects them further to the output with the decoder network.

The encoder maps the input to a latent representation, z, which ideally should be a manifold of the features corresponding to speech. This representation is discrete in nature and in order to make it smooth, noise is added to it as illustrated in figure 5.2. This is an attempt to make the manifold a bit more continuous and introduce a stochastic nature. Samples from a Gaussian distribution with 0 mean and a desired standard deviation σ are added to the compressed latent space, z[1]. The overall variance of z would then at least be its own variance added with the variance of the noise, σ, times the dimensionality of z (assuming noise is independent from the latent representation, however this may not be completely true). In a Variance Constrained Auto-Encoder, the variance of this latent representation is bounded by defining a penalty function in the objective function. Unlike, VAE where

[1]Samples from any type of user-defined distribution with a fixed variance can be added to the latent space z. Here, a Gaussian distribution is used not only because it is independent and identically distributed but also fairly simple to implement.

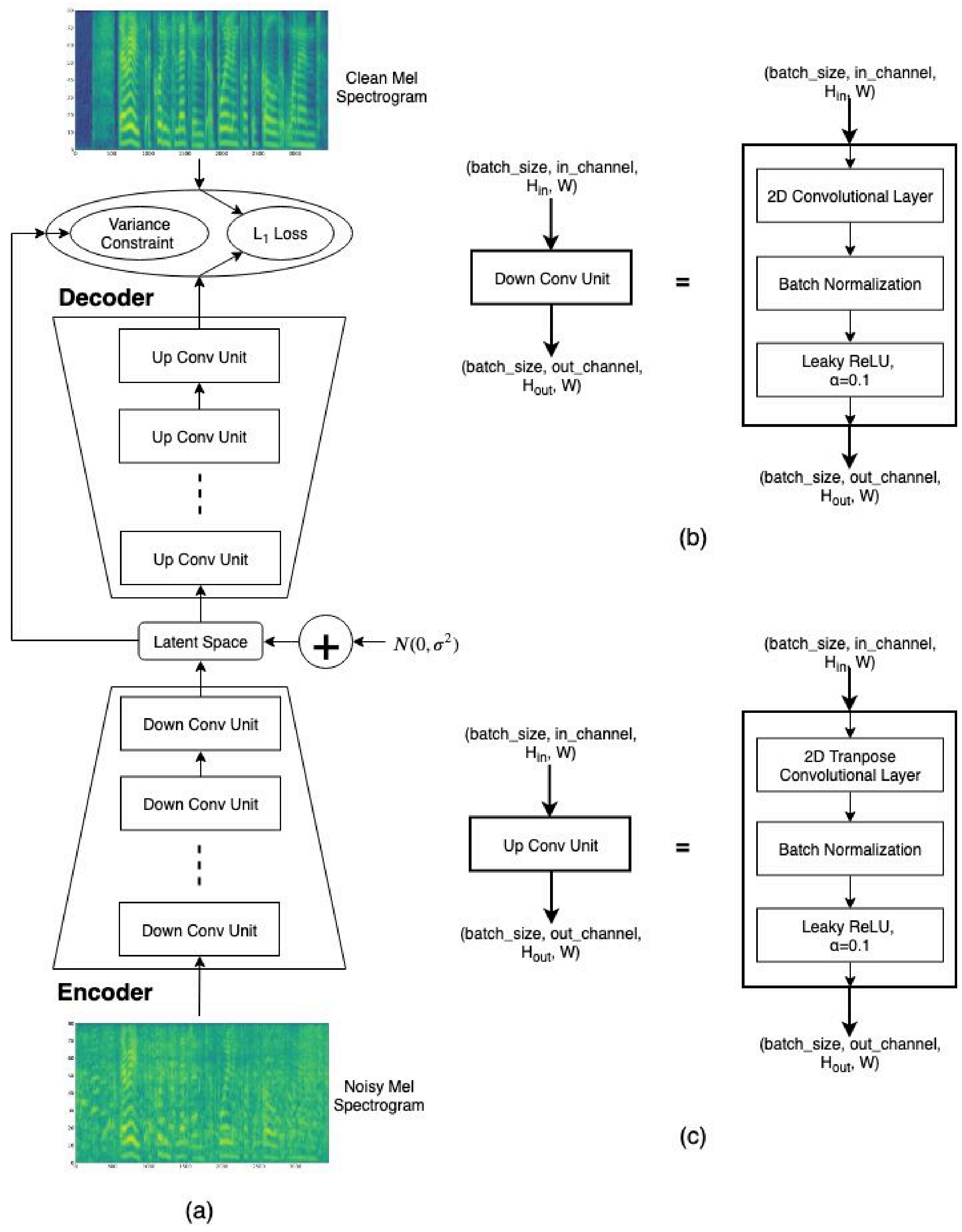


Figure 5.2: Denoising Architecture based on the Variance Constrained Auto-Encoder

samples are drawn from a latent space of a prior fixed shape, in VCAE, the shape of the latent space that is derived from the input features is constrained to have a fixed variance.

It is known that batch normalization encourages training stability and acts as a regularizer, however, it is also hypothesized that the constraint on the variance of the latent space is introduced to serve the same purpose. Batch normalization layers could co-exist while still applying the variance constraint regularization as depicted in figure 5.2. However, to verify this hypothesis, an experiment was conducted by removing the batch normalization layers. The use of leaky ReLU is preferred over other non-linearities as they are better suited for regression based problems as described in segment 5.1.1.

5.2. Training

The denoising system is a regression problem where the network tries to map the noisy features to the clean features. Before delving into the training process for both approaches, it is better to describe the data loading procedure as it is the same for both. A collate function gathers the noisy and clean Mel features from the training dataset and selects a random frame of arbitrary length along the time dimension with all Mel bands. The noisy features are fed into the network and the desired output are the clean features as shown in figures 5.1 and 5.2. After, the data is loaded in batches and presented to the network the training begins.

U-Net

The U-Net architecture is trained by minimizing the L_1 distance, also called, the mean absolute error between the network output and the desired clean Mel features. The objective function can be expressed as follows: ,

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}_{X\sim P_D}[||X - f_\theta(\tilde{X})||_1] \tag{5.1a}$$

where, X is a random variable that stands for clean Mel features and $\tilde{X}$ is a random variable representing the noisy Mel features, i.e., the input to the network, f_θ where, θ are the trainable parameters. The clean features X follow the data distribution denoted by P_D. Thus, the feature enhancement problem can be formulated as learning a mapping to features X given the noisy features $\tilde{X}$ and network parameters θ.

Many regression based speech enhancement solutions rely on the L_2 criterion [2, 8–10]. The Wiener filter is also designed to be optimal in the L_2 sense. For a Gaussian assumption, minimizing the L_2 error is equivalent to maximizing the log likelihood of the output data. However, speech and its features are super-gaussian distributed or, in other words, follow a factored Laplace distribution and so maximizing the log likelihood would be similar to minimizing an L_1 criterion. Moreover, selecting the L_1 criterion as the objective function over the L_2 norm is advantageous because the L_2 severely penalizes the higher energy samples and ends up fitting them correctly, seldom catering to map the lower energy samples. This leads to artifacts and distortions. The L_2 on the other hand leads to a rather smoother mapping in that regard. The network parameter, θ, that include the convolution kernel and batch normalization parameters are updated by backpropagating the weights through the network. The ADAM optimizer is used to minimize the loss function [59]. The hyper-parameters for this network are discussed in section 5.4 table 5.1.

Variance Constrained Auto-Encoder

The Variance Constrained Auto-Encoder network was originally proposed as a generative model, where the output of the decoder, i.e., the enhanced speech, was provided to a discriminative network that minimized the Wasserstein distance between the distribution of the enhanced speech and that of the clean speech [22, 61]. However, there was no control over the content of generated speech signals by minimizing this Wasserstein distance alone [22]. Consequently, an additional L_1 error between the output of the decoder and the desired reference speech was also minimized thus rendering the approach as not fully generative.

In this implementation, the VCAE architecture is not generative and only the L_1 criterion between the decoder output and reference clean features is minimized. Additionally, a constraint on the variance of the latent space is added as a penalty to the L_1 loss. The overall objecive function is expressed as:

,

$$\underset{\theta,\phi}{\text{minimize}} \quad \mathbb{E}_{X\sim P_D}\mathbb{E}_{z\sim Q_{z|\tilde{X};\theta}}[||X - f_\phi(z)||_1] - \lambda||\mathbb{E}_{z\sim Q_{z;\phi}}[||z - \mathbb{E}_{z\sim Q_{z;\phi}}[z]||_2^2] - v||_1 \tag{5.2a}$$

where, X and $\widetilde{X}$ are the random variables representing the clean and noisy Mel features. The network f consists of encoder and decoder structures with parameters θ and ϕ respectively. The Encoder, f_θ, maps the input $\widetilde{X}$ to a latent representation z which follows a distribution $Q_{z|\widetilde{X};\theta}$. Thereafter, the decoder, f_ϕ, maps the latent space to the clean Mel features X which follows the data distribution denoted by P_D. The first term of the objective function represents the L_1 criterion between the decoder network output and the reference. The second term is the penalty function that bounds the variance of the latent space with a desired total variance v (summed across each dimension in the latent space). The hyper-parameter λ controls the trade-off between enforcing the constraint and minimizing the L_1 loss. The distribution of the latent space, $Q_{z;\phi}$, also includes the Gaussian noise $\mathcal{N}(0, \sigma^2)$ added to it. This constraint on the variance controls the information flow of the noisy speech features, $\widetilde{X}$, in the latent space, z, which in turn controls the content of speech in the output features, X. Thus, the feature enhancement problem can be formulated as learning a mapping to clean features X from a compressed state z that is derived from noisy features $\widetilde{X}$ given the network parameters θ and ϕ.

The network parameters, θ and ϕ, that include the convolution kernels for the encoder and decoder networks and batch normalization parameters in every layer are updated by backpropagating the weights through the network. The ADAM optimizer is used to minimize the objective function [59]. The hyper-parameters for this network are discussed in section 5.4 table 5.2.

5.3. Estimating Denoised Features

Once, the networks are trained for the denoising task, the testing or denoised feature estimation process is exactly the same for both the U-Net and VCAE network designs. The noisy Mel features from the test dataset or extracted from any noisy audio file are presented to the model. The trained weight files are loaded into the model and applied to the noisy features to obtain the denoised Mel features. After denoising, these features can be used as context for the WaveRNN synthesizer which is described in chapter 6.

5.4. Setup

Network hyper-parameter definitions and configurations for both the U-Net based design and VCAE system is mentioned in this section. The following tables 5.1 and 5.2 enlist the parameters used in the network, their setup and some remarks for the U-net and VCAE methods respectively.

Table 5.1: Setup for the U-Net based Denoising Architecture

	Parameter name	Format	Remarks
Network Hyper-parameters			
	Convolution Kernel size	$[k_f, k_t]$	First element indicates size along the Frequency axis and second element is the size along the Time axis
	Encoder Channels	$[C_1, C_2, ..., C_n]$	The first element indicates the number of input channels to the encoder. Rest indicate the output channels for every layer in the encoder. Number of layers in the Encoder in this case is $n-1$.
	Decoder Channels	$[C_n, C_{n-1}, ..., C_1]$	The decoder channels may or may not be the exact reverse of the encoder channels. The first element indicates the number of input channels to the decoder. Rest indicate the output channels for every layer in the encoder. Number of layers in the Decoder in this case is also $n-1$ and may or may not be equal to number of layers in the Encoder.
	Stride	$[s_f, s_t]$	First element indicates slide along the Frequency axis and second element is the slide along the Time axis
	Padding	$[p_f, p_t]$	First element indicates padding along the Frequency axis and second element is the padding along the Time axis
	Batch size	b	Batch size of data loaded into the model
	Epochs	e	Training cycles
	Learning rate	lr	Learning rate for the optimizer
Feature configurations			
	Feature		Type of feature, Mel Spectrograms, MFCC, Spectral Delta
	Mel bands	M	Number of Mel frequency bands in the feature
	Mel Window	w	Number of time segments in the feature
	Sampling frequency	sr	
	Window length		Number of samples in a short time frame of audio.
	Hop length		Number of overlapping samples. 50% overlap considered.
	FFT size	N	N-point DFT is computed on every short time frame.

Table 5.2: Setup for the VCAE based Denoising Architecture

	Parameter name	Format	Remarks
Network Hyper-parameters			
	Convolution Kernel size	$[k_f, k_t]$	First element indicates size along the Frequency axis and second element is the size along the Time axis
	Encoder Channels	$[C_1, C_2, ..., C_n]$	The first element indicates the number of input channels to the encoder. Rest indicate the output channels for every layer in the encoder. Number of layers in the Encoder in this case is $n-1$.
	Decoder Channels	$[C_n, C_{n-1}, ..., C_1]$	The decoder channels may or may not be the exact reverse of the encoder channels. The first element indicates the number of input channels to the decoder. Rest indicate the output channels for every layer in the encoder. Number of layers in the Decoder in this case is also $n-1$ and may or may not be equal to number of layers in the Encoder.
	Stride	$[s_f, s_t]$	First element indicates slide along the Frequency axis and second element is the slide along the Time axis
	Padding	$[p_f, p_t]$	First element indicates padding along the Frequency axis and second element is the padding along the Time axis
	Type of Bottleneck		Two types of bottleneck structures namely dense and conv that stand for fully connected network and convolutional network respectively.
	Dense Bottleneck layers	$[n_1, ..., n_L, ..., n1]$	First element is the product of the size of the output of the encoder. It is the flattened representation of the output features. It is calculated as follows: $(M/(s_f * (n-1))) * w * C_n$ Middle element n_L represents the number of nodes in the bottleneck layer, i.e., the latent space First element and the last element should always be the same.
	Variance constraint	v	Constraint on the variance of the latent space
	Standard deviation of noise	σ	Noise added to the bottleneck layer is Gaussian noise with 0 mean and sigma standard deviation
	Batch size	b	Batch size of data loaded into the model
	Epochs	e	Training cycles
	Learning rate	lr	Learning rate for the optimizer
Feature configurations			
	Feature		Type of feature, Mel Spectrograms, MFCC, Spectral Delta
	Mel bands	M	Number of Mel frequency bands in the feature
	Mel Window	w	Number of time segments in the feature
	Sampling frequency	sr	
	Window length		Number of samples in a short time frame of audio.
	Hop length		Number of overlapping samples. 50% overlap considered.
	FFT size	N	N-point DFT is computed on every short time frame.

5.5. Experiments & Results

Experiments were performed by tweaking certain configurations described in section 5.4. The motivation and aim for each experiment is stated in their respective segments. Results obtained from the respective experiments are also exhibited. The general architecture displayed in figures 5.1 and 5.2 remains the same while certain network hyper-parameters and feature configurations were changed. The feature estimation and training procedure remains the same for both the networks as described in sections 5.3 and 5.2 respectively. Note that, all denoising experiments were performed on features extracted from 8kHz speech samples because the denoised features were further used for fine-tuning the synthesizer that was pre-trained for 8kHz speech and features. Also, the network architectures mentioned in sections 5.1.1 and 5.1.2 are capable of denoising all Mel features, i.e., Mel spectrograms, MFCCs and Delta features, however, these experiments only discuss Mel spectrograms. In this section, only a few experiments that align well with the narrative in this chapter are mentioned. However, some extra experiments that also led to important findings are mentioned in Appendix B.

5.5.1. U-Net with 2D strided convolutions

A few experiments had to be conducted before the architecture described in this segment was proposed. These are highlighted in appendix B.2.1 and B.2.2. In those previous experiments, the fundamental properties of an encoder and decoder structure were not fully utilized. This is because the features obtained at the last layer of the encoder, i.e., the intermediate representation had not been correctly compressed in the spatial domain. Hence, they could not precisely capture the true essence or the manifold of speech related information. To achieve the reduction in spatial resolution, the use of strided convolutions with zero padding is proposed here. Now, instead of the kernel shifting by one sample, it shifts by the amount of stride specified for every layer. The stride is applied only along the frequency dimension in order to capture the spectral information along bands. This leads to a compression along the frequency dimension and the number of time segments are preserved. Note that, the intermediate space obtained at the last layer of the encoder is not an overall compressed representation because the number of channels is large. Figure 5.3 depicts the information flow through the network.

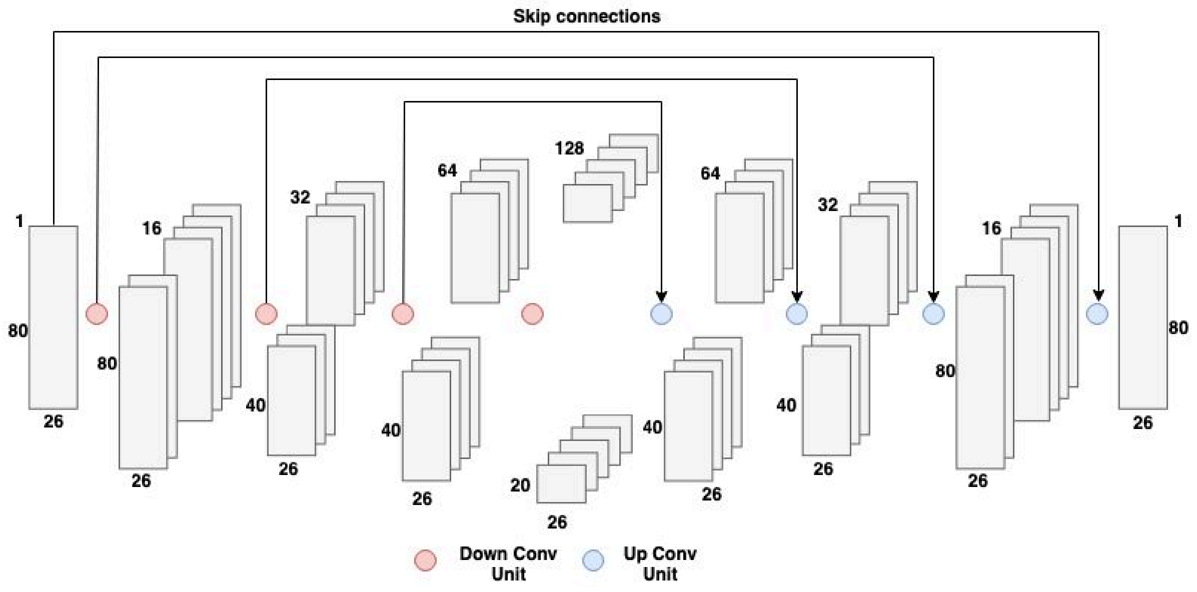


Figure 5.3: Information flow in the U-Net architecture

Additionally, through previous experimentation in B.2.1 and B.2.2, it was observed that the network faced problems in mapping especially the high frequency contents. Most high frequency information has relatively lower energy, as a result, they get heavily suppressed by the addition of noise. Moreover, since regression type losses such as L_1 norm end up mapping high energy content more precisely (since they induce higher loss if not correctly mapped) as compared to low energy samples, the higher frequency information remains under estimated. Hence, it is difficult to retrieve this information. To solve this problem, the clean and noisy speech files were pre-emphasized (as described in chapter 3 section 3.2) before extracting the Mel spectrogram features. It is hypothesized that because of pre-emphasized clean speech features the high frequency information will also be effectively retrieved. The normalization technique was also changed where instead of the traditional min-max normalization, the sigmoidal normalization was applied as explained in chapter 3 section 3.7. The configuration for this experiment is mentioned in table C.8.

After training the U-Net model, Mel spectrograms extracted from the noisy speech in the test set are provided to this system. The U-Net model estimates the denoised features as described in section 5.3. The results obtained are illustrated in figure 5.4, where the Mel spectrograms of the clean and noisy speech along with their estimated denoised version are shown. Both male and female speaker examples and their contamination with two different types of noises at the most extreme noise level in the test dataset are analyzed. The following observations were made:

1. The estimated denoised Mel spectrogram appears to be free from noise characteristics. However, there are some difficulties in removing Cafe noise as some speech-like harmonics do appear in the denoised Mel spectrogram during no speech activity.
2. On comparing with clean Mel spectrogram, the power in denoised Mel spectrogram during non speech activity does not go as low.
3. The use of 2D kernels in the convolution operation leads to smoother transitions especially along the time dimension.
4. Speech harmonics in the higher frequency regions are retrieved fairly well. However, in extreme noise cases, the harmonics are blurred as the power is averaged out across neighbouring bands.

Based on these observations, certain inferences were made. Since, the U-Net architecture consists of skip connections and the overall dimension of the intermediate space is more than the input, it is

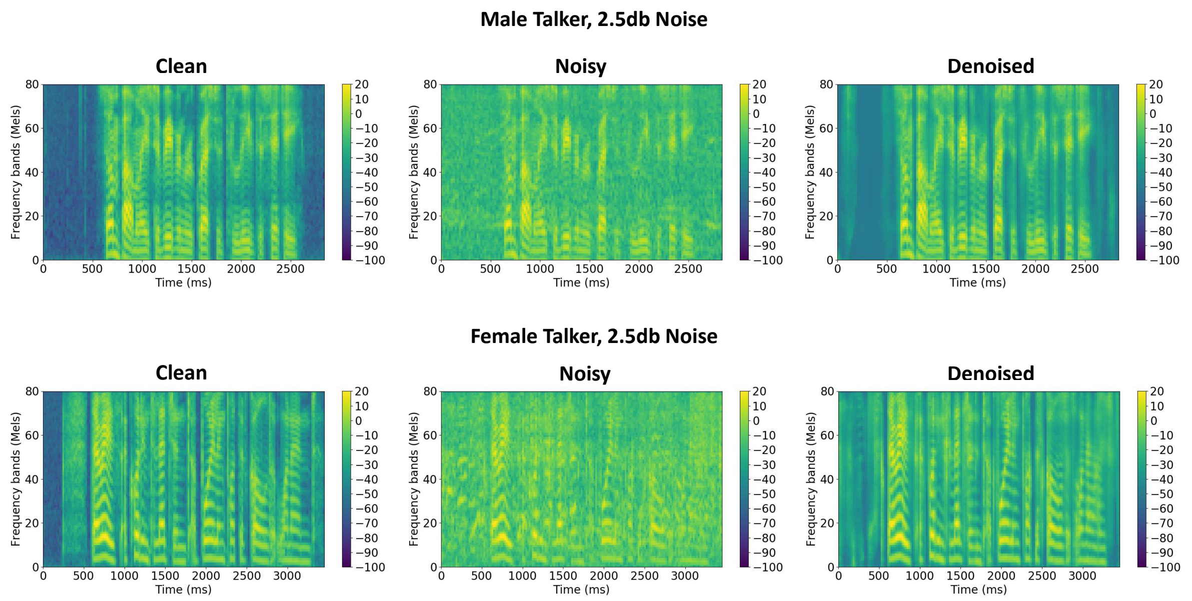


Figure 5.4: **Top Left**: Mel spectrogram of clean speech for a male talker. **Top Middle**: Mel spectrogram for speech corrupted with Living noise at 2.5 dB SNR. **Top right**: Denoised Mel spectrogram of the example (output of the network). **Bottom Left**: Mel spectrogram of clean speech for female talker. **Bottom Middle**: Mel spectrogram for speech corrupted by Cafe noise at 2.5 dB SNR. **Bottom right**: Denoised Mel spectrogram of the example (output of the network).

capable of preserving some minor details. The presence of Cafe noise during non speech activity could occur because Cafe noise has certain speech like characteristics which could have been transferred to the output. There could be two possible reasons for the power level during non speech activity in the denoised Mel spectrogram to not go as low as that in clean Mel spectrogram. First is due to the L_1 regression loss which fails to map low energy values precisely as they do not incur heavy penalties while training. Moreover, some clean speech examples in the dataset have breathing sounds during non speech activity which have relatively higher power than silence. For example, this can be seen in the bottom left image of figure 5.4. Because the network attempts to generalize to the features, the noise power in the input noisy features during non speech activity is mapped to the power level of breathing sounds. The occurrence of speech harmonics in the high frequency bands is because of pre-emphasizing the waveform before extracting features. However, in extreme noisy cases, this structure is lost and becomes very difficult to retrieve. Hence, the best possible estimate is to smooth out the power across the frequency bands.

The model performance and results are analyzed further in section 5.6. The computational and memory resources required by this model are estimated in segment 5.6.1. With the help of evaluation metrics, a comparison of the results with that of other methods is drawn in segment 5.6.2.

5.5.2. VCAE: Effect of using a Dense Bottleneck

After some considerable improvement in results with the U-Net architecture, it was time to test the network with an actual bottleneck. The U-Net does not compress the input features to a condensed space like that seen in auto-encoders. The Variance constrained Auto-Encoder was implemented for denoising noisy Mel spectrograms as described in section 5.2. An extra bottleneck network structure is inserted in between the encoder and the decoder which compresses the encoder features to a latent space. This bottleneck structure is constructed using a dense fully connected network. Hence, the goal of this experiment it to study the performance of a Variance constrained Auto-encoder with a dense bottleneck for the task of feature denoising. The encoder and decoder structure remains similar to that of the U-Net. The output at the encoder is flattened and the number of nodes in this flattened representation equals the number of channels times the number of time segments times the number of bands along the frequency dimension. This flattened layer is mapped to a latent space with lesser number of nodes constituting the bottleneck. Gaussian noise with a small variance of $\sigma = 0.02$ is added in order to make this space continuous. The variance of noise is user defined and its purpose is to smooth the latent space. The network then transforms this space back to the same dimensions as the output of the encoder and provides it to the decoder. The operations in this network are depicted in figure 5.5 and the configurations are mentioned in table C.9.

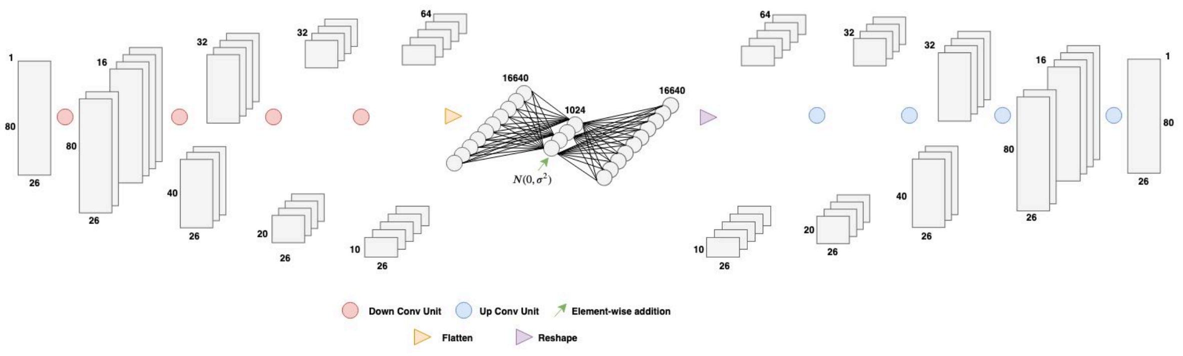


Figure 5.5: Information flow in the VCAE architecture with Dense bottleneck

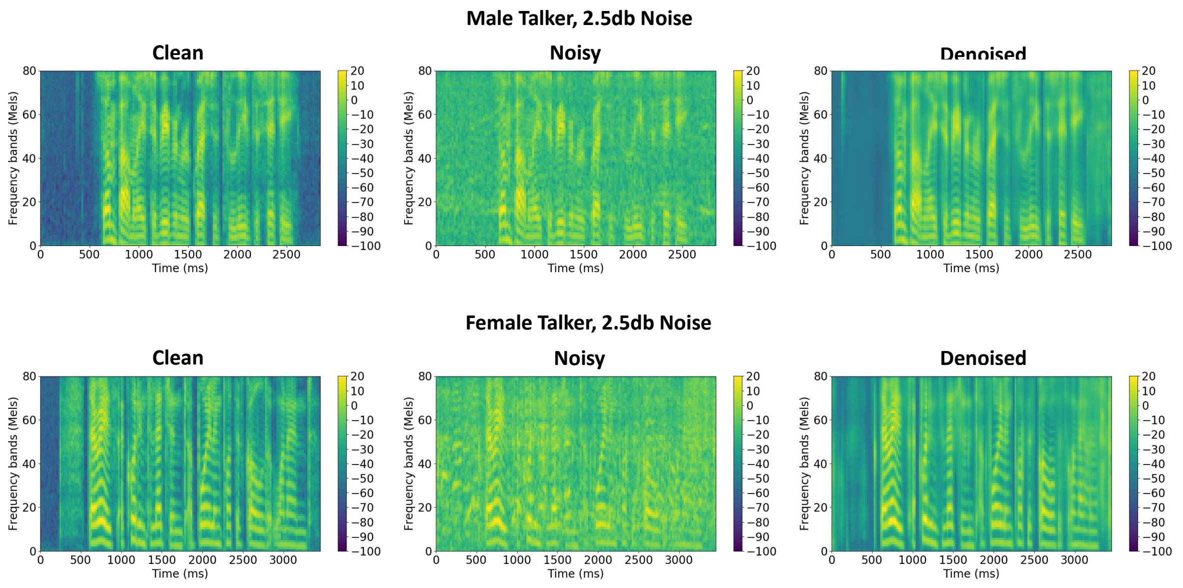


Figure 5.6: **Top Left**: Mel spectrogram of clean speech for a male talker. **Top Middle**: Mel spectrogram for speech corrupted with Living noise at 2.5 dB SNR. **Top right**: Denoised Mel spectrogram of the example (output of the network). **Bottom Left**: Mel spectrogram of clean speech for female talker. **Bottom Middle**: Mel spectrogram for speech corrupted by Cafe noise at 2.5 dB SNR. **Bottom right**: Denoised Mel spectrogram of the example (output of the network).

After training the VCAE model for denoising, mel spectrograms extracted from the noisy speech in the test set are provided to this system. The VCAE model estimates the denoised features as described in section 5.3. The results obtained are illustrated in figure 5.6, where the Mel spectrograms of the clean and noisy speech along with their estimated denoised version are shown. Both male and female speaker cases and contamination with two different types of noises at the same noise power level are analyzed. The following observations were made:

1. The estimated denoised Mel spectrogram appears to be free from noise characteristics even in the case of Cafe noise.
2. The power in the denoised Mel spectrogram during non speech activity does not go as low when compared with the power during non speech activity in the reference Mel spectrogram.

3. The harmonics in speech in the mid and especially higher frequency regions are lost and the power is averaged out with the surrounding frequency bands.

Most of the observations made here are similar to that in experiment 5.5.1. The reasons for such observed behaviour are sought and some observations are made. Since, the VCAE architecture has no skip connections and on top of that it consists of a fully connected bottleneck network structure, it allows sufficient information to pass through but many details are missing. As a result, the harmonic structure in speech also looks blurred in the denoised Mel spectrograms shown in figure 5.6 especially in the mid and high frequency bands despite of using pre-emphasized features. The reasons for a relatively higher power in the denoised Mel spectrogram during non speech activity as compared to the reference clean Mel spectrogram is due to the use of L_1 criterion while training. Moreover, it is suspected that the generalization capability of the network to map silence zones to the power of breathing sounds (present in clean Mel spectrograms) can also cause this.

The model performance and results are analyzed further in section 5.6. The computational and memory resources required by this model are estimated in segment 5.6.1. With the help of evaluation metrics, a comparison of the results with that of other methods is drawn in segment 5.6.2.

5.5.3. VCAE: Effect of using a Fully convolutional structure

One of the issues with the experiment described in segment 5.5.2 is that while estimating the clean features, the input noisy features have to be divided into blocks of fixed dimension due to the dense bottleneck layer. Moreover, due to a dense fully connected network, the number of parameters increase drastically. According to the observations in segment 5.5.2 and evaluation metrics described in segment 5.6.2, there was no noticeable improvement observed in the performance of the VCAE with a dense bottleneck. Moreover, it came at a great computational cost, as shown in segment 5.6.1, which does not make the dense bottleneck a very practical choice. Hence, in this implementation, instead of the dense bottleneck, the latent space is constructed by using convolutional layers. This is achieved by reducing the number of channels in the latent space and also decreasing the number of bands along the frequency dimension with strided convolutions. Additionally, to verify if the constraint on the overall variance of the latent space enforces network stability, the batch normalization layers in the 'Down Conv Unit' and 'Up Conv Unit' of the encoder and decoder networks are removed. Thus, the aim of this experiment is to study the performance of using a fully convolutional architecture for the Variance constrained Auto-Encoder to denoise noisy features. The configurations for this experiment are mentioned in table C.10 and figure 5.7 depicts the operations in the network.

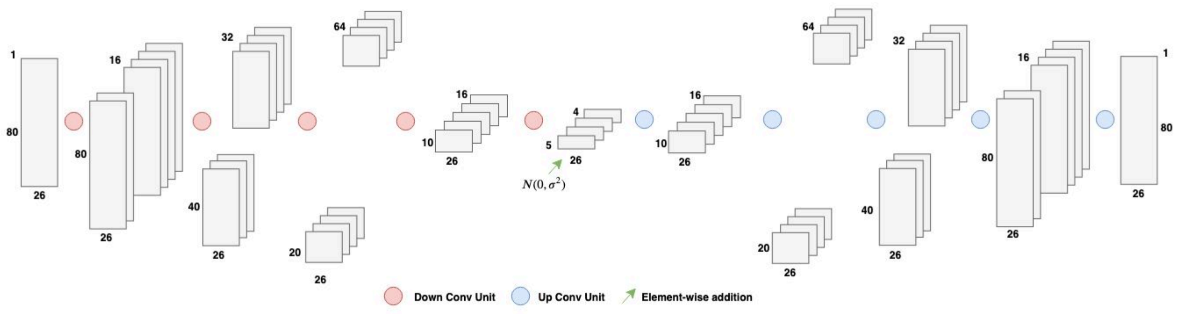


Figure 5.7: Information flow in the fully convolutional VCAE architecture

After training of this model is complete, Mel spectrograms extracted from the noisy speech in the test set are provided to this system. The U-Net model estimates the denoised features as described in section 5.3. The results obtained are illustrated in figure 5.8, where the Mel spectrograms of the clean and noisy speech along with their estimated denoised version are shown. Both male and female speaker cases and contamination with two different types of noises at the same noise power level are analyzed. The following observations were made:

1. The estimated denoised Mel spectrogram appears to be free from noise characteristics even in the case of Cafe noise.

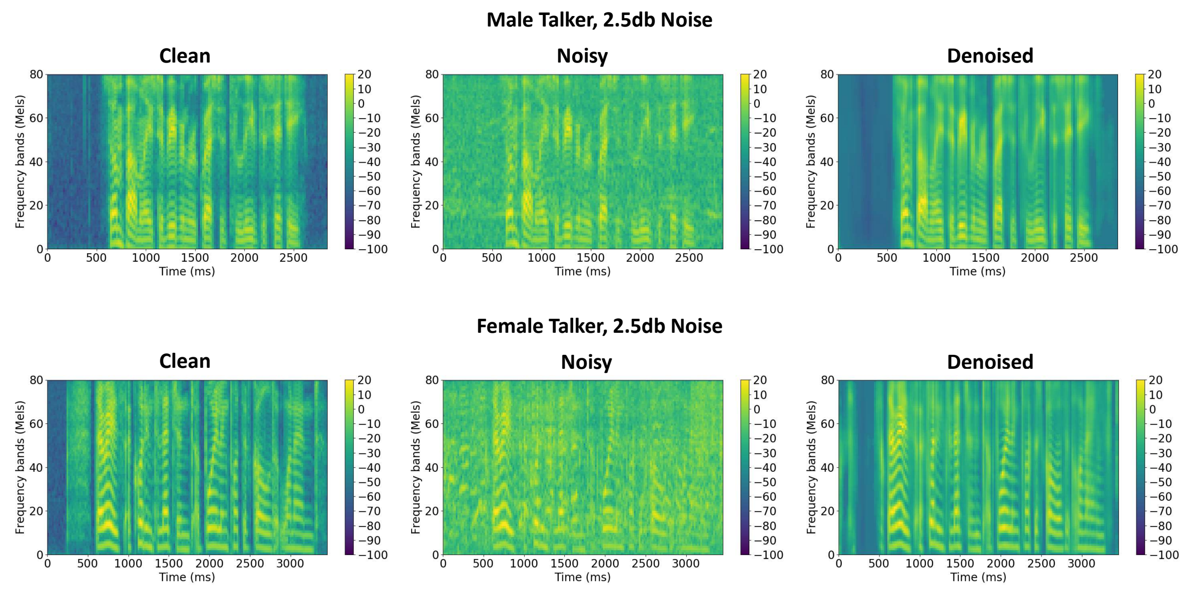


Figure 5.8: **Top Left**: Mel spectrogram of clean speech for a male talker. **Top Middle**: Mel spectrogram for speech corrupted with Living noise at 2.5 dB SNR. **Top right**: Denoised Mel spectrogram of the example (output of the network). **Bottom Left**: Mel spectrogram of clean speech for female talker. **Bottom Middle**: Mel spectrogram for speech corrupted by Cafe noise at 2.5 dB SNR. **Bottom right**: Denoised Mel spectrogram of the example (output of the network).

2. The power during non-speech activity in the denoised Mel spectrograms is mapped to a relatively higher value as compared to that in the clean Mel spectrograms.
3. The harmonic structure in speech in the higher frequency regions are again not reconstructed well.
4. Eliminating the batch normalization layers posed no issue while training as the loss curve converged. However, while testing, the power at some locations exceeded the numerical limit which resulted in NaN values in the decibel scale. Thus, while plotting the results in figure 5.8, the NaN values were clipped to 30 dB.

These observations are also quite similar to those in the previous experiments 5.5.1 and 5.5.2 and so, the reasons behind such behaviour also remain the same. In this case, the VCAE architecture consists of a bottleneck structure that is even smaller than the one made with dense connections. It allows sufficient information to pass through but many details go missing. As a result, the harmonic structure in speech looks blurred especially in the mid and high frequency bands despite of using pre-emphasized features. The reasons for a relatively higher power during non-speech activity in the denoised Mel spectrogram as compared to the reference clean Mel spectrogram remains the same as described in segments 5.5.1 and 5.5.2. The NaN values occur in the decibel scale because of the sigmoidal denormalization function as described in chapter 3 section 3.7. If the extremities of the input range exceed the bounds by a small value, then its mapping to the decibel scale varies exponentially. Hence, the range of the values estimated by the network on the normalized scale must be constrained which is done effectively by the batch normalization layer. Hence, there should be at least one batch normalization layer (preferably at the last layer).

The model performance and results are analyzed further in section 5.6. The computational and memory resources required by this model are estimated in segment 5.6.1. With the help of evaluation metrics, a comparison of the results with that of other methods is drawn in segment 5.6.2.

5.6. Analysis

A network analysis is conducted to get some insights into the functioning and performance of both proposed denoising systems. This section is divided into three segments namely, computational analysis, comparative analysis of network performance and visualizations. In the first segment, the amount of computations and memory resources required for the operations involved in denoising the spectral features are estimated. Secondly, the performance of the different denoising architectures under consideration is evaluated during training and testing with the help of regression based evaluation metrics. Lastly, the layer components and states within the network are visualized to gain an insight into their functionality.

5.6.1. Computational Analysis

In this segment, the calculations involved in obtaining the total number of floating point operations for a convolutional neural network is mentioned. These operations are expressed in terms of the variables used in chapter A segment A.2.5. The number of operations involved in the denoising process are determined by the depth of the network, i.e., the number of layers, and number of convolution operations within a layer, i.e., the number of channels. In a convolutional neural network, the learnable kernel is applied on the input features at every layer with a convolution operation. The convolution operation is also expressed as a flip-multiply-add-shift operations in loop. Out of these, the multiplication and addition operations require higher computations than the rest and hence flops are calculated mainly based on these two operations.

The total number of parameters n in a layer can be calculated as the product of the number of input channels and the kernel size and expressed as $n = C_{in} \times K_1 \times k_2$. Hence, the total number of multiplications and additions for a single convolution instance, i.e., flops per instance, f_i, is given by $f_i = 2 \times n - 1$. As mentioned in chapter A section A.2.5, the size of the features can vary at every layer in a convolutional neural network depending on the padding and stride parameters as expressed by equation A.18. The total number of instances I of applying a convolution is given by the product of the size of the features at the output of that layer and the number of output channels, i.e., $I = C_{out} \times H_{out} \times W_{out}$. Thus, the total number of flops is given by the number of instances I times the flops per instance f_i and expressed as $I \times f_i$. This is repeated for every layer in the network. The procedure remains the same for both the encoder and decoder structures in the networks. In this calculation, operations such as bias parameter addition, batch normalization and activation functions have not been considered as they do not contribute as strongly as the convolution operation.

The procedure for calculating flops for the U-Net and fully convolutional VCAE architectures remains the same as described above. The skip connections in the U-Net architectures have not been considered for the calculation of flops as the number of addition operations is not significantly large. In the VCAE with dense bottleneck architecture, the flops for the dense layers constitute of the matrix-vector dot product between the dense layer weights and the nodes. The flops in a fully connected mapping from I input nodes to J output nodes, is given by $(2I - 1) \times J$. Table 5.3 shows the number of floating point operations for the denoising architectures discussed in this chapter. The U-Net architecture seems to be the approximately 3.5 times more computationally expensive as compared to the VCAE architectures. There was not much difference in terms of computation between the fully convolutional VCAE and the VCAE with dense bottleneck. Using more convolutional layers and large number of channels increase the computational complexity by a sufficiently large amount.

Table 5.3: Computational and memory resource requirements for the denoising architectures

Architecture	Flops	# parameters	Memory
U-Net	1.91 Gflops	0.97M	3.88 MB
VCAE dense bottleneck	0.56 Gflops	32.97M	131.88 MB
VCAE fully convolutional	0.58 Gflops	0.37M	1.48 MB

Along with the number of computations, the memory requirements for the network weights is also an important factor. This depends on the number of learnable parameters n in the network which varies for different architectures and their configurations. Table 5.3 shows the number of parameters (in million) for the U-Net and both VCAE architectures. Since, each element is a $32-\mathrm{bit}$ floating point number, then the size the weight file is $4 \times n$. The VCAE with dense bottleneck has a significantly large number of parameters and hence consumes a lot of memory as compared to the fully convolutional VCAE and U-Net architectures. Thus, due to the sharing of parameters, convolutional neural networks require less memory as compared to dense networks but require more computations. The significance of the numbers in table 5.3 with respect to different hardware considerations is described in chapter 7.

5.6.2. Comparative Analysis of Network Performance

A comparative analysis of the training loss curves and the results obtained after fully training a network are conducted here. Figures 5.9 and 5.10 display the loss curves while training the denoising architectures with Mel spectrograms and Delta features respectively. From the loss curves, it is observed that the VCAE architectures converge at a point that is higher than that of the U-Net architecture. It is suspected that the bottleneck structure in the VCAE model causes such a behaviour since they restrict the flow of information.

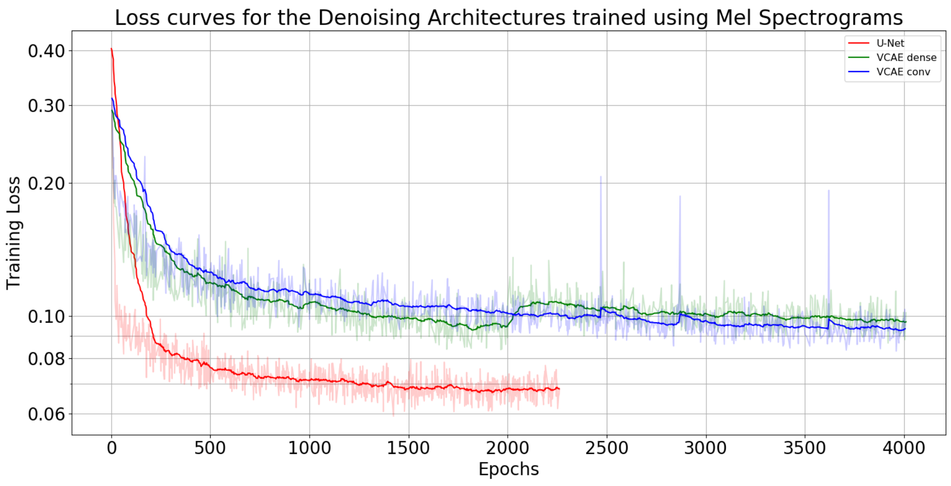


Figure 5.9: Training loss curves for different denoising architectures trained on Mel spectrograms

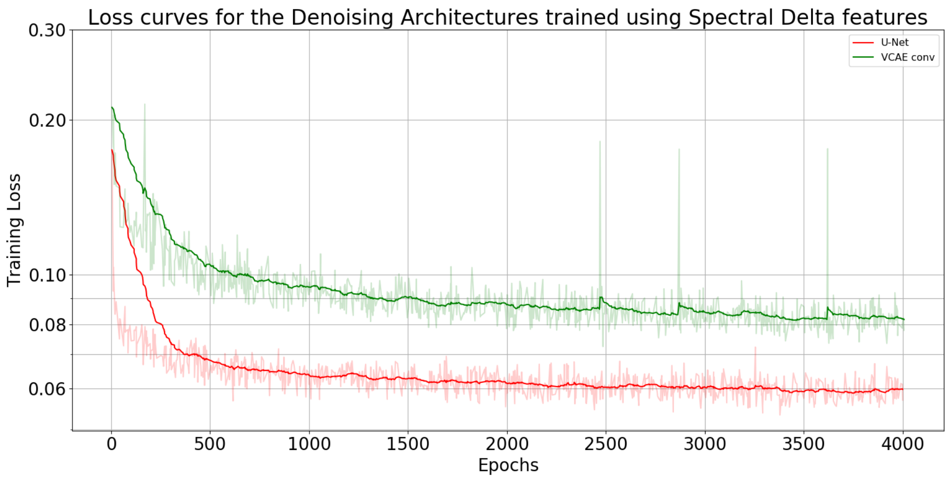


Figure 5.10: Training loss curves for the architectures trained on Spectra Delta features

Two metrics are used to evaluate and compare the results of the different denoising methods. Since, the features are enhanced using regression based methods, the mean square error metric was used to evaluate the performance of this network. The metric is calculated as follows:

$$D = \sqrt{\frac{1}{S} \sum_{i \in M_n, M_c} (L_\sigma(f(N_\sigma(M_n^{(i)}))) - M_c^{(i)})^2} \tag{5.3}$$

where, D is the error metric expressed in decibels as the noisy and clean reference Mel spectrograms, M_n and M_c respectively, are expressed in the same scale. The operations N_σ and L_σ denote the sigmoidal normalization and its inverse transform. f stands for the neural network architecture. S denotes the size of the features, i.e., the product of the number of Mel bands and time segments in the features. Table 5.4 shows the root mean square error (RMSE) of the noisy and denoised Mel spectrograms calculated with respect to the clean reference Mel spectrogram for different noise types and signal-to-noise ratios.

Table 5.4: Root mean square error (RMSE) metric for the Noisy and Enhanced Mel spectrograms with respect to reference Mel spectrograms

Noise Power	Noisy Type	Noisy	U-Net	VCAE dense bottleneck	Fully convolutional VCAE
2.5 dB	Cafe	21.62 dB	**13.58** dB	13.93 dB	14.22 dB
2.5 dB	Bus	8.29 dB	**4.85** dB	5.98 dB	5.53 dB
7.5 dB	Bus	6.38 dB	**4.10** dB	5.26 dB	5.08 dB
7.5 dB	Living	15.59 dB	**7.31** dB	10.99 dB	10.37 dB
12.5 dB	Cafe	13.38 dB	**6.26** dB	7.45 dB	7.04 dB
12.5 dB	Public Square	20.57 dB	10.96 dB	9.52 dB	**9.34** dB
17.5 dB	Public Square	17.05 dB	8.93 dB	9.80 dB	**8.36** dB
17.5 dB	Living	13.18 dB	**5.45** dB	7.00 dB	5.53 dB

Lower the value of the root mean square error means that the average power in the enhanced features is closer to that in the reference features. On comparing the results of the VCAE with a dense bottleneck and the U-Net architecture for their respective configurations, it is found that there is no noticeable improvement in the results despite a dramatic increase in the number of parameters due to the fully connected layers. Thus, using a dense bottleneck consumes more resources than necessary for the task. On the other hand, the fully convolutional VCAE network consumes lesser parameters and delivers a performance very close to or better than the U-Net architecture in case of 'Public Square' noise irrespective of noise power. On an average over different SNRS and noise types, Mel spectrograms enhanced by the U-Net model have the least root mean square error with the reference Mel spectrograms. Note that, this metric does not indicate how precisely are noisy features mapped to clean features but rather provides a rough estimate of the difference in their mean power. For example, the features enhanced by the fully convolutional VCAE do not capture the details of the harmonics in the mid and higher frequency bands very well as the power level across those bands is almost the same. Whereas, the U-Net architecture can produce outputs that can encapsulate the speech harmonic details to a certain extent.

An alternative metric called the Pearson correlation coefficient was used to evaluate the output of the denoising networks. Recall that this metric was also used in chapter 3 section 3.9 to determine the effect of noise in different feature representations. The diagonal elements of the Pearson correlation coefficients between the denoised Mel spectrograms and its corresponding clean Mel spectrogram indicate the strength and the direction of the linear relationship between the bands of the two entities. Since, the features are 2 dimensional, the correlation coefficient is computed by keeping the Mel axis as the variables and the time axis as observations. The correlation coefficient matrix is calculated as follows:

$$\begin{gathered} C(M_T, M_C) = \frac{\mathrm{cov}(\mathrm{M_T}, \mathrm{M_C})}{\mathrm{cov}(\mathrm{M_T}).\mathrm{cov}(\mathrm{M_C})}, \\ \mathrm{cov}(\mathrm{x}, \mathrm{y}) = \mathbb{E}((\mathrm{x} - \bar{\mathrm{x}})(y - \bar{y})), \quad \& \quad \mathrm{cov}(\mathrm{x}) = \mathbb{E}((\mathrm{x} - \bar{\mathrm{x}})^2), \end{gathered} \tag{5.4}$$

where, C stands for the Pearson correlation coefficient between the test Mel spectrogram M_T and clean Mel spectrogram M_C. The test Mel spectrogram can be either the noisy or enhanced by the U-Net or VCAE models. Thus, the diagonal elements of the matrix of correlation coefficients represent the relationship between the power levels at a particular Mel band in the features and the power levels at the same Mel band in the clean features. Figure 5.11 displays the diagonal elements of the Pearson correlation coefficient for the clean and noisy features pair and clean and denoised features pair obtained from all the denoising methods discussed above.

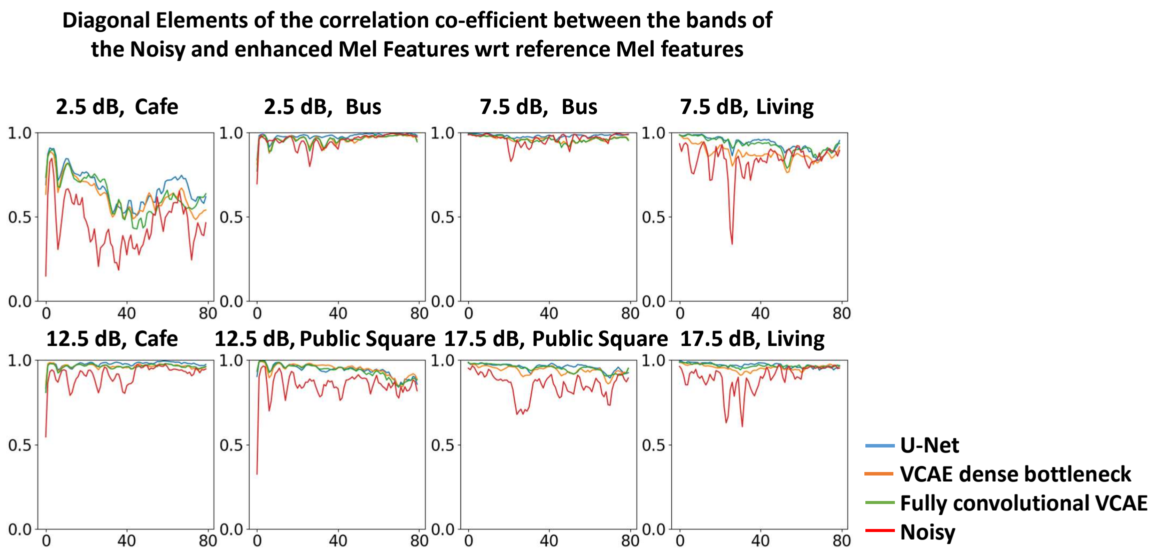


Figure 5.11: Diagonal elements of the Pearson Correlation Coefficient between the Noisy and Denoised Mel Spectrograms obtained from experiments 5.5.1, 5.5.2 and 5.5.3 with respect to the Clean reference Mel spectrograms.

Figure 5.11 shows that the output obtained from the denoising architectures is indeed better than the noisy features as the correlation between bands of the enhanced features with that of the clean features is stronger. For the example with 'Cafe' noise at 2.5 dB SNR, the improvement is not very drastic and is also indicated by the RMSE error. Since, 'Cafe' noise consists of human babble sounds, it is difficult for the network to reduce its effect. The bands in the lower frequencies of the enhanced features are highly correlated with the reference features but the correlation drops for higher bands indicating that the network cannot map the power in higher frequencies precisely despite of using pre-emphasized features. Based on this metric, it is difficult to decide between the performance of the U-Net and the VCAE networks. It is preferable to subjectively evaluate the outputs after fine-tuning the synthesizer with the enhanced features obtained from the denoising architectures.

5.6.3. Visualizations

In order to gain some perspective about how the network functions and what are the purpose of different layers, certain components of the network were visualized.

Latent Space

Both the U-Net and fully convolutional VCAE architectures first transform, or rather, encode the input noisy Mel features to a latent feature space. This section depicts these latent representations visually so as to gain an understanding of the information processed by these networks. The main difference between the two different latent representations are that the space derived from the encoder of a U-Net are only compressed spatially but not in volume, whereas, the VCAE network fully compresses the input feature volume as shown in figures 5.12 and 5.13 respectively. As a result of this, the information encapsulated by these feature spaces are very distinct from each other.

The latent space of the U-Net architecture is shown in figure 5.12 is obtained after the network is fully trained. It consists of 128 channels which equal to the number of square boxes in the figure. Each box has a dimension of 20 bands along the vertical direction and T time segments along the horizontal direction, where T stands for the number of time-segments in the Mel features. As mentioned in experiment 5.5.1, the convolution units in the encoder preserve the dimension along the time axis and compresses the number of bands along the frequency axis. However, the volume of the latent space here is larger than the volume of the input due to the large number of channels in the space (as per the current setup, the latent space volume is 32 times the input volume).

The latent space encapsulates large amounts of details as seen in multiple channels in figure 5.12. To illustrate, channel 17 activates during speech activity in the input features and captures that infor-

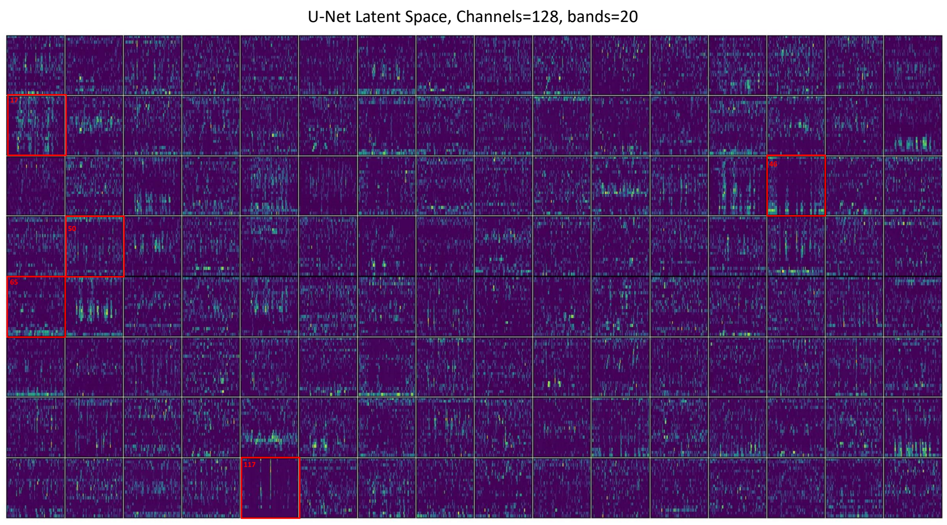


Figure 5.12: Output of the encoder, i.e., the latent feature representation for the U-Net architecture.

mation. On the other hand channel 46 captures only the silence parts in the input features. Channel 117 activates for speech on-sets. This activation also depends on the amount of noise present in the input. For a high SNR text example, this channel activates on every on-set where-as for a low SNR example, it may miss some since this information is difficult to capture under influence of high noise. Channel 50 captures the high and mid frequency information whereas channel 65 contains low frequency information. The latent space visualization shown here is for one test example from the test dataset, but the characteristics captured by the channels remain similar for all examples. This depicts the generalization of neural networks for different types of input features, the channel interpretation remains the same.

The latent space of the fully convolutional VCAE architecture is shown in figure 5.13 is obtained after the network is fully trained. It consists of 4 channels which equal to the number of rectangular boxes in the figure. Each box has a dimension of 5 bands along the vertical direction and T time segments along the horizontal direction, where T stands for the number of time-segments in the Mel features. As mentioned in experiment 5.5.3, the convolution units in the encoder preserve the dimension along the time axis and compresses the number of bands along the frequency axis. Additionally, the volume of the latent space for a VCAE network is lesser than the volume of the input features which constitutes a bottleneck (as per the current setup, the latent space is compressed to 4 fold of the input volume).

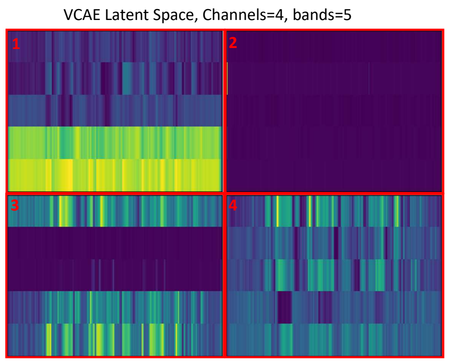


Figure 5.13: Output of the encoder, i.e., the latent feature representation for the fully convolutional VCAE architecture.

Due to a compression in the latent space of the VCAE network, it is difficult to interpret the the latent feature representation as compared to the U-Net latent space. The VCAE encapsulates most of the information that a U-Net does in fewer dimensions, however, it is believed that the amount of details captured here are lesser than what a U-Net latent space contains. This is reflected in the output of the VCAE where mid to high frequency information if smooth and the speech harmonic structure is not captured. Channels 1, 3 and 4 capture the onset of speech and speech activity in different bands. Channel 2 only activates at one band and the rest of space contains values very close to 0. Not much can be inferred from this latent space visualization due to the heavy compression. This visualization shown is for the same test example as the , but the characteristics captured by the channels remain similar for all examples. This depicts the generalization of neural networks for different types of input features, the channel interpretation remains the same.

Concluding Remarks

This chapter introduced two different architecture designs namely the U-Net and the Variance Constrained Auto-Encoder. Their setup, training and testing procedures were discussed along with experimentation and analysis of their components. This section summarizes the key findings mentioned in the various topics covered in the chapter. These conclusions then form the baseline for the upcoming chapter that involve combining the architectures discussed in this chapter with the WaveRNN synthesizer.

The following conclusions are drawn from this chapter:

- Both the proposed network designs, i.e., the U-Net and VCAE architectures perform significantly well at removing the noise from Mel features.
- Observations indicate that features denoised by the U-Net architecture tend to capture the harmonic structure in speech very well and even at the higher frequency bands. This is attributed to the large amounts of details captured by the latent space of the U-Net.
- On the other hand, both VCAE architectures tend to level the speech power in the higher frequencies and are unable to effectively encapsulate the harmonic structure in speech. Visualizing the compressed latent space in the VCAE architecture is difficult to interpret.
- From the two different structures proposed for the VCAE network, the fully convolutional structure seems to be a more practical choice to continue with. Not only does it perform better than the the structure with a dense bottleneck but also is more efficient in terms of memory and computational resources. Moreover, the size of input features can also be chosen arbitrarily and does not depend on the architecture.
- It is difficult to make a choice between the U-Net and the fully convolutional VCAE architectures based on the evaluation metrics since their performances are quite competitive.
- Computational analysis for the U-Net and fully convolutional VCAE networks indicate that the latter requires lesser memory resources. As far as these differences are not orders of magnitude apart this is not a strong distinguishing factor between the two architectures.

6
Fine-tuning the Synthesizer

From chapter 4, it was observed that when conditioning Mel features provided to the WaveRNN synthesizer are clean, then the synthesized speech sounds very similar to the reference. Whereas, when the context is noisy, it cannot produce understandable speech at all. Thus, it is crucial to remove noise from these features before they are provided to the WaveRNN for synthesis as described in chapter 5. The idea behind synthesizing clean sounding speech from noisy features is inspired by a phenomenon known as Transfer Learning [71, 72]. Thus, instead of training the synthesizer with a random initialization on the context provided by enhanced features, the synthesizer can first be trained on clean features, and then this pre-trained model can be fine-tuned by training with enhanced features. The enhanced features are obtained from the denoising architectures described in chapter 5. This allows the synthesizer to learn the transformation from clean features to speech in advance, which can be beneficial in terms of faster convergence.

This chapter is organized as follows; first, an overview of literature on transfer learning is provided in section 6.1. This is followed by section 6.2 which contains a description on the complete system formed by integrating the denoising architectures with the WaveRNN synthesizer. Later, the training process for the combination of the denoising and synthesizer networks is highlighted in section 6.3. Section 6.6 showcases the experiments performed by combining the pre-trained U-Net and VCAE networks with the synthesizer and results obtained from both these configurations are discussed. Results from both these experiments reveal that the for the selected model configurations, the U-Net outperformed the VCAE in denoising features. Hence, the final system was constructed with a U-Net and described in experiment 6.6.3. Thereafter, an analysis is reported in section 6.7 on the MUSHRA subjective listening test [73] conducted and also on an objective intelligibility measure known as the Word Error Rate (WER) [74]. In the end, concluding remarks from this are summarized to elucidate the findings in this chapter.

6.1. Background on Transfer Learning

Transfer learning is a phenomenon used in machine learning that is said to be inspired by human behaviour of applying knowledge gained from previously learnt domains to newer tasks. GoodFellow *et al.* refer to transfer learning in the context of generalization where some aspects learnt in one particular setting can be exploited to improve generalization in another setting [75]. This means that instead of learning a task ground up, it can be better to re-use the learnt knowledge for solving problems in other domains. For instance, the initial layers of convolutional networks trained on images capture simple features such as edges or blobs and the last layers are responsible for the task specific features. Thus, one could re-use features from the initial layers of the pre-trained network off-the-shelf to train it for a different task.

A mathematical definition for the transfer learning problem can be utilized to understand different strategies for applying them [71]. Consider a domain, $\mathcal{D} = (\chi, P(X))$, that consists of a feature space χ and a marginal probability distribution of the features $P(X)$, where $X = [x_1, ..., x_n] \forall x_i \in \chi$. For the given domain $\mathcal{D}$, a task $T = (\gamma, \eta)$ is defined by two components, namely, the label space, γ, and the objective function, η. The objective function can also be expressed as a conditional probability distribution denoted as $P(Y|X)$, where $Y = [y_1, ..., y_n] \forall y_i \in \gamma$. With these representations, transfer learning is defined as, given a source domain, $\mathcal{D}_S$, a corresponding source task, $\mathcal{T}_S$, as well as a target domain, $\mathcal{D}_T$ and a target task, $\mathcal{T}_T$, the objective of transfer learning is to enable the target conditional probability distribution $P(Y_T|X_T)$ in $\mathcal{D}_T$ be learned with the information gained from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and/or $\mathcal{T}_S \neq \mathcal{T}_T$. Based on these definitions, transfer learning can be applied for the following scenarios:

1. $\chi_S \neq \chi_T$, the feature space of the source and target domains are different.
2. $P(X_S) \neq P(X_T)$, the marginal probability distributions of source and target domains are different.
3. $\gamma_S \neq \gamma_T$, the label spaces between two tasks are different.
4. $P(Y_S|X_S) \neq P(Y_T|X_T)$, the conditional probability distributions of the source and target tasks are different.

It is crucial to categorize the problem at hand and then apply appropriate strategies for transfer learning. To realize this, the following classes help to decide so:

- **Homogeneous Transfer** - If the feature space and label space between the source and target domains for the task are similar, i.e., $\chi_S = \chi_T$ and $\gamma_S = \gamma_T$, but the data distributions between them are different, $P(X_S) \neq P(X_T)$ and $P(Y_S|X_S) \neq P(Y_T|X_T)$, then the problem is classified as a Homogeneous transfer learning problem.
- **Heterogeneous Transfer** - If the feature space and/or label space between the source and target domains for the task are not the same, i.e., $\chi_S \neq \chi_T$ and/or $\gamma_S \neq \gamma_T$, then the problem is classified as a Heterogeneous transfer learning problem. Solutions in this case attempt at bridging the gap between feature spaces to reduce the problem to a Homogeneous transfer learning problem.

For the speech synthesis problem, clean feature space, denoted by χ_S, and the noisy feature space, denoted by χ_T, are not the same at all. This is a case of Heterogeneous transfer learning problem. Hence, before synthesizing clean sounding speech the gap between the clean and noisy features must be reduced. This feature transformation is known as domain adaptation [76]. Hence, the denoising architectures introduced in chapter 5 are used for transforming or rather enhancing the noisy feature space to the clean feature space. To be more specific to this application, the transform is termed as an asymmetric feature transform since they have the same target label space, i.e., the reference clean speech that can be denoted by γ_S or γT [76].

In addition to domain adaptation, the WaveRNN synthesizer should also be well trained on the clean feature space or the source domain, χ_S so as to learn well-defined s structure. Since, the task of synthesizing speech from features is related for both source and target domains, the structure of the transformation can also be transferred to the target model. This category of transfer learning is called Parameter-based transfer learning since the parameters of the networks are shared for both source and target domain tasks. The initialization of weights for the target task are the same as the previously trained weights for the source task and is known as hard weight sharing [77]. This reduces convergence time for fine-tuning the network for the target task. Moreover, fine-tuning the network parameters along with a transfer of features can improve generalization of networks [78].

6.2. System Workflow

Combining the WaveRNN synthesizer with the denoising network architecture can allow generation of clean sounding speech given noisy features. One way to achieve this is to directly train the WaveRNN synthesizer with the denoised features obtained from the denoising architecture. Instead of randomly initializing the WaveRNN model weights, it would be beneficial to train the network from a point where it already possesses some knowledge about the transformation from the feature space to the waveform as described in section 6.1. Therefore, the WaveRNN synthesizer is pre-trained on clean features and later fine-tuned by training with noisy features that are pre-processed by a denoising architecture. Thus, the transfer learning technique is a combination of domain adaptation and parameter sharing. Figure 6.1 shows the entire workflow of the system where it is divided into three phases, namely Training and Fine-tuning described in section 6.3 and Generation described in section 6.4.

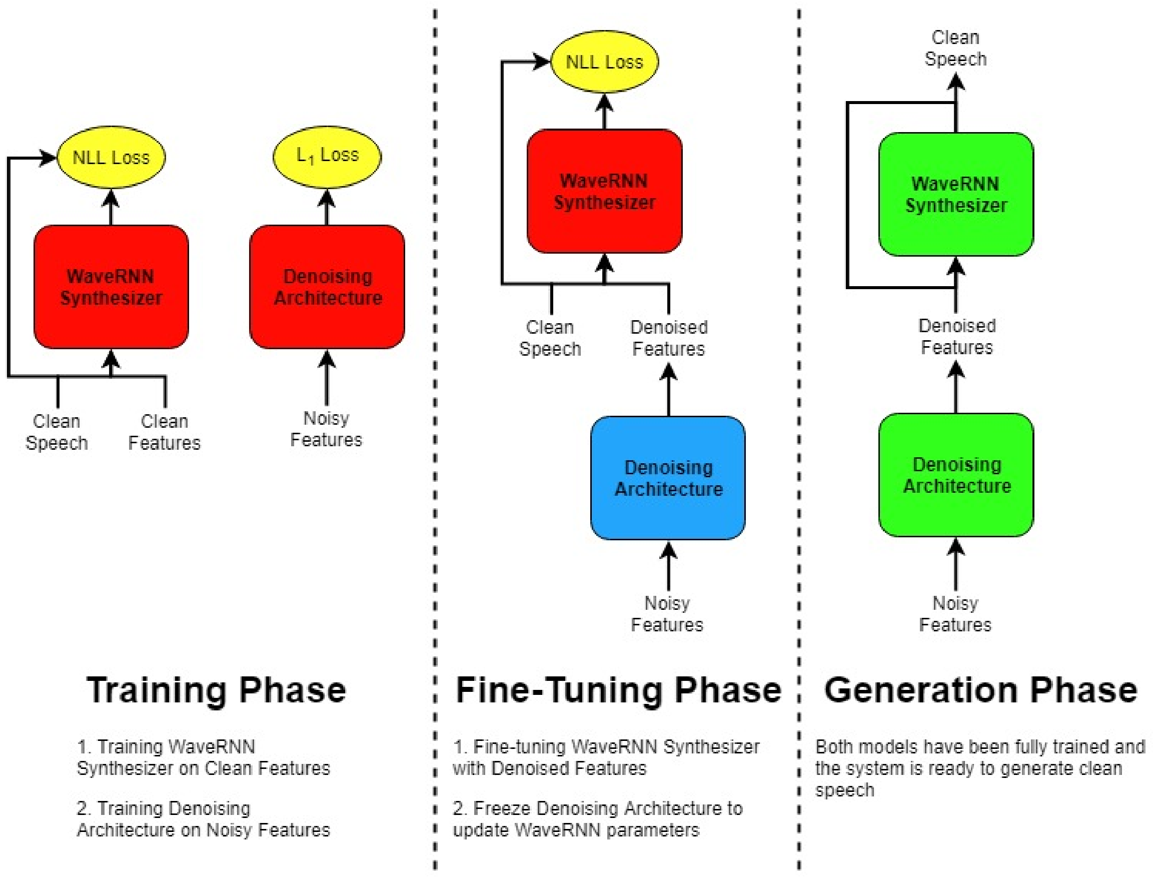


Figure 6.1: Using Transfer Learning to improve the synthesis process

6.3. Initial Training & Fine-tuning

In order to share parameters of the source task with the target task, the pre-trained weights of the model for the source task must be available. In this case, a pre-trained model was not readily available to be used since the authors of the WaveRNN system [21] have not made it public. Hence, the training phase shown in the leftmost part of figure 6.1 illustrates the training of a WaveRNN synthesizer on clean speech and clean Mel features. This training procedure was already mentioned in chapter 4 section 4.3 with the configurations from experiment 4.5.2. Alongside, the denoising network is also trained to map the noisy Mel features to clean Mel features. The training process for domain adaptation was mentioned in chapter 5 section 5.2 for all denoising architectures with their respective configurations. Once these models are trained, the initial training phase is completed, i.e., the pre-trained models are ready and the fine-tuning phase begins.

During the fine-tuning phase, only the parameters of the WaveRNN model are updated and the weights of the denoising model are frozen as shown in the middle part of figure 6.1. The pre-trained WaveRNN network weights trained for the source task, i.e., synthesizing clean speech from clean features, are loaded and act as an initial point for the fine-tuning to begin. The batch of noisy Mel features are first provided to the pre-trained denoising model in order to obtain the enhanced features. Thereafter, a batch of clean speech segments of sequence length along with the denoised features (context) of the corresponding example are presented to the WaveRNN synthesizer. The remainder of the training process for the WaveRNN remains the same as described in chapter 4 section 4.3. The weights of the WaveRNN model are updated for each batch on every iteration in every training cycle. The convergence of the loss function is an indication of correct training and once this model is fine-tuned, it can be used to generate clean sounding speech when noisy features are provided as context to the system.

6.4. Generating Speech

The generation process for synthesis using the WaveRNN model remains the same as described in chapter 4 section 4.2. The only difference is that instead of providing noisy context directly to the synthesizer, they are processed by the denoising architecture before they are supplied to the WaveRNN model as shown in the rightmost part of figure 6.1. Both synthesis and denoising models are in the evaluation mode where no gradients are backpropagated and thus no parameter update takes place. By fine-tuning the WaveRNN synthesizer, it has learned the transformation from denoised Mel features to clean speech.

Even though the synthesizer generates speech samples in an auto-regressive fashion as described in chapter 4 section 4.2, the provision of context to the synthesizer is not auto-regressive in nature. This means that the context has to be provided to the synthesizer as whole and not one time segment at a time. Because, the denoising architecture is a convolutional neuranl network, the minimum number of time segments that can be provided to the synthesizer is limited by the size of the kernel along the time axis (which is greater than 1.[1]) Hence, if the complete system were to be implemented in an auto-regressive fashion, it would lead to large tracking delays. With the current setup, the delays can be greater than twice the window length of the frame under analysis if implemented in an auto-regressive fashion.

6.5. Setup

The network setup for all networks remains the same as described in tables 4.1, 5.1 and 5.2 during the initial training phase. While fine-tuning the synthesizer, the number of time-segments for the noisy Mel features are selected according to the sequence length of the clean speech provided to the WaveRNN model.

6.6. Experiments & Results

The experiments performed in this chapter are a continuation of the one described in chapter 4, i.e., training WaveRNN synthesizer on clean features 4.5.2. As concluded in chapter 5, denoised features obtained with the U-Net, 5.5.1, and fully convolutional VCAE, 5.5.3 (from now on will be referred to as VCAE), architectures are used here. All these models act as pre-trained models and constitute the initial training part in figure 6.1. Here, they were combined with the synthesizer to generate speech by fine-tuning the WaveRNN synthesizer on denoised features. The aim for each experiment is stated in their respective segments. Results obtained from the respective experiments are also shown. There were no changes made in the network architecture and neither in network hyper-parameters nor feature configurations. The generation of speech and training procedure for the combined network is described in sections 6.3 and 6.4 respectively. Also note that the experiments in segments 6.6.1 and 6.6.2 were initially conducted on features extracted from speech sampled at 8kHz for time and complexity reasons.[2] Once the results with the best combination were identified, the final system was trained and tested again for 16kHz speech as shown in segment 6.6.3.

6.6.1. Combining U-Net with WaveRNN

The aim of this experiment was to evaluate the performance of the system when the WaveRNN is fine-tuned with Mel features denoised by the U-Net architecture. In this configuration, the U-Net was set to evaluation mode and its weights were frozen, whereas the weights of the WaveRNN synthesizer were updated on every training cycle. Freezing the weights mean that the gradients are not backpropagated to update the parameters of the U-Net. The configuration for the synthesizer remains the same as described in table C.4. Experiments were conducted with both Mel spectrograms and Delta features. First, the results obtained from Mel spectrograms are described after which the results for Delta features are reported.

After training was complete, Mel spectrograms extracted from the noisy speech present in the test set were first provided to the U-net architecture of the fine-tuned system for denoising. These enhanced features were provided to the WaveRNN synthesizer where they act as context for generation of clean

[1]For more details, check the network configurations for the experiments in Appendix C
[2]Refer to Appendix B.1.3 for more details

speech as described in section 6.4. The results obtained are illustrated in figures 6.2 and 6.3 where the waveform of the clean, noisy and generated speech along with their respective spectrograms are shown for the same examples as in the experiments of chapter 4. On listening to the audio files, the following observations were made:

1. No distortions or artifacts during speech activity. However, there are some instances during generation when the network becomes unstable and some segments of speech become corrupted as discussed in chapter 4 experiment 4.5.2.
2. The voice of the speaker is correctly reproduced with some minor quavering. The intonation of the speaker is also fairly captured but it varies in case of extreme noise scenarios.
3. Some information is missing or mumbled especially for low SNR samples. There is a drastic improvement in intelligibility as compared to the case when noisy context was directly provided to the synthesizer. And there is scope for further enhancement.
4. Consonant sounds and some phonemes such as 'f', 'p', 'b', 'd', etc. are not produced correctly in case of extreme noise and this leads to misidentifying the correct word.

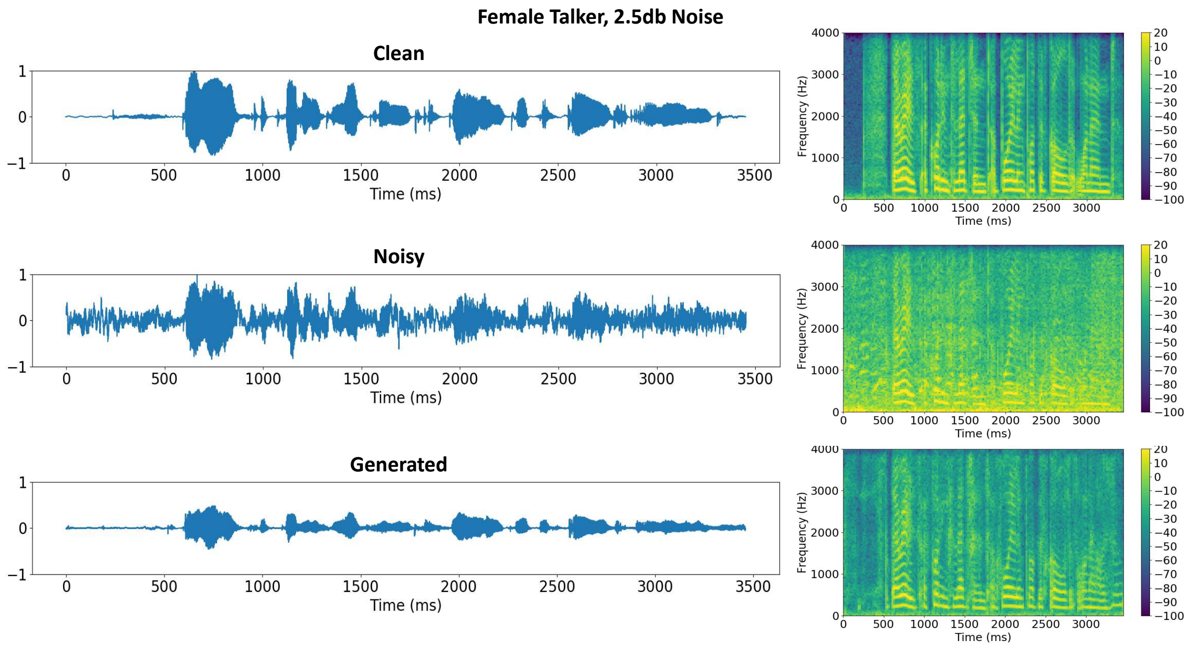


Figure 6.2: **Top**: Clean speech waveform and its spectrogram for a female talker. **Middle**: Cafe noise added at 2.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from denoised Mel spectrogram as context, and its spectrogram.

On comparing the spectrogram obtained from the generated speech with that of the clean speech, it is observed that in most cases there are no abrupt discontinuities that represent distortions. However, sometimes there are a few examples that exhibit an unusual behaviour similar to the description in chapter 4 experiment 4.5.2. Consider, for instance, in figure 6.3, where in the beginning of the waveform, there is an artifact. Such occurrences have been attributed to instability in the GRU layer as discussed in chapter 4 section 4.6.3.1. This instability in the GRU layer can be either due to insufficient training or over-training with a large learning rate [60]. Also note that, the existence of such artifacts is very random and does not occur always. If the same example is generated once again with the same conditions, the waveform may or may not contain such artifacts.

The pitch of the speaker is effectively reproduced as shown in both examples. The difference in the speech harmonics indicates the pitch frequency and this difference remains approximately the same as seen in the spectrograms of clean and generated speech. However, the intonation is slightly different from the clean reference which can be understood by observing the gradients in the harmonic peaks or the formants along the time axis. The quavering in voice can be spotted in the spectrogram

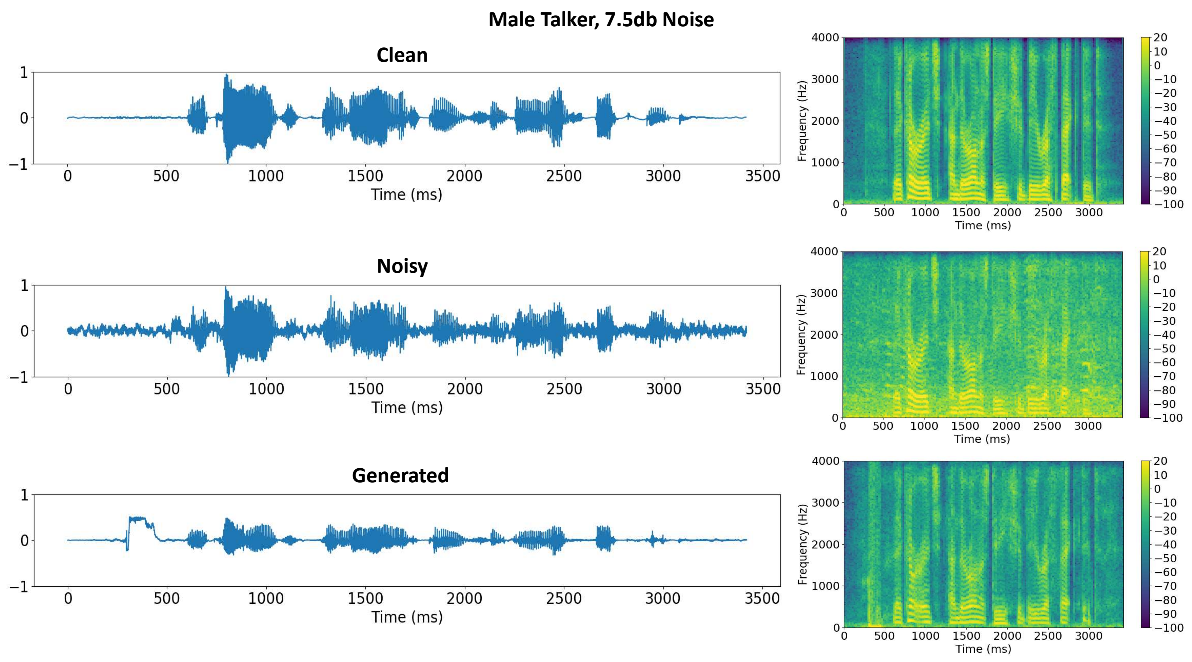


Figure 6.3: **Top**: Clean speech waveform and its spectrogram for a male talker. **Middle**: Public square noise added at 7.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from denoised Mel spectrogram as context, and its spectrogram.

of generated speech in figure 6.3 in between time segments $2900 - 3100$ ms. This occurs due to extra peaks occurring in between the speech harmonic peaks. Mumbled speech may occur due to the existence of harmonics where they are not supposed to be present. For instance, on comparing the last few time segments of the spectrogram for generated speech with that of the clean speech in figure 6.2 shows the presence of speech harmonics in the former whereas they are not present in the latter.

Figure 6.4 displays the reference (clean) Mel spectrograms, denoised Mel spectrograms (output of the pre-trained U-Net model and input to the WaveRNN) and the Mel spectrograms obtained from generated speech for the same examples as shown in figures 6.2 and 6.3. The denoised Mel spectrograms for the less noisy case, i.e., the top part of figure 6.4, is able to fairly capture the onsets of speech at the appropriate bands. However, when the input features are noisier, for instance, the lower part of figure 6.4, the denoised Mel spectrogram can rarely capture these onsets correctly at the higher frequency bands. As a result, the consonants and unvoiced speech segments do not sound very clear for generated speech in most examples. If the input features cannot provide appropriate context for generation, the synthesizer network cannot produce speech with correct sounding words.

To solve the problem of capturing consonant sounds, the use of Delta features described in chapter 3 section 3.6 was considered. The aim of this experiment is to examine the effect of fine-tuning the synthesizer with Delta features that are denoised by the U-Net. The proposed delta features not only include information about the harmonics in speech but also relatively amplify the power on speech onsets. After training the synthesizer with clean Delta features as described in chapter 4 experiment 4.5.2 and enhancing the noisy ones with the U-Net architecture, the WaveRNN is fine-tuned with these denoised Delta features. On testing this system, there was a considerable improvement in producing consonant sounds for most examples which in turn increased the intelligibility. There also exist cases with extreme noise conditions where the use of Delta features did not seem to alleviate the problem as expected, however, there was noticeable enhancement as compared to the results with Mel spectrograms. Figures 6.5 and 6.6 show the clean, noisy and generated speech waveforms along with their respective spectrograms.

Figures 6.7 and 6.8 displays the reference (clean) Delta features, noisy Delta features (input to the U-Net model) and denoised Delta features (output of the U-Net model and input to the WaveRNN synthesizer) and the Delta features obtained from generated speech for the same examples. The

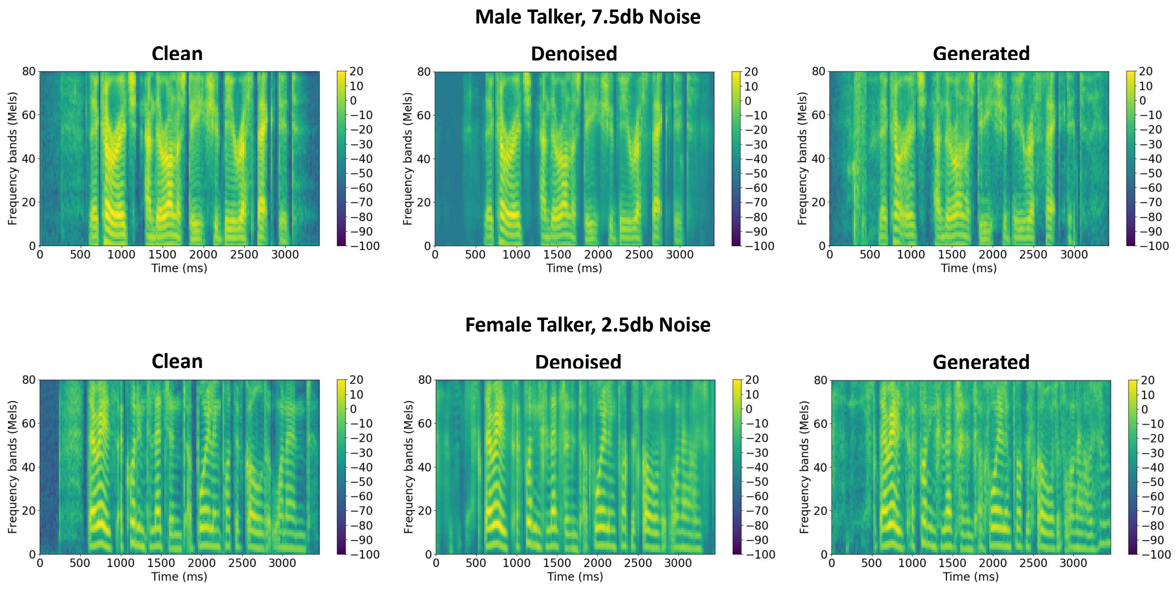


Figure 6.4: **Top Left**: Mel spectrogram of clean speech for a male talker. **Top Middle**: Denoised Mel spectrogram obtained from the U-Net model when fed with noisy features from speech corrupted with public square noise at 7.5 dB SNR. **Top right**: Mel spectrogram of the generated speech for the example.**Bottom Left**: Mel spectrogram of clean speech for female talker. **Bottom Middle**: Denoised Mel spectrogram obtained from the U-Net model when fed with noisy features from speech corrupted with cafe noise at 2.5 dB SNR. **Bottom right**: Mel spectrogram of the generated speech for the example

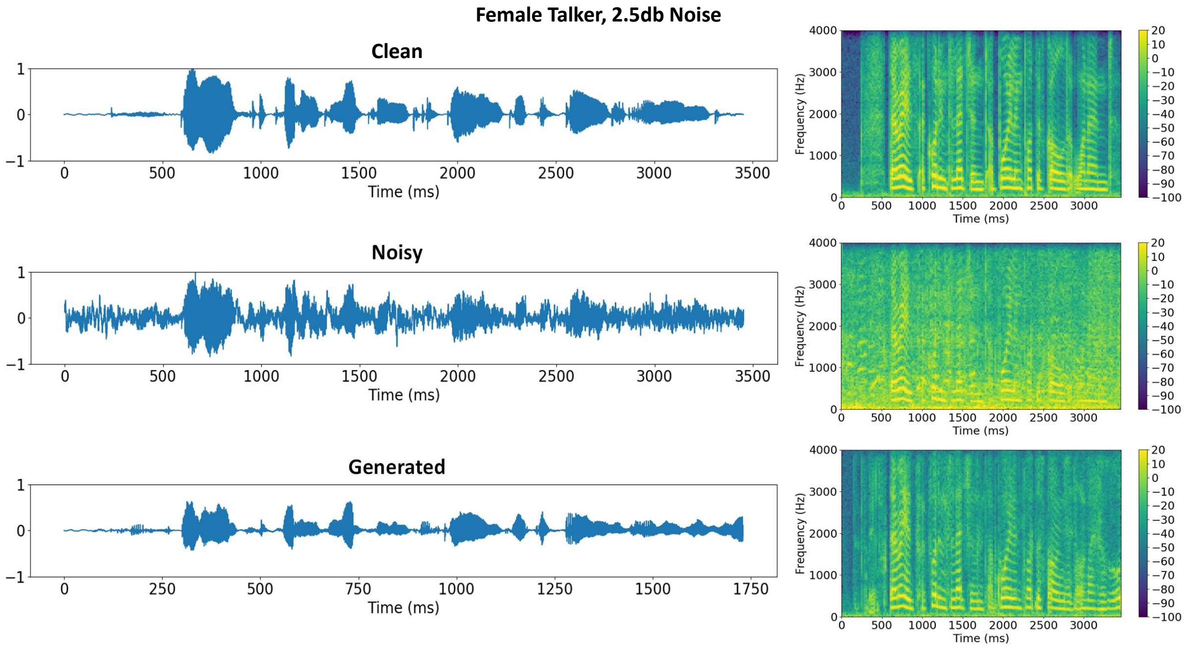


Figure 6.5: **Top**: Clean speech waveform and its spectrogram for a female talker. **Middle**: Cafe noise added at 2.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from denoised Delta features as context, and its spectrogram.

denoised Delta features can encapsulate the speech onsets correctly irrespective of the noise power level. However, in case of extreme noise as shown in figure 6.7, the power is not correctly distributed across bands. For instance, observe the speech onset at the 2000 ms time segments and compare the denoised and reference Delta features. Power in the higher bands has not been captured correctly and this causes the consonant sound to be different from the reference. In the Delta features obtained from

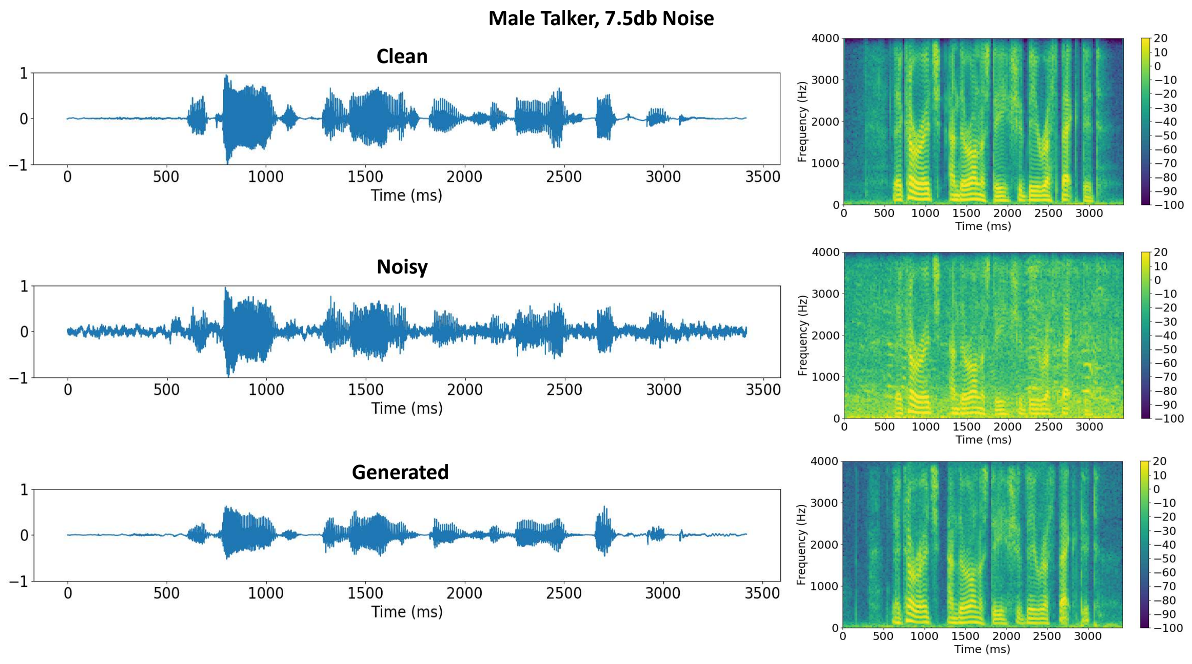


Figure 6.6: **Top**: Clean speech waveform and its spectrogram for a male talker. **Middle**: Public square noise added at 7.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from denoised Delta features as context, and its spectrogram.

generated speech, it is seen that the synthesizer does not compensate for such missing characteristics. This indicates that the WaveRNN simply produces speech based on what input it receives and does not try to rectify for such loss in information. On the other hand, there are no noticeable missing traits in the Denoised delta features and delta features from the generated speech for the example with less noise power as shown in figure 6.8.

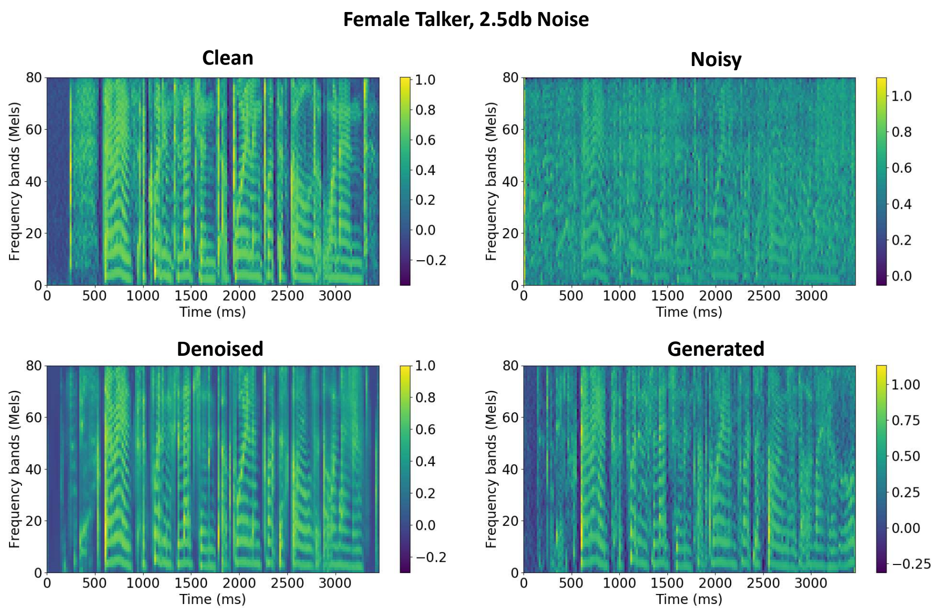


Figure 6.7: **Top Left**: Delta Features of clean speech for a female talker. **Top Right**: Delta features from speech corrupted with cafe noise at 2.5 dB SNR. **Bottom Left**: Denoised Delta features obtained from the U-Net model. **Bottom right**: Delta features of the generated speech for the example.

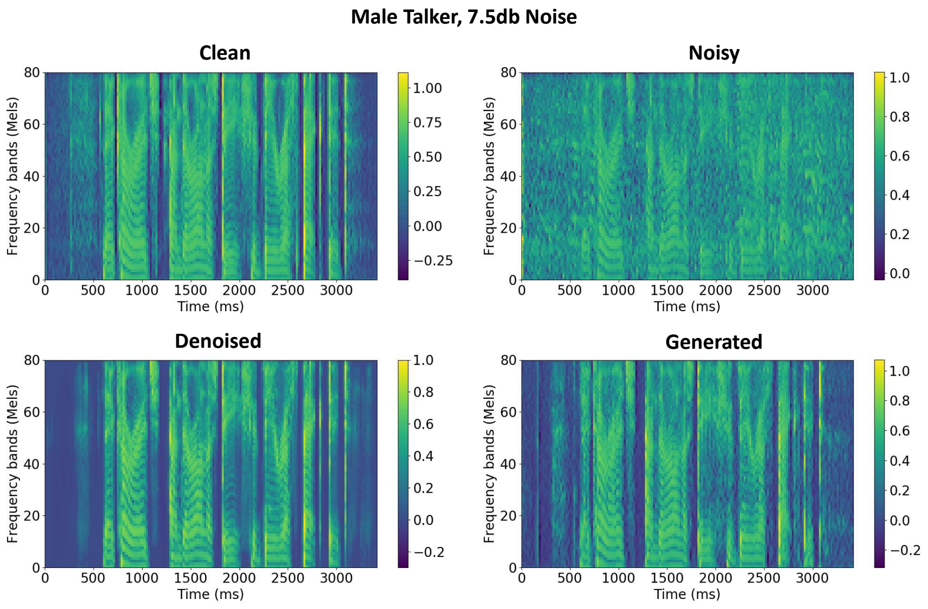


Figure 6.8: **Top Left**: Delta Features from clean speech for a male talker. **Top Right**: Delta features from speech corrupted with public square noise at 7.5 dB SNR. **Bottom Left**: Denoised Delta features obtained from the U-Net model. **Bottom right**: Delta features of the generated speech for the example.

6.6.2. Combining VCAE with WaveRNN

This experiment was performed to evaluate the performance of the system when the WaveRNN is fine-tuned with features denoised by the VCAE network. In this configuration, the VCAE network is set to evaluation mode and its weights are frozen, whereas the weights of the WaveRNN synthesizer are updated on every training cycle. Freezing the weights mean that the gradients are not backpropagated to update the VCAE network parameters. The configuration for the synthesizer remains the same as described in table C.4. While implementing, experiments were conducted for both Mel spectrograms and Delta features but speech synthesized from Delta features sound clear and better in terms of quality. Hence, this experiment directly showcases results obtained from training and fine-tuning the system with Delta features.

After training was complete, Delta features extracted from the noisy speech present in the test set are provided to the VCAE network of the fine-tuned system for denoising. These denoised Delta features are then presented to the WaveRNN synthesizer where they act as context for generation of clean speech as described in section 6.4. The results obtained are illustrated in figures 6.9 and 6.10 where, the waveform of the clean, noisy and generated speech along with their respective spectrograms for the same examples are shown. On listening to the audio files, the following observations were made:

1. No distortions or artifacts during speech activity. However, there are some instances during generation when the network becomes unstable and some segments of speech become corrupted.
2. The pitch seems to be reproduced well enough but quavering in the speakers voice is more prominent than the U-Net trained with Delta features in experiment 6.6.1. The intonation also varies in some cases. For extreme noisy scenarios, the speaker voice seems to differ a bit from the original reference.
3. Some information is missing or mumbled up especially for low SNR samples. There is a noticeable improvement in intelligibility as compared to the case when noisy context was directly provided to the synthesizer. And there is scope for further enhancement.
4. Consonant sounds are produced correctly even for moderately high noise power cases but there is some missing clarity as compared to experiment 6.6.1 for using Delta features. There exist extreme noise scenarios where some phonemes are not produced correctly which causes misidentifying some words.

On comparing the spectrogram obtained from the generated speech with that of the clean speech, it is observed that in most cases there are no abrupt discontinuities that lead to distortions. Though, not visible in the figures present here, there exist other examples that exhibit artifacts in generated speech similar to the ones described in experiment 6.6.1. The reason for such unusual behaviour seems to be due to instability in the GRU layer either due to insufficient training or over-training with a large learning rate. The quavering voice of the speaker can be observed in the spectrogram of the generated speech in figure 6.10 between time segments $1800 - 2000$ ms and $2600 - 3000$ ms. The region in between peaks of the speech harmonics have relatively similar power and existence of false peaks. The presence of harmonics where they are not supposed to be present leads to mumbled speech which makes it difficult to understand. For instance, on comparing the the last few time segments of the spectrogram for generated speech with that of clean speech in figure 6.9 shows the presence of extra harmonics in the former whereas they are not present in the latter.

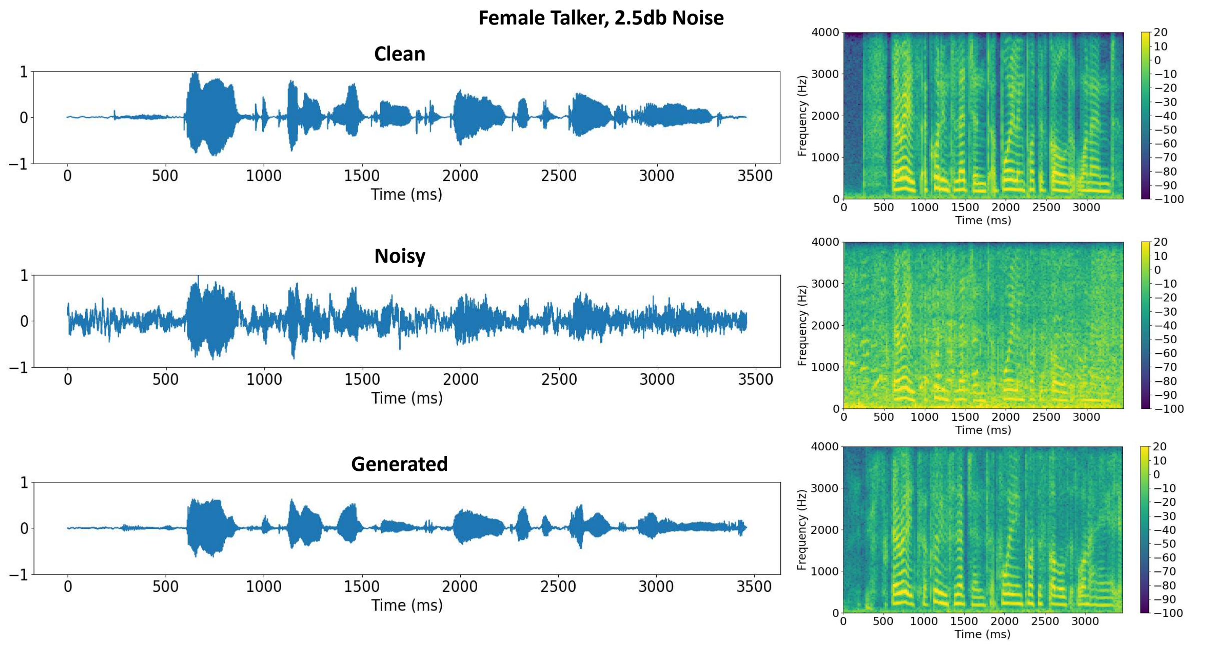


Figure 6.9: **Top**: Clean speech waveform and its spectrogram for a female talker. **Middle**: Cafe noise added at 2.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from denoised Delta features as context, and its spectrogram.

Figures 6.11 and 6.12 displays the reference (clean) Delta features, noisy Delta features (input to the U-Net model) and denoised Delta features (output of the U-Net model and input to the WaveRNN synthesizer) and the Delta features obtained from generated speech for the same examples. The denoised Delta features can encapsulate the speech onsets correctly irrespective of the noise power level. However, in case of extreme noise as shown in figure 6.11, the power is not correctly distributed across bands. For instance, observe the speech onset at the 2000 ms time segments and compare the denoised and reference Delta features. Power in the higher bands has not been captured correctly and this causes the consonant sound to be different from the reference. In the Delta features obtained from generated speech, it is seen that the synthesizer does not compensate for such missing characteristics. This indicates that the WaveRNN simply produces speech based on what input it receives and does not try to rectify for such loss in information. On the other hand, there are no noticeable missing traits in the Denoised delta features and delta features from the generated speech for the example with less noise power as shown in figure 6.12. On top of this, due to the compression of the latent space in the VCAE, the power in the higher frequency bands of voiced speech segments appears to be at the same level. The harmonics structure in speech is not effectively captured.

Based on intuition gathered from listening to multiple generated speech files obtained from both experiments 6.6.1 and 6.6.2, it was concluded that generated audio from the former experiment sounded

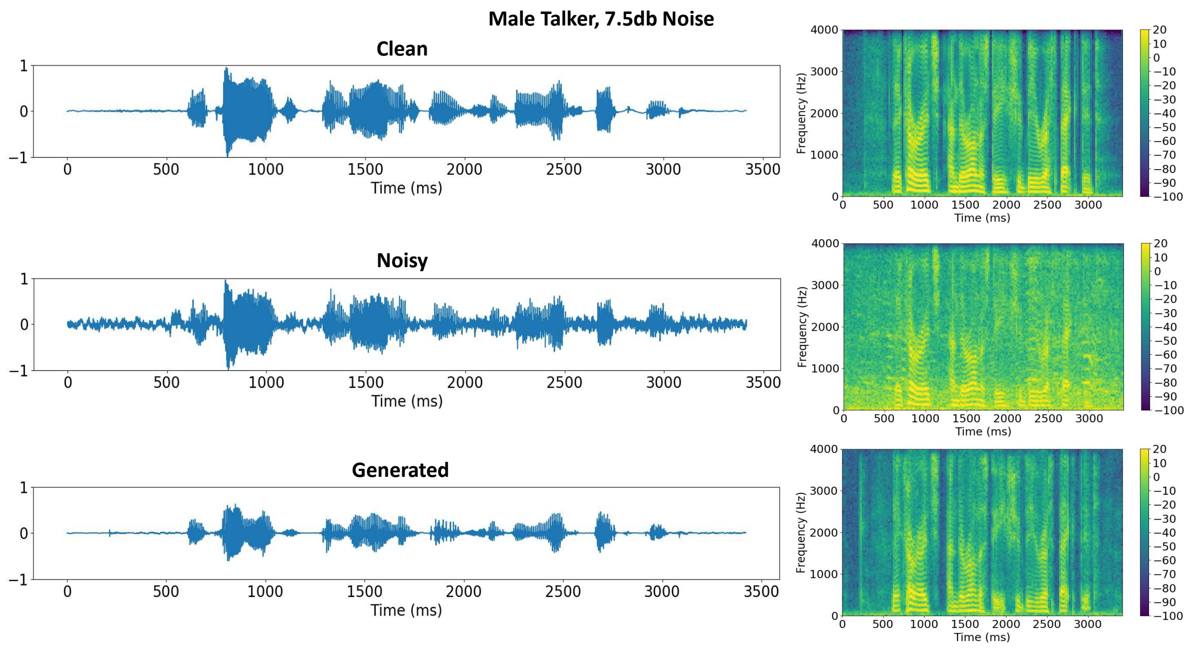


Figure 6.10: **Top**: Clean speech waveform and its spectrogram for a male talker. **Middle**: Public square noise added at 7.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from denoised Delta features as context, and its spectrogram.

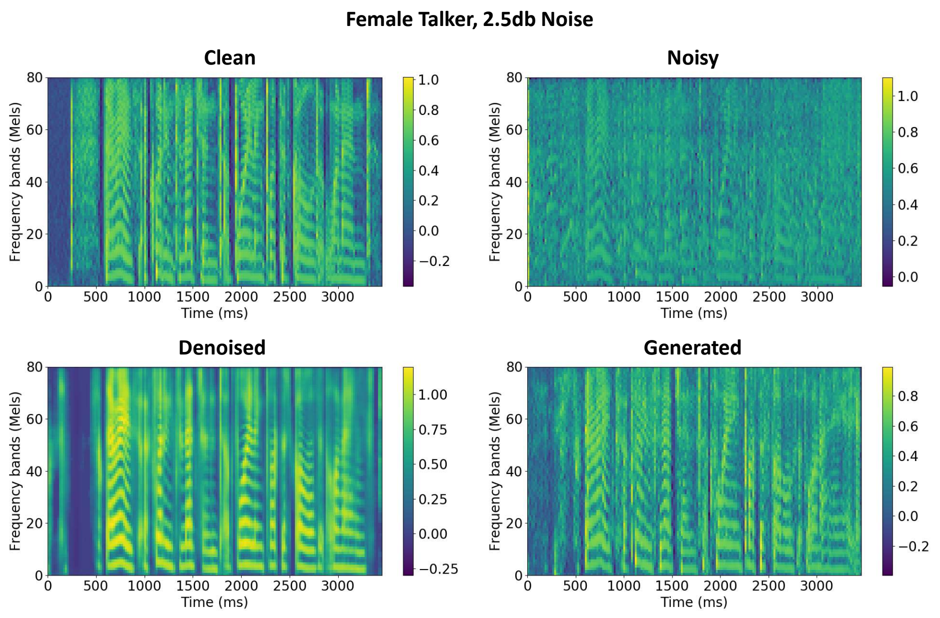


Figure 6.11: **Top Left**: Delta Features of clean speech for a female talker. **Top Right**: Delta features from speech corrupted with cafe noise at 2.5 dB SNR. **Bottom Left**: Denoised Delta features obtained from the VCAE model. **Bottom right**: Delta features of the generated speech for the example.

better than the latter not only in terms of quality but also intelligibility. Note, that no objective metrics were used to determine the quality or intelligibility of generated speech. Observations from both experiments also indicate that the combination of the WaveRNN synthesizer with U-Net was able to capture more details in the synthesized speech. Whereas, speech generated by the WaveRNN model when combined with the VCAE network lacked clarity. Moreover, for instances with higher noise power in the input the system could not reproduce the speaker voice correctly. Hence, the final system comprises of the U-Net architecture with the WaveRNN synthesizer trained with Delta features. For the

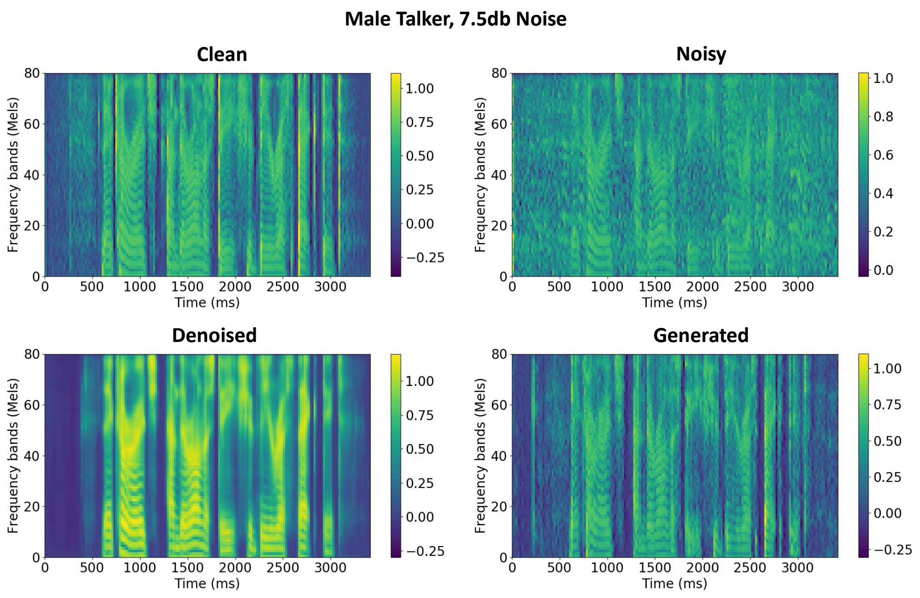


Figure 6.12: **Top Left**: Delta Features of clean speech for a male talker. **Top Right**: Delta features from speech corrupted with public square noise at 7.5 dB SNR. **Bottom Left**: Denoised Delta features obtained from the VCAE model. **Bottom right**: Delta features of the generated speech for the example.

final system, the sampling rate of audio is 16kHz.

6.6.3. The Final System

Based on the results obtained, observations and inferences made from both experiments 6.6.1 and 6.6.2, the U-Net architecture seems to be the most preferred choice. Though the number of parameters and computation in the U-Net is greater than that for the given configuration of the VCAE architecture, the WaveRNN is able to encapsulate more details from features that are denoised by the the U-Net as compared to the VCAE. Thus, for the final experiment, the combination of the WaveRNN with U-Net seems to be a good fit.[3] Apart from the denoising architecture, it is also crucial to choose appropriate features, as inferred from previous experiments in this chapter. Intuition based on listening and observations indicate that speech synthesized using denoised Delta features sound better in terms of quality and are more understandable than speech synthesized using denoised Mel spectrograms. As a result, Delta features have been selected for the final system. This experiment is conducted to evaluate the performance of the selected features and denoising architecture with the WaveRNN synthesizer for 16kHz speech.

The setup for this system is described in table C.11. Some synthesizer hyper-parameters and feature configurations were changed for this experiment as compared to the setup for experiment 4.5.2 in chapter 4. As a result, some hyper-parameters for the initial pre-training of the WaveRNN with clean Delta features also had to be changed. Firstly, the number of Mel bands in the features were increased to 128. The synthesizer network was trained for 8000 epochs (1.6million iterations). In order to alleviate the problem of GRU instability, after 4000 epochs the learning rate was reduced by a factor of 2. Simultaneously, the U-Net was also trained to remove noise from the Delta features obtained from noisy speech. After training both the synthesizer and denoising models, the denoising model is used to fine-tune the synthesizer parameters. The synthesizer is fine-tuned for 12000 epochs (2.4million iterations) and the learning rate was reduced by a factor of 2 after fine-tuning for 8000 epochs. After fine-tuning the synthesizer was complete, results obtained are illustrated in figures 6.2 and 6.3 where, the waveform of the clean, noisy and generated speech along with their respective spectrograms are shown. On listening to the audio files, the following observations were made:

[3]It is unfair to claim that the U-Net model outperforms the VCAE network in general solely based on the results obtained from experiments 6.6.1 and 6.6.2, since the hyper-parameter configurations set during experimentation are not the most optimal ones for the respective networks.

1. No distortions or artifacts heard in generated speech.
2. Voice of the speaker is correctly reproduced. Minor quavering in voice exists in case of extreme noise. The intonation of the speaker is also fairly reproduced but it can vary from the reference when speech is generated from context contaminated with extreme noise.
3. Most of the words are recognizable but some words are still difficult to understand due to mumbled speech especially for certain regions in low SNR samples. There is a drastic improvement in intelligibility as compared to the case when noisy context was directly provided to the synthesizer. However, there is scope for more enhancement.
4. Consonant sounds and some phonemes such as 'f', 'p', 'b', 'd', etc. are fairly reproduced but in case of extreme noise it leads to misidentifying the correct word.

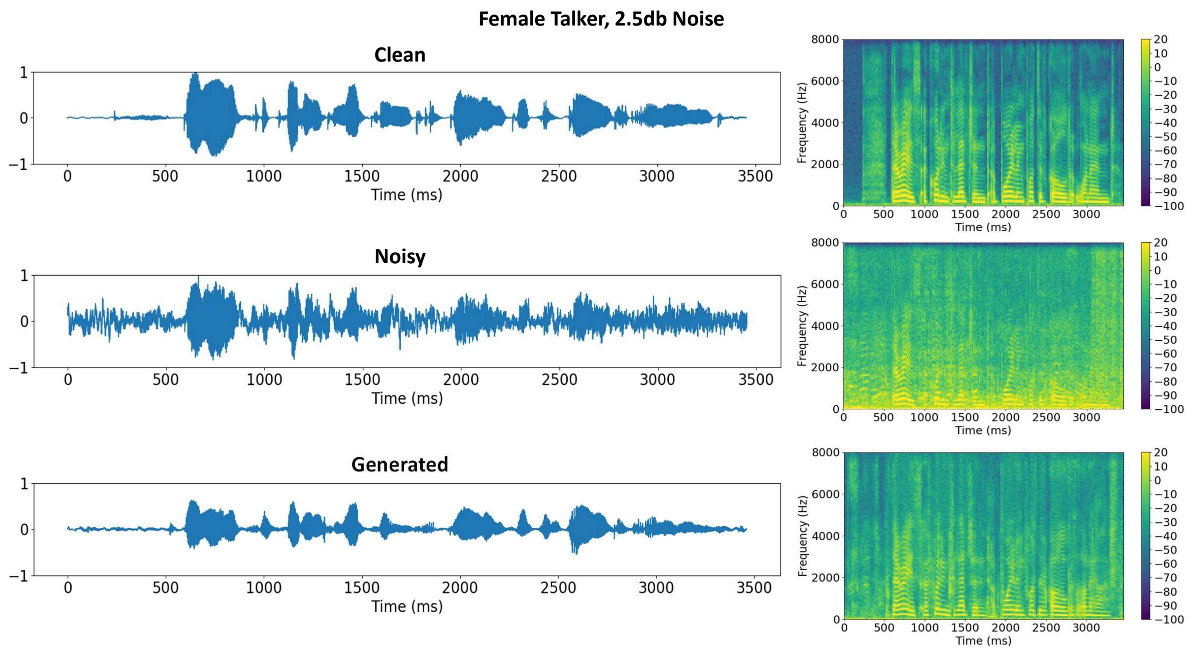


Figure 6.13: **Top**: Clean speech waveform and its spectrogram for a female talker. **Middle**: Cafe noise added at 2.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from denoised Delta features as context, and its spectrogram.

Training the synthesizer for a longer time with a reduced learning rate after convergence can improve stability in the GRU layer. After, convergence, the loss values of the network still keep varying around the true minimum if the learning rate is high. Stopping training at this point could either lead to a sub-optimal or even an inadequate solution. If the learning rate is reduced, then the network still converges towards the minimum and the variance in the loss values is reduced. Hence, decreasing the learning rate caused the GRU layer parameters to be finely updated. As a result, there were no abrupt variations in the hidden state of the GRU layer while generating speech. Hence, there were no observable artifacts present in generated speech as seen before. This solution for preserving the GRU stability during generation does not seem to be very reliable. Later,the cause for GRU instability in previous examples was analyzed and some unusual patterns were observed in the hidden state as discussed in chapter 4 segment 4.6.3. Thus, an alternative solution was suggested for solving this problem.

The correct production of pitch and intonation can be seen in most segments of the spectrogram obtained from generated speech when compared with the clean speech spectrogram. Mumbling of words and production of incorrect consonant sounds is a bit less than before but it still exists as seen in the last few segments of the spectrogram of generated speech in figure 6.13. Here, the sound of consonant in the end of the word is not produced correctly. Figures 6.15 and 6.16 display the reference (clean) Delta features, noisy Delta features (input to the U-Net model) and denoised Delta features

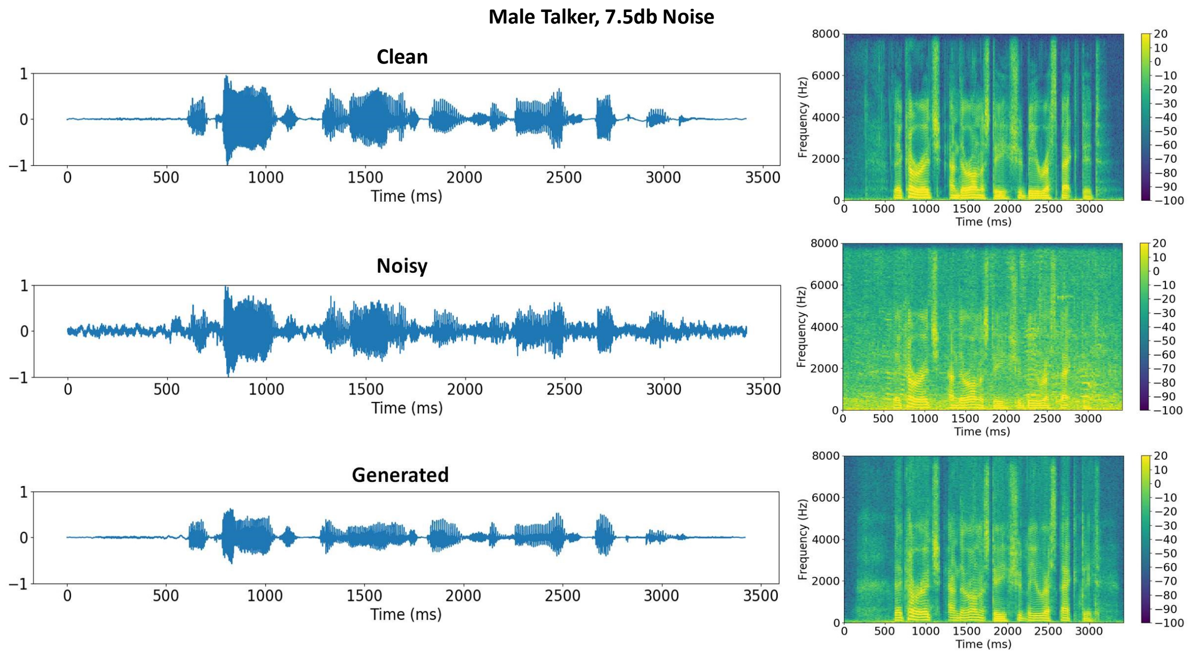


Figure 6.14: **Top**: Clean speech waveform and its spectrogram for a male talker. **Middle**: Public square noise added at 7.5 dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Speech waveform generated from denoised Delta features as context, and its spectrogram.

(output of the U-Net model and input to the WaveRNN synthesizer) and the Delta features obtained from generated speech. The observations made for these figures are not different from the ones made in experiment 6.6.1. Neither were any signs of noticeable improvement in the denoised delta features or delta features obtained from generated speech.

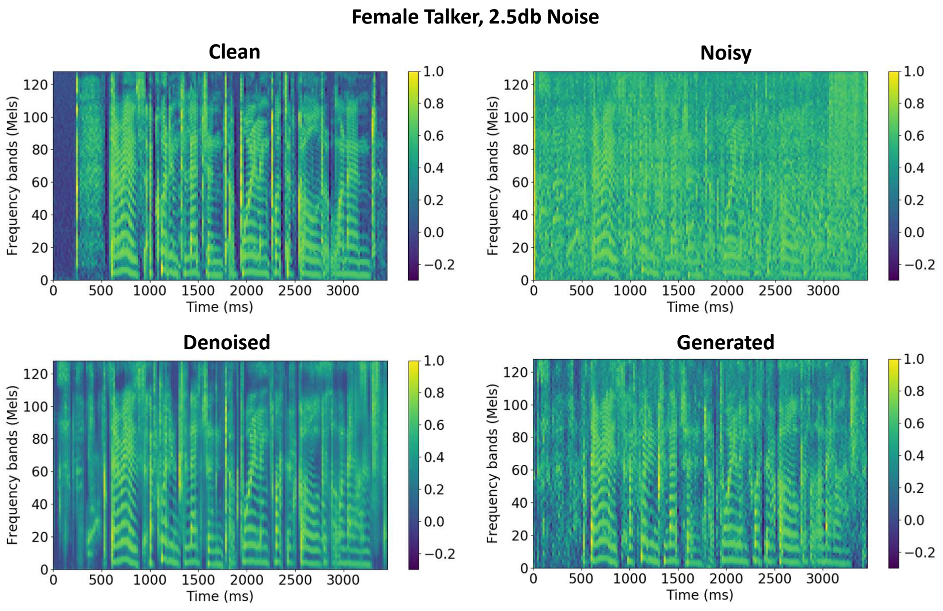


Figure 6.15: **Top Left**: Delta Features of clean speech for a female talker. **Top Right**: Delta features from speech corrupted with cafe noise at 2.5 dB SNR. **Bottom Left**: Denoised Delta features obtained from the U-Net model. **Bottom right**: Delta features of the generated speech for the example.

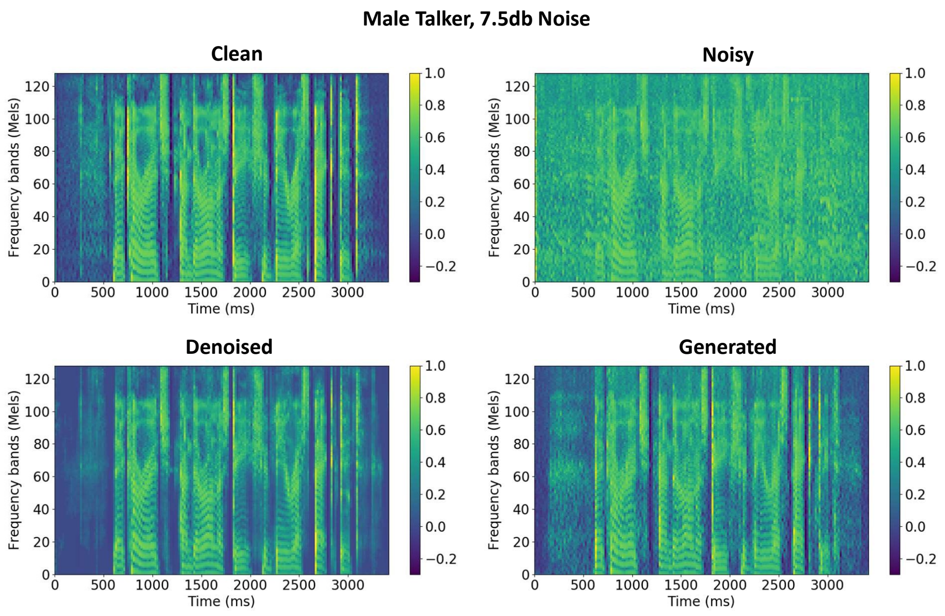


Figure 6.16: **Top Left**: Delta Features from clean speech for a male talker. **Top Right**: Delta features from speech corrupted with public square noise at 7.5 dB SNR. **Bottom Left**: Denoised Delta features obtained from the U-Net model. **Bottom right**: Delta features of the generated speech for the example.

6.7. Analysis

In this segment, an analysis conducted on the the final system described in experiment 6.6.3 is reported. This section is divided into two segments namely, training performance and evaluation of synthesized speech. In the first segment 6.7.1, the training performance of the proposed system is described. Later, the speech synthesized by this configuration was evaluated with a MUSHRA subjective listening test as described in 6.7.2.1 and an objective intelligibility measure called the Word Error Rate (WER) as shown in segment 6.7.2.2.

6.7.1. Network Training Performance

The performance of the network while training is determined by the negative log likelihood (NLL) loss function (described in chapter A section A.2.2.2) with respect to the clean reference audio sample. Figure 6.17 depicts the training loss curve with the loss expressed in bits on the vertical axis and the number of training cycles or epochs on the horizontal axis. This curve has been obtained for initial pre-training phase with clean features and the fine-tuning phase with noisy features. It is observed that both curves converge close to the same point, i.e., roughly around 2.5 bits. This means that a generated sample can be represented by 2.5 bits/sample on an average. When, the fine-tuning phase begins, the loss curve tends to increase a bit but later stabilizes and starts converging towards 2.5 bits again. This indicates that while fine-tuning, the pre-trained weights do not differ by large margins. This further indicates that the domain adaptation performed by the U-Net architecture is successful. There is a strong overlap between the feature spaces of the clean and enhanced Delta features.

6.7.2. Evaluation of Synthesized Speech

Speech generated by the proposed system in experiment 6.6.3 were evaluated for quality using subjective listening tests. The quality of generated speech could not be evaluated using objective measures since they compare the sample alignment with the reference speech. Even though generated speech may sound similar to the reference, their waveforms do not have the same sample alignment as a result such metrics render futile. The intelligibility was also evaluated using a metric generally used for speech recognition, i.e., the Word Error rate. Objective intelligibility metrics for speech enhancement also compare test examples with the reference based on sample alignment.

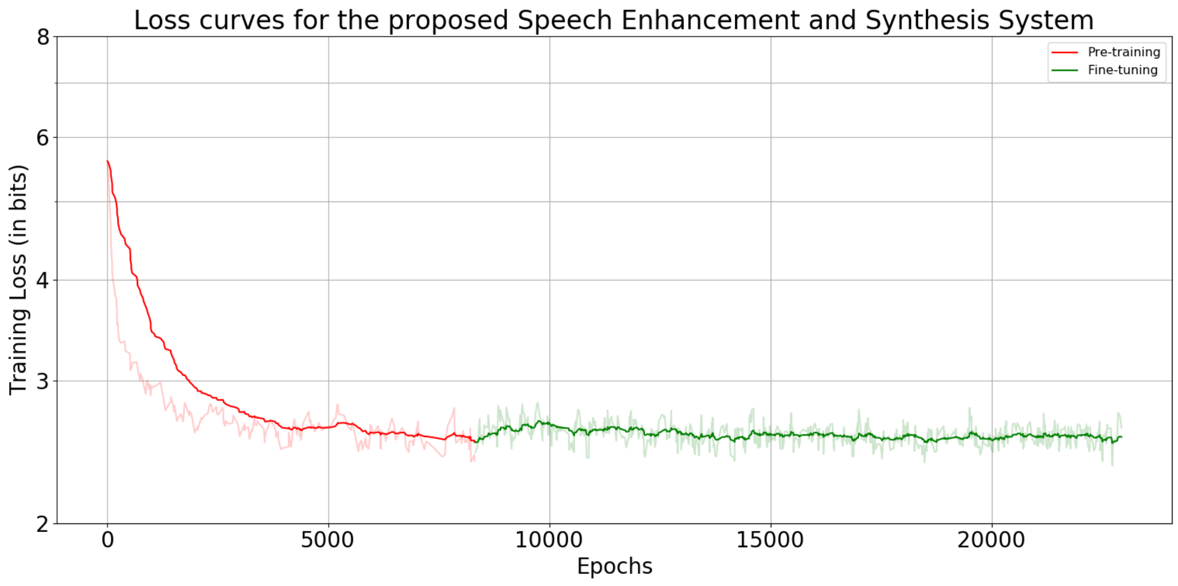


Figure 6.17: Loss curves for the proposed Speech Enhancement and Synthesis System

MUSHRA Test

Speech synthesized by the final system described in experiment 6.6.3 was subjectively evaluated with a MUSHRA listening test. MUSHRA stands for 'MUltiple Stimuli with Hidden Reference and Anchor' and it is a methodology for conducting a listening test to evaluate the perceived quality of speech processed by either enhancement or compression algorithms. It is defined by ITU-R recommendation BS.1534-1 [73]. The main advantage of using MUSHRA over MOS (Mean Opinion Score) are as follows:

1. MUSHRA requires fewer participants to obtain statistically significant results (20 participants are sufficient).
2. The scale of $0 - 100$ used by MUSHRA makes it possible to rate very small differences.

In a MUSHRA test, the listener is presented with the reference and a certain number of test samples which include a hidden version of the refernce and one or more anchors. The purpose of using anchors is to calibrate the scale so that minor artifacts are not unduly penalized. Ideally, MUSHRA tests call for trained experienced listeners who know what typical artifacts sound like [73].

Procedure

Speech files generated from the proposed enhancement system were tested against speech files obtained from several state-of-the-art speech enhancement methods namely SE-GAN [12], SE-VCAE [22] and SE-WaveNet [13]. A web client interface developed using HTML5 and Javascript known as MushraJS [79] was utilized. This web client, mushrajs, is hosted on a virtual compute engine launched on Amazon Web Services (AWS). 20 utterances from the test dataset were randomly selected such that all 4 different Signal-to-Noise Ratio examples exist equally. Moreover, the number of male and female speaker examples were the same. A total of 10 listeners appeared for the test and the ratio of experienced listeners to untrained listeners were $1:1$. During the test, the identity of the files was not revealed to the listeners and their order was shuffled randomly. The following instructions were provided to the listeners before taking the test:

1. Use high quality studio headphones and a good sound-card!
2. Listen through all test files and test sets before you do any ratings to get used to the material.
3. Rate the quality of the test items only compared to the reference on top.
4. There are in total 7 files to be graded and they consist of 1 hidden reference, 1 noisy speech file, 1 anchor file and 4 test examples.
5. Try to rate the overall impression of a test item and don't concentrate on single aspects.

6. Grade according to naturalness of speech sound, good clarity and less noise, distortions and artifacts.

Results

It was a difficult task to assess and analyze the results of the subjective test since many listeners had a different mindset while rating the test files. This was realized by personally interviewing the candidates to know their experience, rating criterion, etc.[4] Only five listeners graded based on an overall impression and the remaining five listeners focused more on how understandable the test speech examples were. Thus, they weighed intelligibility more over quality while scoring the test files. During the interview, candidates revealed that they found it difficult to grade the test files because the noise and distortions present in the noisy examples did not affect their perception and understanding of speech. Although there were some test files that sounded clean, they did not give a high score because they found it difficult to understand certain words.

Table 6.1 displays the average scores with confidence intervals calculated over all SNR's and noise types. Results from the MUSHRA test were obtained by taking an average of the five listeners who graded based on an overall impression. The table shows that all methods were graded higher than noisy speech on an average. Though the generated speech files from proposed WaveRNN method sound better than the noisy speech, it is graded lower than other state-of-the-art enhancement methods. The reason for such a performance is that the presence of distortions in the speech files obtained from other techniques were not perceived as unpleasant by the listeners. On the contrary, speech files generated by WaveRNN sounded free from artifacts but contained some examples with missing content or unclear words which was not appealing to the listeners.

Table 6.2 shows the average scores with confidence intervals calculated over different SNR's. For lower SNRs ,i.e., 2.5 dB and 7.5 dB, the proposed WaveRNN method performs at par with other enhancement techniques and outperforms the SE-WaveNet method. For higher SNRs, i.e., 12.5 dB and 17.5 dB, other methods outperform the proposed WaveRNN. VCAE consistently outperforms other enhancement methods at lower SNRs and WaveNet performs best at high SNRs. At lower SNRs, distortions present in the enhanced speech estimated from other methods are more prominent and sounds unpleasant to the listeners. Whereas the proposed method sounds free from distortions but misses out on information. Table 6.3 shows the average scores with confidence intervals calculated over different noise types. For noise types 'Living' and 'Public Square' the proposed technique scores almost similar to other methods. However, for noise types 'Cafe' and 'Bus', the proposed method does not perform very well. VCAE performs best for 'Living' and 'Cafe' sounds whereas WaveNet performs best for 'Public Square' and 'Bus'.

Table 6.1: Average scores with confidence intervals (across all SNR and noise type) obtained from MUSHRA listening test.

Noisy	SEGAN	VCAE	Wavenet	WaveRNN
45.81 ± 7.17	58.48 ± 12.95	60.83 ± 13.50	58.66 ± 17.43	53.19 ± 9.06

Table 6.2: Average scores with confidence intervals (split by SNR) obtained from MUSHRA listening test.

SNR (dB)	Noisy	SEGAN	VCAE	Wavenet	WaveRNN
2.5	38.48 ± 4.18	46.04 ± 11.40	49.64 ± 14.26	40.88 ± 10.28	40.52 ± 5.61
7.5	43.56 ± 4.42	56.36 ± 9.72	57.44 ±7.40	53.12±13.79	56.16±4.40
12.5	52.48±7.13	68.00±10.80	70.08±10.53	64.56±12.41	58.52±6.91
17.5	48.72±4.06	63.52±10.55	66.16±13.54	76.08±11.70	57.56±4.55

Figure 6.18 shows the count of the ranks received by a particular technique where 1 stands for the highest rank and 5 is the lowest rank. The Speech enhancement VCAE outperforms all other methods where out of 20 test examples, 11 examples received the highest rank. The proposed WaveRNN technique stands 4^{th} overall and has three examples that ranked first among other techniques. This shows that at the moment partially generative methods outperform the existing regressive methods.

[4]The identity of candidates is not revealed due to privacy concerns.

Table 6.3: Average scores with confidence intervals (split by noise type) obtained from MUSHRA listening test.

Noise type	Noisy	SEGAN	VCAE	Wavenet	WaveRNN
Living	42.75±4.07	55.90±10.55	59.25±13.54	49.95±11.70	52.00±4.54
Public square	42.80±5.60	52.50±10.25	54.73±8.97	55.30±17.24	51.28±10.79
Cafe	48.63±7.43	66.43±13.38	67.10±12.34	63.87±15.28	54.87±6.50
Bus	55.50±6.36	63.70±0.42	69.6±8.20	73.90±3.25	58.20±8.49

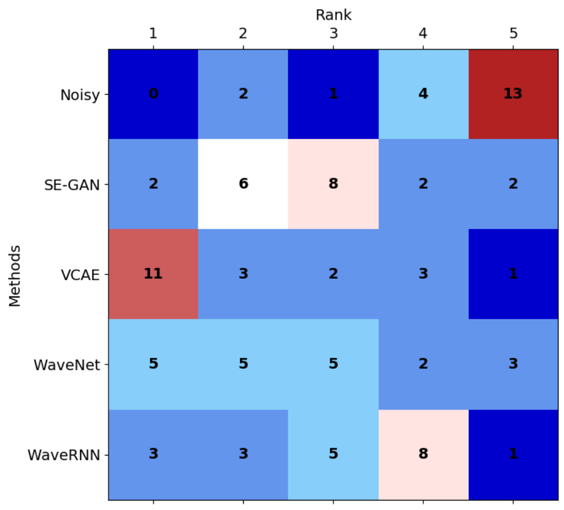


Figure 6.18: Rank count for the considered speech enhancement methods obtained from the MUSHRA listening test.

There is quite a large scope of improvement for fully generative speech enhancement models especially in terms of intelligibility.

Word Error Rate

Word error rate (WER) is a common metric of performance mainly for speech recognition systems. There exist several intelligibility metrics for speech such as Speech intelligibility index (SII), Short Time Objective Intelligibility (STOI) [80], etc. However, these metrics also compare the test waveform with a reference waveform. Speech that is generated by the WaveRNN synthesizer has an infinitesimal chance of having the same sample alignment in its waveform as the reference even if both sound very similar. Thus, any objective metric that directly compares waveforms will not give reliable results for speech obtained from fully generative models. Hence, WER was used as a metric to measure intelligibility. Speech-to-text models are readily available on Google Cloud Services where audio files can be uploaded and text files are returned. The VCTK dataset described in chapter 3 section 3.1 also provides text files for their speech and they can be used as a reference to compare with. Thus, the clean, noisy and generated speech files obtained from the proposed system that were used for the MUSHRA analysis were converted to text with Google's Speech-to-text engine. This was followed by some text pre-processing after which the WER was calculated. The pre-processing steps include removing punctuation symbols, making all letters lowercase and extracting the words from the string to a list.

The calculation of WER is derived from a widely used metric in information theory for linguistics called the Levenshtein's distance [81]. This metric is used for measuring the difference between two string sequences. The Levenshtein's distance between two sentences is the minimum number of word edits (insertions, deletions or substitutions) required to change one sentence into another. The Levenshtein's distance between two strings a and b (of length $|a|$ and $|b|$ respectively) is given,

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j)+1 \\ \operatorname{lev}_{a,b}(i,j-1)+1 \\ \operatorname{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise,} \end{cases} \tag{6.1}$$

where, $\mathrm{lev}_{a,b}(i,j)$ stands for the Levenshtein's distance between the words i and j present in strings a and b respectively. The first element of the minimum corresponds to deletion, the second represents insertion and the third stands for match or mismatch. $1_{(a_i \neq b_j)}$ is the indicator function which equals to 0 when $a_i = b_j$ and equals to 1 otherwise. Thereafter, the word error rate is calculated as:

$$\mathrm{WER(a,\ b)} = \frac{\mathrm{lev}_{a,b}(a,b)}{|a|}, \tag{6.2}$$

where, $\mathrm{WER(a,\ b)}$ stands for the word error rate between two strings a and b. Assuming, a as the reference string, the Levenshtein's distance metric between the strings is divided by the length of the reference string.

The percent average WER calculated over all the same examples used for the MUSHRA test are showcased in tables 6.4. The table shows an average increase by approximately 7% in the Word Error Rate from noise speech to generated speech. This indicates an overall improvement in intelligibility, but, the increase is minute. There is an error of 13.3% between the text response of the clean speech and the reference text provided by the dataset. This shows that the speech-to-text engine is also prone to errors. The speech-to-text engines may also perform some enhancement on the input signal before recognizing the text as many noisy examples have a fairly decent word prediction and the error rate is not very high as expected.

Table 6.4: Percent Average Word Error Rate (across all SNRs and noise types) obtained from the text response of the clean, noisy and generated speech.

Reference vs. Clean	Reference vs. Noisy	Reference vs. Generated	Clean vs. Noisy	Clean vs. Generated
13.3	35.11	28.19	30.16	26.46

Table 6.5: Percent Average Word Error Rate (split by SNR) obtained from the text response of the clean, noisy and generated speech.

Signal-to-Noise Ratio (dB)	Reference vs. Clean	Reference vs. Noisy	Reference vs. Generated	Clean vs. Noisy	Clean vs. Generated
2.5	5.71	32.86	22.86	30	25.71
7.5	15	45	27.5	50	27.5
12.5	20.83	31.25	31.25	18.37	24.49
17.5	16.67	33.33	36.67	23.33	30

Table 6.6: Percent Average Word Error Rate (split by noise type) obtained from the text response of the clean, noisy and generated speech.

Noise Type	Reference vs. Clean	Reference vs. Noisy	Reference vs. Generated	Clean vs. Noisy	Clean vs. Generated
Living	18.75	46.88	21.88	33.33	21.21
Public Square	10.13	36.71	27.85	37.18	30.77
Cafe	22	40	42	30	32
Bus	0	13.33	20	13.33	20

On analyzing the WER for different SNR's as shown in table 6.5, there is an improvement in intelligibility for low SNRs, i.e., 2.5 dB and 7.5 dB since the error reduces. However, when noise power is low, i.e., 12.5 dB and 17.5 dB, then there is either no observed change in the WER metric or a decrease in intelligibility. This indicates that the proposed system is robust to noise as there is a noticeable improvement in case of low SNR signals. On analyzing different noise types as shown in table 6.6, there is an improvement in intelligibility for 'Living' and 'Public Square' noise types. It is difficult to conclude for noise types 'Cafe' since the scores are almost at par. For the noise type 'Bus', the intelligibility decreases in generated speech. This is an unusual phenomenon since, the noisy features were least affected by this noise type as indicated in chapter 3 section 3.9 and yet the system reduces the intelligibility in such examples. This indicates that when the synthesizer becomes robust to noise, it deteriorates the intelligibility in speech that is either almost clean or least affected by noise.

The detailed results are displayed in Appendix B table B.1 where certain striking examples contaminated with 4 different noise types at 4 SNRs are considered. The reference text file that is provided within the dataset and the text files for clean, noisy and generated speech obtained from the speech-to-text engine are also shown so as to justify the numbers obtained for the WER. Words in the text files of clean, noisy and generated speech that are different from the reference have been highlighted. The last column named 'comments' contains inferences and remarks for all the examples. There are some examples with differences between the text response for the clean speech (predicted by the Speech-to-text engine) and the reference text provided by the dataset. Objectively, the system shows a minor increase in intelligibility from noisy speech to generated speech. However, on subjectively evaluating the text responses, shown in table B.1, by considering the semantics of the sentences, speech generated by the proposed system can sound confusing as reflected in the MUSHRA listening tests.

Concluding Remarks

The proposed speech enhancement model is a combination of the WaveRNN speech synthesizer described in chapter 4 and the U-Net architecture described in chapter 5. The U-Net is responsible for enhancing noisy Delta features by removing the noise and the WaveRNN generates speech from the context provided by denoised Delta features. This chapter discussed the training, fine-tuning and speech generation procedures. Experiments and analysis conducted with the system led to important findings. This section summarizes the key takeaways from the topics discussed in this chapter.

The following inferences have been drawn from this chapter:

- A fully generative neural network model for producing clean and natural sounding speech from context provided by noisy features was successfully implemented.
- Using Delta features over Mel spectrograms yield better sounding speech since Delta features present the network with information regarding consonants and unvoiced speech sounds much more explicitly. Thus, the WaveRNN synthesizer can capture those sounds effectively.
- Speech generated from features enhanced by the U-Net architecture sound better than the speech generated from features enhanced by the VCAE network. This is because the U-Net can encapsulate more details as compared to the VCAE network yielding more enhanced features. Hence, the proposed method for speech enhancement involves the combination of the U-Net architecture with the WaveRNN synthesizer.
- Speech generated by the proposed method described in experiment 6.6.3 is clean and natural sounding and does not contain any noise characteristics, distortions or artifacts. However, it suffers in terms of intelligibility due to missing clarity in some words. It must be noted that, there is considerable improvement in intelligibility as compared to providing noisy features directly to the synthesizer (described in chapter 4).
- Results from the MUSHRA subjective evaluation reveal that the proposed system does produce speech that is preferred over listening to noisy speech. However, other state-of-the-art speech enhancement methods outperform the system proposed. Reasons for this are attributed to the lack of enhancement in terms of intelligibility.
- Intelligibility in generated speech was also evaluated objectively with the Word Error Rate (WER) measure. The metric shows an improvement over noisy speech on an average. However, if semantics in the text responses are also taken into consideration then it can be difficult to understand the generated speech which is also reflected in the MUSHRA test. Analysis indicates that the system is robust to noise but deteriorates for less noise in input.
- There is a need to develop an objective metric to evaluate the quality of the speech produced by generative models. The existing metrics are not so helpful since they compare the sample alignment of test examples with the reference.

7

Conclusion & Discussions

This chapter summarizes the important outcomes of this study and throws light on future advancements in this field of research. It is organized as follows where first a discussion section reflects upon the proposed speech enhancement method in segment 7.1.1. This is followed by another discussion in segment 7.1.2 where the applicability of this technique on a hearing aid system is highlighted. Thereafter, a synopsis of the most salient concluding points are provided in section 7.2. Finally, the future prospects and possibilities for further improvements are mentioned in section 7.3.

7.1. Discussions

This section is divided into two categories where first the research goals of this thesis that were mentioned in chapter 1 section 1.2 are reflected upon. This segment answers the research questions that were the driving force behind this study. The following segment enlists the obstacles of implementing the proposed speech enhancement method on a hearing aid system in terms of the speech information they process and hardware constraints.

7.1.1. Proposed Speech Enhancement technique

This segment aims to address the research questions that were stated in the introductory chapter of this thesis. This discussion helps evaluate the extent to which the goals of this thesis are met.

How can clean sounding and intelligible speech be generated when the context provided to a generative model is noisy?

The generative model chosen for the task of speech enhancement is called WaveRNN and the reasons for selecting this network were mentioned in the concluding remarks from chapter 2. From the experiments conducted in chapter 4, it was concluded that when noisy context is directly provided to the WaveRNN synthesizer it can produce clean sounding speech but this speech is not meaningful at all. Moreover, the model could not identify the correct speaker voice as it kept varying for different segments within the same utterance. This occurred because of the presence of noise in the context which interfered with the speech information. Furthermore, the synthesizer was not trained to neglect or rather remove noise from the context. As a result, the noisy features had to be enhanced before being presented to the WaveRNN synthesizer. This enhancement of features was performed separately for which two different architectures namely the U-Net and the Variance Constrained Auto-Encoder (VCAE) were proposed in chapter 5. With a series of experimentation, it was unveiled that the U-Net model outperformed the VCAE network for the selected set of configurations. Hence, the pre-trained U-Net model was deployed to fine-tune the parameters of the WaveRNN synthesizer. For faster convergence of the model, the synthesizer was initially pre-trained with clean context so as to learn the transformation from context to speech. Thus, by using the principles of transfer learning, the WaveRNN also learned to generate clean and natural sounding speech from denoised features as described in chapter 6. Thus, this combination of the WaveRNN synthesizer with the U-Net model could now successfully accept noisy context to produce desired speech. This led to a drastic improvement as compared to the results obtained from directly synthesizing speech from noisy features not only in terms of intelligibility but also in artificially producing speech with the same voice and intonation of the speaker. However, speech generated from denoised features was still not perfect as there were some segments within a few examples that lacked clarity and led to misunderstanding the correct words. In order to further improve the intelligibility in these systems, some recommended changes are summarized in section 7.3.

What are the functionalities of the key components used in the proposed generative model?

The analysis sections in chapters 4 and 5 investigate the role of the components in the proposed system by visualizing their outputs. For instance, the most important component of the WaveRNN is the Gated Recurrent Unit (GRU) layer. The hidden state of the GRU layer was visualized by stacking it along the temporal axis. Repetitive patterns in this state were observed and the frequency of repetition of these patterns aligned with the pitch frequency of the speaker. Thus, one of the most important aspects of the GRU layer is to encapsulate the speaker identity. The GRU layer also exhibits the presence and absence of speech in its hidden state. Moreover, the instability in the GRU layer while generating a waveform is also reflected within its hidden state. A large change in the variance of the values is observed in case of instability.

The output of the WaveRNN synthesizer was also visualized which displayed the probability distribution of the sample value. When the probability distribution is stacked along the temporal axis, it shows a hazy resemblance with the speech waveform. Moreover, the WaveRNN was highly confident about samples belonging to voiced speech segments since the distribution was very narrow whereas for unvoiced speech segments or consonants, the distribution was very wide. Thus, the uncertainty in predicting an unvoiced speech sample is much higher than that for a voiced speech sample. The average entropy rate is calculated to be approximately 2.5 bits/sample or 40 kbits/second for all audio files.

The weights of the WaveRNN synthesizer are distributed as a super-Gaussian or a factored Laplacian. This indicates two properties of the network, first being that the network learned the distribution of speech by itself, and secondly, the weights of the network are sparse. Hence, pruning can be applied in order to reduce the number of computational resources even further and hence reduce the time in operations. The convolution kernels in the up-sampling network were also visualized. The kernels apply a low pass filter on the Mel features and the stretching operation increases the temporal resolution of the Mel features thus providing the features to the WaveRNN at the sampling frequency of audio. Due to the use of a convoltuional operation, the up-sampling network makes the generation process of the overall system slower. At the moment, the system needs information up to 3 time segments of Mel feature vectors in order to generate speech samples. Please note, that the WaveRNN on its own does not need future information since it only needs the upsampled Mel feature vector at the current index and the previous speech samples.

The latent space of the denoising architectures were also visualized in order to understand the role of U-Net and VCAE architectures. The latent space of the U-Net is much larger than the input to the network. On the other hand, the latent space of the VCAE is compressed. The latent space of the U-Net is able to encapsulate detailed information from the input such as speech activity, silence regions, information at different frequency ranges and speech onset all in different channels. This is also reflected in the output of the U-Net, where the denoised Mel features can preserve speech related details. The latent space of the VCAE is difficult to understand since there are no distinct patterns observed in the channels. Activation at different time-band bins align with the speech activity in the input features but apart from that no other details were interpretive. The features enhanced by the VCAE network level the power in the speech regions especially in the mid and high frequency bands. Thus, it is not able to capture all details effectively in such a small space.

7.1.2. How far is this technology from a hearing aid system?

Chapters 4 and 5 mentions the amount of computations and memory requirement along with the time needed for training and execution of this system. Recall from chapter 4 section 4.6.1, the WaveRNN synthesizer requires 2.61 MFlops to compute one sample of speech. Thus, to produce 1 second of speech at a sampling rate of 16 kHz, the processor must be capable of carrying out approximately 41.76 GFlops/second. Additionally, the denoising architecture requires 1.91 GFlops to denoise a frame of noisy Mel features extracted from 540 ms of speech. Thus, to denoise a frame of 1 second of speech, the process requires approximately 3.54 GFlops/second. With the given configurations, it is possible to execute the proposed system on a Samsung Galaxy Note 8 or Google Pixel 2 XL that use the Snapdragon 835 cpu as they are capable of conducting the required number of flops/second using all 8 cores [82, 83].[1] Pruning the weights of the network can considerably reduce the computations to using at most 2 cores on the CPU [21]. A hearing aid device uses a Digital Signal Processor (DSP) and the resources available are not capable of carrying out such a large number of computations. However, a mobile phone is also a part of hearing aid systems now-a-days and the proposed speech enhancement system can be implemented on a mobile that can communicate with a hearing aid. Apart from computational power, another worrisome factor is the delay in hearing due to the latency of the system.

Perceptual studies indicate that the tolerable delay at which listeners report being annoyed or bothered by the change in sound quality is as low as 10 ms. This delay depends on the frequency of the sound and the speaker. For speech from other speakers, the tolerable delay is reported to be $24 - 30$ ms when it is constant across frequency and about 15 ms when it is variable [84]. For a person's own voice, the tolerable delays are $9 - 10$ ms [84]. With the current configuration settings mentioned in table C.11, the proposed system, requires at most 80 ms of noisy speech to start processing it. This delay can be reduced by decreasing the frame length and hop length of the window under consideration. On top of that, the processing time of the system on a GPU is approximately 0.625 ms/sample as mentioned in chapter 4 segment 4.6.1. This means that with the current setup, the synthesizer needs 10 seconds to generate 1 second of speech. The generation time can be considerably reduced by pruning the weights of the network to generate speech at the desired sampling frequency [21, 53, 85]. Assuming, if the system were to be implemented on a mobile phone after pruning the network weights and reducing the frame and hop lengths, there will still be an additional lag due to communication between the devices which increases the overall latency.

In order to reduce the delay in synthesis, if the frame length is reduced to 10ms or lower, then the features extracted from such a short segment are very different from the features that were used to train the proposed system. It will be difficult for the network to learn the pitch information from these wide-band spectral features. Hence, it will have to be estimated separately from the correlation of the signal. Other features such as the cepstrum, or delta cepstrum will have to be extracted separately and appended with the pitch before waveform synthesis. This will also bring a change in the system design.

7.2. Conclusion

Every chapter in this thesis includes concluding remarks at its end and these remarks have formed a foundation for the subsequent chapters. This section gives a synopsis of the most cardinal outcomes of this study. Chapter 4 introduced the WaveRNN synthesizer, a fully generative network that models audio as a conditional probability distribution depending on the previous audio samples and the context provided to it. As far as the context is clean, the quality and intelligibility of generated speech is extremely high and sounds almost indistinguishable from the reference speech. This is irrespective of the type of features used as context. Hidden state in the Gated Recurrent Unit is capable of capturing the voice identity of a speaker since repetitive patterns synchronous with the pitch frequency of the speaker are observed. The weights of the network follow a super Gaussian distribution which is similar to the distribution of speech. Hence, the network learns the natural distribution of speech unlike classical approaches that rely on a Gaussian assumption to analytically derive their models.

The presence of noise in the context does not affect the quality of the synthesized speech as it sounds very natural. However, generated speech lacks intelligibility due to mumbled words and difficulty in reproducing consonant sounds. Simply tuning network hyper-parameters is not the key to improve the model performance and can not alleviate the problem when noisy context is provided. A good understanding of input features is required. As a result, denoising or enhancing the features separately before they are provided to the WaveRNN synthesizer seems to be a more practical solution. Moreover, instead of training the synthesizer ground up on denoised features as context, it is beneficial

[1]No tests have been conducted to verify this and it is a ballpark estimate.

to fine-tune the WaveRNN that is pre-trained on clean context. This is inspired from a principle called Transfer Learning. In this way, the entire system remains generative in nature as there is no comparison of waveforms.

Two architectures for denoising the features were introduced in chapter 5. It was difficult to make a choice between the architectures based on their denoising performance. Later, after fine-tuning the synthesizer with features obtained from both architectures indicated that speech generated from features enhanced by the U-Net architecture sound better than the speech generated from features enhanced by the VCAE network (for the given configuration). This is because the U-Net can encapsulate more details as compared to the VCAE network yielding more enhanced features. Hence, the proposed method for speech enhancement involves the combination of the U-Net architecture with the WaveRNN synthesizer as shown in chapter 6.

Speech generated by the proposed method described in experiment 6.6.3 is clean and natural sounding and does not contain any noise characteristics, distortions or artifacts. However, it suffers in terms of intelligibility due to missing clarity in some words. It must be noted that, there is considerable improvement in intelligibility as compared to providing noisy features directly to the synthesizer (described in chapter 4). Results from the MUSHRA subjective evaluation reveal that the proposed system does produce speech that is preferred over listening to noisy speech. However, other state-of-the-art speech enhancement methods outperform the system proposed. Reasons for this are attributed to the lack of enhancement in terms of intelligibility. Intelligibility in generated speech was also evaluated objectively with the Word Error Rate (WER) measure. There is a noticeable improvement in intelligibility over noisy speech with low SNR but the intelligibility drops for high SNR signals. The metric shows a slight improvement over noisy speech on an average over all SNRs and noise types. However, if semantics in the text responses are also taken into consideration then it can be difficult to understand the generated speech which is also reflected in the MUSHRA test.

In chapter 3, a modification was proposed while calculating the Delta features. A new parameter is introduced to control the difference between the successive bands. This not only amplifies the power during speech onsets or unvoiced speech segments but also preserves the harmonics in voiced speech segments. Sigmoidal normalization technique was introduced to avoid the clipping problem that arises with the min-max normalization. An analysis conducted to study the effect of noise on different features in the Mel domain revealed that the Mel Frequency Cepstral Coefficients are most affected by the introduction of noise irrespective of the noise type and its power. The analysis did not show a vast difference between Mel spectrograms and the proposed Delta features. In chapter 6, using Delta features over Mel spectrograms yield better sounding speech since Delta features present the network with information regarding consonants and unvoiced speech sounds much more explicitly.

7.3. Future Work

There are a lot of possible directions of advancements from this point forward. Some of these are mentioned in this section.

Improve Generation Stability in WaveRNN

In chapter 4, there were some cases when the generated audio suffered from the presence of artifacts. These artifacts occurred randomly and their existence was attributed to the instability in the GRU layer while generating audio. It was suspected that such situations can occur if the WaveRNN is not trained sufficiently or trained longer with a large learning rate despite convergence. This problem was solved by reducing the learning rate of the model after convergence during training. Thereafter, the production of speech with artifacts seldom occurred. However, it is believed that this problem has only been solved partially by tweaking the hyper-parameter. Later, on analyzing the samples with artifacts, it was observed that the hidden state of the GRU layer had eccentric patterns that were different from the ones indicating speech presence. The variance in the hidden state of that particular time slice was observed to be larger than usual. Hence, a solution is inspired from the VCAE network to enforce a bound on the variance of the hidden state while training. It is believed that with such a constraint, this instability would be curbed and not occur again. It would be interesting to attempt training the WaveRNN synthesizer with this penalty function alongside the objective function.

Add an Intelligibility Metric to the Objective

The MUSHRA listening tests in chapter 6 indicated that the speech generated by the proposed speech enhancement system did not perform as well as other state-of-the-art speech enhancement models. The reason for such a performance was due to lower intelligibility of generated speech as compared to other methods. Thus, one of the ways to improve intelligibility would be to add a penalty function that encourages intelligibility to the objective minimization. Some existing objective intelligibility metrics compare the features of the generated speech with that of the clean reference speech. One of the ways of doing so would be to design an additional network that either maps the output or the latent space of the denoising architecture to a space of $1/3^{rd}$ octave bands. Thereafter, the STOI metric [80] can be maximized to improve the correlation between the $1/3^{rd}$ octave bands. This can also help in retrieving the high frequency information that was lost while predicting the enhanced features in chapter 5. The suggestion of this penalty function has not been mathematically verified to check if the function is differentiable. However, it would be an interesting topic of research since no cost function has been proposed to improve speech intelligibility.

A Neural Network based Objective Speech Quality Metric

One of the major problems with generative speech synthesizers is to be not able to measure their quality with objective metrics. The existing objective metrics such as Segmental SNR, Percptual Evaluation of Sound Quality (PESQ), etc. render futile because they compare the test example with the reference and most of them check for sample alignment. Since, generated speech despite having the same context as the reference speech does not have a similar waveform or sample alignment, such metrics give a low score. One of the ways of developing an objective metric for evaluating quality can be done by designing a discriminative neural network that compares a feature encoding derived from clean reference speech and test examples using a triplet loss [86]. Inspired by the DeepFace [87] research on face verification, a similar Siamese network architecture can be redesigned for speech quality evaluation. A subjective Mean Opinion Score metric [88] can be used for labeling the data.

Generate Speech from Deep Features

Chapter 3 section 3.8 discussed the possibility of using self-derived features by a neural network instead of providing manually extracted features. This allows the network to learn its own meaningful representations from raw wave inputs. An auto-encoder architecture, for instance, the VCAE could be used to map the input waveform to a compressed latent space. This latent space can then be provided to the WaveRNN synthesizer to generate speech as illustrated in figure 7.1. Perhaps, the idea proposed in segment 7.3 can also be combined with this proposition.

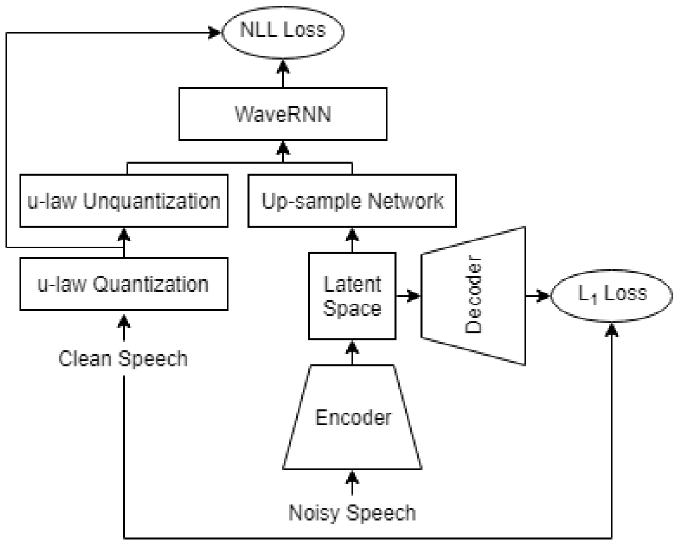


Figure 7.1: Learning to generate speech from a Latent Space

Towards Faster Implementation on Devices with Hardware Constraints

The computational analysis in chapter 4 showed that the network requires a significantly large number of computations to predict a sample. Despite, it being able to run on a mobile CPU, the generation speed of speech samples does not match the sampling frequency of audio. In order to generate speech at that rate, the number of computations must be reduced and this can be achieved by pruning the weights of the synthesizer model. The distribution of the synthesizer network weights show that the matrix is sparse in nature and many weight values can be pruned in order to reduce the model size further. Pruning the weights of the WaveRNN model has already been implemented in [21], however, the challenge here is to do so when noisy or enhanced features are an input to the synthesizer such that there is minimum loss in generated speech quality. An advancement in research towards this direction would probably make neural networks fit on a hearing aid system in the near future.

Bibliography

[1] S. M. McOlash, R. J. Niederjohn, and J. A. Heinen. "A spectral subtraction method for the enhancement of speech corrupted by nonwhite, non-stationary noise". In: *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*. Vol. 2. 1995, 872–877 vol.2.

[2] A. A. Azirani, R. Le Bouquin Jeannès, and G. Faucon. "Speech enhancement using a wiener filtering under signal presence uncertainty". In: *1996 8th European Signal Processing Conference (EUSIPCO 1996)*. 1996, pp. 1–4.

[3] O. Cappe. "Elimination of the musical noise phenomenon with the Ephraim and Malah noise suppressor". In: *IEEE Transactions on Speech and Audio Processing* 2.2 (1994), pp. 345–349. issn: 1063-6676. doi: 10.1109/89.279283.

[4] J. Sohn, N.S. Kim, and W. Sung. "A statistical model-based voice activity detection". In: *IEEE Signal Processing Letters* 6.1 (1999), pp. 1–3. issn: 1070-9908. doi: 10.1109/97.736233.

[5] R. Martin. "Noise power spectral density estimation based on optimal smoothing and minimum statistics". In: *IEEE Transactions on Speech and Audio Processing* 9.5 (2001), pp. 504–512. issn: 1063-6676. doi: 10.1109/89.928915.

[6] T. Gerkmann and R. C. Hendriks. "Unbiased MMSE-Based Noise Power Estimation With Low Complexity and Low Tracking Delay". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.4 (2012), pp. 1383–1393.

[7] R. C. Hendriks, R. Heusdens, and J. Jensen. "MMSE based noise PSD tracking with low complexity". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 4266–4269.

[8] Y. Xu et al. "A Regression Approach to Speech Enhancement Based on Deep Neural Networks". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.1 (2015), pp. 7–19.

[9] Se Rim Park and Jinwon Lee. "A Fully Convolutional Neural Network for Speech Enhancement". In: *CoRR* abs/1609.07132 (2016). url: http://arxiv.org/abs/1609.07132.

[10] K. Osako, R. Singh, and B. Raj. "Complex recurrent neural networks for denoising speech signals". In: *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 2015, pp. 1–5.

[11] Jean-Marc Valin. "A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement". In: *CoRR* abs/1709.08243 (2017). arXiv: 1709.08243. url: http://arxiv.org/abs/1709.08243.

[12] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. "SEGAN: Speech Enhancement Generative Adversarial Network". In: *CoRR* abs/1703.09452 (2017). arXiv: 1703.09452. url: http://arxiv.org/abs/1703.09452.

[13] Dario Rethage, Jordi Pons, and Xavier Serra. "A Wavenet for Speech Denoising". In: *CoRR* abs/1706.07162 (2017). arXiv: 1706.07162. url: http://arxiv.org/abs/1706.07162.

[14] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *CoRR* abs/1812.04948 (2018). arXiv: 1812.04948. url: http://arxiv.org/abs/1812.04948.

[15] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. url: http://arxiv.org/abs/1810.04805.

[16] Chung-Cheng Chiu et al. "State-of-the-art Speech Recognition With Sequence-to-Sequence Models". In: *CoRR* abs/1712.01769 (2017). arXiv: 1712.01769. url: http://arxiv.org/abs/1712.01769.

[17] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv e-prints* (2013). arXiv: 1312.6114 [stat.ML].

[18] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. url: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[19] Aäron van den Oord et al. "WaveNet: A Generative Model for Raw Audio". In: *CoRR* abs/1609.03499 (2016). arXiv: 1609.03499. url: http://arxiv.org/abs/1609.03499.

[20] Aäron van den Oord et al. "Conditional Image Generation with PixelCNN Decoders". In: *CoRR* abs/1606.05328 (2016). arXiv: 1606.05328. url: http://arxiv.org/abs/1606.05328.

[21] Nal Kalchbrenner et al. "Efficient Neural Audio Synthesis". In: *CoRR* abs/1802.08435 (2018). arXiv: 1802.08435. url: http://arxiv.org/abs/1802.08435.

[22] D.T.Braithwaite and W.B.Kleijn. "Speech Enhancement with Variance Constrained Autoencoders". In: *To Appear, InterSpeech* (2019).

[23] Francois G Germain, Qifeng Chen, and Vladlen Koltun. "Speech denoising with deep feature losses". In: (2018). url: https://arxiv.org/abs/1806.10522.

[24] Kaizhi Qian et al. "Speech Enhancement Using Bayesian Wavenet". In: *INTERSPEECH*. 2017.

[25] *PyTorch Github Repository*. 2013. url: https://github.com/pytorch/pytorch.

[26] J-H. Chang, N.S. Kim, and S. K. Mitra. "Voice activity detection based on multiple statistical models". In: *IEEE Transactions on Signal Processing* 54.6 (2006), pp. 1965–1976. issn: 1053-587X. doi: 10.1109/TSP.2006.874403.

[27] Y. Ephraim and D. Malah. "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.6 (1984), pp. 1109–1121. issn: 0096-3518. doi: 10.1109/TASSP.1984.1164453.

[28] L. R. Rabiner and M. R. Sambur. "An algorithm for determining the endpoints of isolated utterances". In: *The Bell System Technical Journal* 54.2 (1975), pp. 297–315. issn: 0005-8580. doi: 10.1002/j.1538-7305.1975.tb02840.x.

[29] S. Gazor and W. Zhang. "Speech probability distribution". In: *IEEE Signal Processing Letters* 10.7 (2003), pp. 204–207. issn: 1070-9908. doi: 10.1109/LSP.2003.813679.

[30] Stefaan Van Gerven and Fei Xie. "A comparative study of speech detection methods". In: *EUROSPEECH*. 1997.

[31] R. C. Hendriks, T. Gerkmann, and J. Jensen. *DFT-Domain Based Single-Microphone Noise Reduction for Speech Enhancement: A Survey of the State of the Art*. Morgan & Claypool, 2013. isbn: 9781627051446. url: https://ieeexplore.ieee.org/document/6813348.

[32] E. W. Healy et al. "An algorithm to improve speech recognition in noise for hearing-impaired listeners". In: *The Journal of the Acoustical Society of America* 34.4 (2013), pp. 3029–3038.

[33] A. Narayanan and D. Wang. "Ideal ratio mask estimation using deep neural networks for robust speech recognition". In: *Proc. ICASSP* (2013), 7092–7096.

[34] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. "On the Surprising Behavior of Distance Metrics in High Dimensional Spaces". In: *Proceedings of the 8th International Conference on Database Theory*. ICDT '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 420–434. isbn: 3-540-41456-8. url: http://dl.acm.org/citation.cfm?id=645504.656414.

[35] Morten Kolbæk. "Single-Microphone Speech Enhancement and Separation Using Deep Learning". In: *CoRR* abs/1808.10620 (2018). arXiv: 1808.10620. url: http://arxiv.org/abs/1808.10620.

[36] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. issn: 0899-7667. doi: 10.1162/neco.1997.9.8.1735. url: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[37] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. url: http://arxiv.org/abs/1412.3555.

[38] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: *CoRR* abs/1803.01271 (2018). arXiv: 1803.01271. url: http://arxiv.org/abs/1803.01271.

[39] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. url: http://arxiv.org/abs/1603.08155.

[40] Shan Qin and Ting Jiang. "Improved Wasserstein conditional generative adversarial network speech enhancement". In: *EURASIP Journal on Wireless Communications and Networking* 2018.1 (2018), p. 181. issn: 1687-1499. doi: 10.1186/s13638-018-1196-0. url: https://doi.org/10.1186/s13638-018-1196-0.

[41] Soroush Mehri et al. "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model". In: *CoRR* abs/1612.07837 (2016). arXiv: 1612.07837. url: http://arxiv.org/abs/1612.07837.

[42] X. Mao et al. "Least Squares Generative Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2813–2821.

[43] Husain Kapadia. "Speech Enhancement using Deep Neural Networks, GN Internship Project Report". In: (2018).

[44] Tim Salimans et al. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 2234–2242. url: http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf.

[45] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 214–223. url: http://proceedings.mlr.press/v70/arjovsky17a.html.

[46] Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5767–5777. url: http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf.

[47] Santiago Pascual. *Segan Issues - Negative Results*. 2017. url: https://github.com/santi-pdp/segan/issues/3#issuecomment-293025199.

[48] Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *International Conference on Learning Representations*. 2018. url: https://openreview.net/forum?id=B1QRgziT-.

[49] Matthias Holschneider et al. "A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform". In: *Wavelets, Time-Frequency Methods and Phase Space* -1 (1989), p. 286. doi: 10.1007/978-3-642-75988-8_28.

[50] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *CoRR* abs/1511.07122 (2015). arXiv: 1511.07122. url: http://arxiv.org/abs/1511.07122.

[51] Recommendations G.711 ITU-T. "Pulse Code Modulation (PCM) of voice frequencies". In: (1988).

[52] Aäron van den Oord et al. "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *CoRR* abs/1711.10433 (2017). arXiv: 1711.10433. url: http://arxiv.org/abs/1711.10433.

[53] Jean-Marc Valin and Jan Skoglund. "LPCNet: Improving Neural Speech Synthesis Through Linear Prediction". In: *arXiv e-prints* (2018). arXiv: 1810.11846 [eess.AS].

[54] Cassia Valentini Botinhao et al. "Speech Enhancement for a Noise-Robust Text-to-Speech Synthesis System using Deep Recurrent Neural Networks". English. In: *Proceedings of Interspeech 2016*. 2016, pp. 352–356. doi: 10.21437/Interspeech.2016-159.

[55] Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent. *DEMAND: a collection of multi-channel recordings of acoustic noise in diverse environments*. Supported by Inria under the Associate Team Program VERSAMUS. 2013. url: https://doi.org/10.5281/zenodo.1227121.

[56] S. Davis and P. Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4 (1980), pp. 357–366. issn: 0096-3518. doi: 10.1109/TASSP.1980.1163420.

[57] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. isbn: 0130226165.

[58] M. Abdel-rahman. "Deep Neural Network Acoustic Models for ASR, PhD Thesis". In: (2014).

[59] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[60] MorganCZY. *Bad cases of waves generated by WaveRNN conditioned on Mel #73*. 2019. url: https://github.com/fatchord/WaveRNN/issues/73.

[61] Daniel T. Braithwaite and W. Bastiaan Kleijn. "Bounded Information Rate Variational Autoencoders". In: *CoRR* abs/1807.07306 (2018). arXiv: 1807.07306. url: http://arxiv.org/abs/1807.07306.

[62] W. B. Kleijn et al. "Wavenet Based Low Rate Speech Coding". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 676–680. doi: 10.1109/ICASSP.2018.8462529.

[63] C. Breithaupt, M. Krawczyk, and R. Martin. "Parameterized MMSE spectral magnitude estimation for the enhancement of noisy speech". In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2008, pp. 4037–4040. doi: 10.1109/ICASSP.2008.4518540.

[64] Elliot J. Crowley et al. "A Closer Look at Structured Pruning for Neural Network Compression". In: *arXiv e-prints*, arXiv:1810.04622 (2018), arXiv:1810.04622. arXiv: 1810.04622 [stat.ML].

[65] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. url: http://arxiv.org/abs/1505.04597.

[66] M. D. Zeiler et al. "On rectified linear units for speech processing". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 3517–3521. doi: 10.1109/ICASSP.2013.6638312.

[67] Pascal Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec (2010), pp. 3371–3408.

[68] Laxmi Pandey, Anurendra Kumar, and Vinay Namboodiri. "Monoaural Audio Source Separation Using Variational Autoencoders." In: *Interspeech*. 2018, pp. 3489–3493.

[69] Simon Leglaive, Laurent Girin, and Radu Horaud. "A variance modeling framework based on variational autoencoders for speech enhancement". In: *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2018, pp. 1–6.

[70] Alexander A Alemi et al. "Fixing a broken ELBO". In: *arXiv preprint arXiv:1711.00464* (2017).

[71] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. issn: 1041-4347. doi: 10.1109/TKDE.2009.191.

[72] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big Data* 3.1 (2016), p. 9. issn: 2196-1115. doi: 10.1186/s40537-016-0043-6. url: https://doi.org/10.1186/s40537-016-0043-6.

[73] ITU-R.BS.1534-1. "Method for the subjective assessment of intermediate sound quality (MUSHRA)". In: (2003).

[74] Ahmed Ali and Steve Renals. "Word error rate estimation for speech recognition: e-WER". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2018, pp. 20–24.

[75] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[76] Judy Hoffman et al. "Asymmetric and Category Invariant Feature Transformations for Domain Adaptation". In: *Int. J. Comput. Vision* 109.1-2 (2014), pp. 28–41. issn: 0920-5691. doi: 10.1007/s11263-014-0719-3. url: http://dx.doi.org/10.1007/s11263-014-0719-3.

[77] Baruch Epstein. Ron Meir and Tomer Michaeli. "Joint auto-encoders: a flexible multi-task learning framework". In: (2017).

[78] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *CoRR* (2014). arXiv: 1411.1792. url: http://arxiv.org/abs/1411.1792.

[79] Chris Baume. *mushraJS*. url: https://github.com/chrisbaume/mushraJS.

[80] C. 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.

[81] Vladimir I Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. 1966, pp. 707–710.

[82] *Geekbench - Samsung Galaxy Note 8*. 2018. url: https://browser.geekbench.com/v4/cpu/14604547.

[83] *Geekbench - Google Pixel 2 XL*. 2018. url: https://browser.geekbench.com/v4/cpu/6473830.

[84] Joshua Alexander et al. "Hearing aid delay and current drain in modern digital devices". In: *Canadian Audiologist* 3.4 (2016).

[85] Jean-Marc Valin and Jan Skoglund. "A Real-Time Wideband Neural Vocoder at 1.6 kb/s Using LPCNet". In: *arXiv e-prints* (2019). arXiv: 1903.12087 [eess.AS].

[86] S. Chopra, R. Hadsell, and Y. LeCun. "Learning a similarity metric discriminatively, with application to face verification". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 539–546 vol. 1. doi: 10.1109/CVPR.2005.202.

[87] Y. Taigman et al. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1701–1708. doi: 10.1109/CVPR.2014.220.

[88] ITU-T.P.800.1. "Methods for objective and subjective assessment of speech and video quality". In: (2016).

[89] Monson H Hayes. *Statistical digital signal processing and modeling*. John Wiley & Sons, 2009.

[90] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Omnipress, 2010, pp. 807–814. isbn: 978-1-60558-907-7. url: http://dl.acm.org/citation.cfm?id=3104322.3104425.

[91] Anders Krogh and John A. Hertz. "A Simple Weight Decay Can Improve Generalization". In: *Proceedings of the 4th International Conference on Neural Information Processing Systems*. NIPS'91. Morgan Kaufmann Publishers Inc., 1991, pp. 950–957. isbn: 1-55860-222-4. url: http://dl.acm.org/citation.cfm?id=2986916.2987033.

[92] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456. url: http://dl.acm.org/citation.cfm?id=3045118.3045167.

[93] Yuxin Wu and Kaiming He. "Group normalization". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 3–19.

[94] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization". In: *arXiv preprint arXiv:1607.08022* (2016).

[95] Jost Tobias Springenberg et al. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[96] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway Networks". In: *CoRR* abs/1505.00387 (2015). arXiv: `1505.00387`. url: `http://arxiv.org/abs/1505.00387`.

[97] K. He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[98] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: `1412.3555`. url: `http://arxiv.org/abs/1412.3555`.

[99] Christopher M Bishop. "Mixture density networks". In: (1994).

A
Background Study

This appendix mentions some fundamentals of general Digital Signal Processing and Deep learning methods. A brief explanation of several modeling and analysis techniques will be provided. The goal of this appendix is to highlight the evolution from processing and predicting signals using heuristic mathematical models to data driven approaches like Machine learning and Deep learning. Some fundamentals about Digital Signal Processing techniques related to speech processing are described in A.1. Basics of Deep learning methods are described in section A.2.

A.1. Digital Signal Processing

Before the era of Machine learning and Deep learning, scientists created mathematical models for synthesizing, analyzing and modifying signals of various types such as time-series financial data, bio-signals, audio, images, etc. Some of the classical approaches for analyzing speech such as the Discrete Fourier Transform A.1.1 and spectral analysis A.1.2 are enlisted here. Thereafter, the Wiener filter approach for noise reduction has been described in segment A.1.3.

A.1.1. Signal Decomposition

When a signal is decomposed into a its components or mapped to a different domain, it can be easier to understand and analyze it from a different perspective. Hence, it is an important analysis and design tool that provides insights into the solution of many problems. Some of the widely used signal decomposition and analysis techniques are the Fourier transform, z-transform and the Wavelet transform. Here, the Fourier transform will be discussed in more detail since it has been used for analyzing the speech signals as described in chapter 3.

The Fourier Transform is known to decompose a time signal into its constituent frequencies also known as a spectrum. Thus, the representation of a signal in the frequency domain conveys the contribution of different frequencies that would form the same signal in the time domain. It is an invertible transform and the time domain signal can be recreated from its frequency domain representation. Since, computation happens digitally, the analysis of discrete signals is performed with a Discrete Fourier Transform (DFT) and this representation is amenable for finite length sequences. The transformation is very efficient and has low complexity and yields almost uncorrelated spectral coefficients. Analysis of speech signals is performed on short-time frames. The reason is based on the principle of stationarity which assumes that the audio signal does not change much (statistically). Thus, an ideal frame length is $20-40$ms. If the frame is much shorter we don't have enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame. Thus, the DFT applied to a single short-time frame of speech is expressed as:

$$S(k)=\sum_{n=0}^{N-1} w(n)s(n)\exp\left(\frac{-j2\pi kn}{N}\right), \tag{A.1}$$

where, $S(k)$ represents the spectrum of a short-time frame of the signal $s(n)$ with a frame length of N samples. n stands for the time index in the signal and k is the frequency bin in the spectrum. w is a window function and their purpose is to reduce the spectral leaks. This transform is carried out for every short-time frame in the speech utterance. After, analyzing the signal in the frequency domain, different types of filters can be applied so as to process the signal. After doing so, the processed spectrum can be synthesized back to the waveform with the help of an inverse of this transform that is expressed as:

$$s(k) = \frac{1}{N}\sum_{k=0}^{N-1} w(k)S(k)\exp\left(\frac{-j2\pi kn}{N}\right). \tag{A.2}$$

A.1.2. Spectral Analysis

Spectral analysis involves estimating the power spectrum of signals which is called a periodogram. This estimate should ideally be as close as possible to the true power spectrum so as to accurately design filters such as the Wiener filters described in segment A.1.3. Since, analysis of speech involves restricting the length of the time frame due to the stationarity assumption, it is challenging to estimate the true power spectrum due to limited data. Moreover, the data is often corrupted by noise which makes it even more difficult. Although computing a periodogram is easy, it is limited in its ability to produce an accurate estimate of the power spectrum particularly for short data records. By definition, the periodogram is proportional to the squared magnitude of the DFT of a signal $s(n)$ which is expressed as:

$$\hat{P}_S(k) = \frac{1}{N}|S(k)|^2, \tag{A.3}$$

where, $P_S(k)$ represents the periodogram or power spectral estimate of the signal and $S(k)$ stands for the spectrum of the signal as shown in equation A.1. Since, while calculating the DFT, the signal was multiplied with a window function, the power spectrum P_S can also be termed as a modified periodogram. The periodogram is said to be a biased estimate of the power spectrum but in order to reduce this bias, Welch [89] proposed to average these modified periodograms by overlapping the frames. As a result, the periodogram estimate by Welch's method is given by:

$$\hat{P}_W(k) = \frac{1}{KL}\sum_{i=0}^{K-1}\left|\sum_{n=0}^{L-1} w(n)s(n+iD)\exp\left(\frac{-j2\pi kn}{N}\right)\right|^2, \tag{A.4}$$

where, $P_W(k)$ represents the periodogram or power spectral estimate of the signal obtained by Welch's method. The data length N can be expressed in terms of K sequences of short-time frames of length L with an overlap of D samples as $N = L + D(K-1)$. By allowing the sequences to overlap it is possible to increase the number of sequences and reduce its length so as to decrease the variance of the periodogram even further thus giving smooth estimates. This idea of overlapping frames to calculate the power spectrum has been borrowed by spectrograms as dexcribed in chapter 3 section 3.3. The difference between a spectrogram and a periodogram is that instead of averaging the short-time frames, these frames are stacked along the temporal axis to give a time-frequency visualization of speech.

A.1.3. Wiener Filtering

The Wiener filter is extensively used in speech processing for the problem of noise reduction. The noise reduction problem is defined as follows, given the observation of a noise corrupted version of a speech signal $x(n) = s(n) + v(n)$, the goal is to estimate the desired speech signal $s(n)$. The Wiener filter has been designed to produce the minimum mean square error estimate of the speech signal $s(n)$. Thus, the Wiener filter is said to be the most optimal in the least squares sense. After minimizing the mean square error between the desired speech signal $s(n)$ and the filtered output of the observed speech signal $x(n)$ in the frequency domain [1], for the most optimal filter coefficients, the Wiener filter can be

[1]The Wiener filter coefficients can also be calculated in the time domain using cross-correlations and auto-correlations between the desired and observed noisy speech signals. however, it is preferred to express it in the frequency domain due to the ease in designging and analyzing filters.

expressed as:

$$H(k) = \frac{P_S(k)}{P_X(k)} = \frac{P_S(k)}{P_S(k) + P_V(k)} = 1 - \frac{P_V(k)}{P_X(k)}, \tag{A.5}$$

where, $H(k)$ represents the Wiener filter coefficients and $P_S(k)$ and $P_X(k)$ stand for the power spectral densities of the desired speech signal $s(n)$ and the noise corrupted speech signal $x(n)$ at the frequency bin k. $P_V(k)$ is the power spectral density of the noise signal. This expression has been derived after considering the following assumptions that only additive noise is added to speech, speech and noise DFT coefficients are uncorrelated and they are assumed to be wide-sense stationary. The Wiener filter can also be expressed in terms of the signal-to-noise ratio (SNR) as:

$$H(k) = \frac{P_S(k)/P_V(k)}{P_S(k)/P_V(k) + 1} = \frac{\zeta(k)}{\zeta(k) + 1}, \tag{A.6}$$

where, $\zeta(k)$ is the Signal-to-noise ratio at the k^{th} frequency bin. Thus, it can also be inferred that the Wiener filter suppresses spectral regions with low SNR while it does not modify spectral regions with high SNR.

After defining the most optimal filter coefficients for noise reduction, the main problem is to correctly estimate the power spectral densities of speech and noise. These variables are unknown because the signal under observation is $x(n)$. Hence, the power spectrum estimation techniques described in segment A.1.2 comes into role. Using, Welch's method for power spectral density estimation, the power spectrum of the noisy signal can be estimated. The power spectrum of noise can be estimated assuming that the first few milliseconds of an utterance only contains noise. With the following set of assumptions, the Wiener filter coefficients can be calculated. But, these assumptions are very ideal and many techniques have been developed to effectively estimate the signal-to-noise ratios, speech and noise power spectral densities, etc. that are discussed in chapter 2 section 2.1.

A.2. Deep learning

The introduction of Deep learning not only enabled scientists and researchers to tackle signal processing problems from a different perspective but also opened possibilities for implementing several ideas which could not have been feasible with traditional approaches. In this section, some deep learning fundamentals about non-linear functions A.2.1, objective functions A.2.2 and regularization functions A.2.3 are explained. Moreover, a theoretical description about neural network architectures such as Fully connected networks A.2.4, Convolutional neural networks A.2.5, Recurrent Neural networks A.2.6 that are used during implementation is provided. Thereafter, the concept of generative modeling with neural networks is described in segment A.2.7.

A.2.1. Non-Linear Units

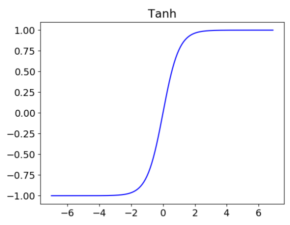
These functions are responsible for making neural networks non-linear. After, augmenting the input data with the trainable parameters of the network; blocks called 'Activation functions' are applied for inducing an element of non-linearity. If non-linear functions are not used then the depth of a network i.e. dividing it into several layers would not play an important role as it could all be represented by one big linear linear operation. Because of applying non-linear units after every layer, it is possible to map the input space to a complex representation which makes neural networks so fascinating. Some widely used non-linear activation functions have been enlisted in this segment.

Tanh & Sigmoid

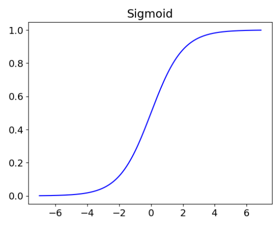
Tanh stands for the hyperbolic tangent function and calculated as shown in equation A.7 and illustrated in figure A.1a. The sigmoid function also called the logistic function is calculated as described in equation A.8 and displayed in figure A.1b. Tanh squashes the input values to the range $[-1, 1]$ thus preserving the sign of the input. Thus, it is used very often for tasks involving regression. Sigmoid squashes its input to the range $[0, 1]$. It can be interpreted as a gating function to control the amount of information flow.

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad x \in \mathbb{R} \tag{A.7}$$

$$f(x) = \frac{1}{1 + e^{-x}}, \quad x \in \mathbb{R} \tag{A.8}$$



(a) Tanh function



(b) Sigmoid function

Figure A.1: Tanh & Sigmoid non-linearities

Both these functions encourage high non-linearity at every input value which favours the main purpose of a neural network. Secondly, even though it gets close to flat at the extremities, it isn't completely flat anywhere thus reflecting changes in the input. However, the derivative of the function gets infinitesimally small as the input values approach the extremes and so it is difficult to update weights with the optimizers. This problem is named as the 'vanishing' gradient problem. It is also computationally expensive because it involves calculating the exponential. Fortunately, the gradients of both Tanh and Sigmoid depend on their values and hence, it is saved in memory during the forward pass so that it can be recomputed easily instead of calculating the exponential again.

ReLU

ReLU is the abbreviation for a Rectified Linear Unit and was introduced by G. Hinton *et al.* [90]. It's function is analogous to a half wave rectifier, so it returns 0 if it receives a non-positive input and returns the same for any positive value. It is mainly used for regression purpose and hence mainly in the hidden layers of the network. It can be represented by the equation A.9 and it is graphically displayed in figure A.2a. If, this function is differentiated it has two distinct slope values i.e. 1 for positive inputs and 0 for non-positive inputs. It can be seen that it is more like a piecewise linear function instead of a strict non-linearity but it still performs significantly well. But, being so limited in the extent of non-linearity it offers, how is it able to achieve it? The answer is two-fold. Firstly, most models include a bias parameter (along with the weight) for each node which controls the movement of the input to the function about 0. And, secondly, every layer has multiple nodes with different bias values for each input. So, the resulting function changes slopes in many places thus making it non-linear in the feature space encouraging better interaction amongst features.

$$f(x) = \max(0, x), \quad x \in \mathbb{R} \tag{A.9}$$

There are also different types of ReLUs such as the Leaky ReLU (LReLU) and the Parametric Relu (PReLU) which are represented by equations A.10 and A.11 and graphically displayed in A.2b and A.2c respectively. These functions have a theoretical advantage that by being influenced by the input value, it may consider more complete representations of the information contained in the input. However, depending on the type of the network being used, it is likely to suffer from the exploding gradients problem. They are known to improve the performance of neural network based acoustic systems [66].

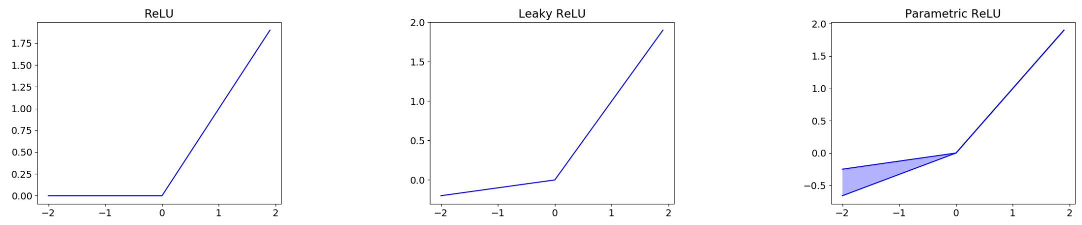
$$f(x) = \max(0, x) + 0.1 * \min(0, x), \quad x \in \mathbb{R} \tag{A.10}$$

$$f(x) = \max(0, x) + a * \min(0, x), \quad x \in \mathbb{R} \tag{A.11}$$

Besides being computationally simple, one main advantage of all types of ReLUs is that they do not suffer from the 'vanishing' gradient problem. When training on a reasonable batch size, there will be some points giving positive values to the nodes and some negative values and thus, the average derivative is rarely close to 0 which allows the optimizer to keep progressing.

Softmax

A softmax function can be viewed as a normalized exponential function since it normalizes the input vector into a probability distribution. After applying softmax, each component in the input vector are mapped to the interval $(0, 1)$ and the components add up to 1 (difference between sigmoid and



(a) ReLU function (b) Leaky ReLU function (c) Parametric ReLU function

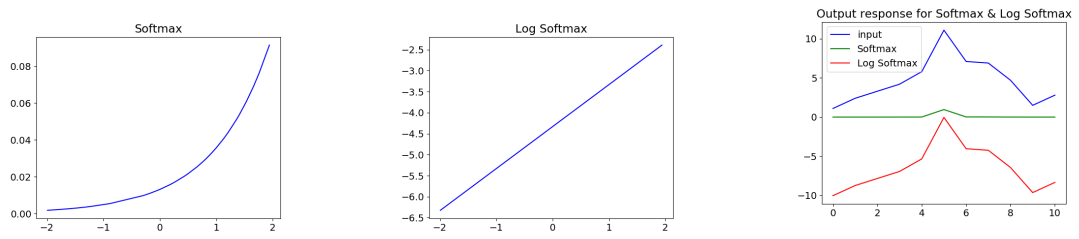
Figure A.2: Different types of ReLU non-linearities

softmax). The larger input components correspond to larger probabilities. Softmax is often used in neural networks, to map the non-normalized output of a network to a probability distribution over predicted output classes. It is based on the equation A.12 and is illustrated in figure A.3a

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{K} e^{x_j}}, \quad \forall i = 0, 1, ..., K \quad \& \quad x \in \mathbb{R}^K \tag{A.12}$$

A log-softmax function can also be used instead of a softmax. Ideally, both serve the same purpose but log softmax has some advantages over a softmax function. Besides practical reasons like improved numerical performance and gradient optimization that improve training performance, log softmax function is known for heavily penalizing the model when it fails to predict the correct sample value. Moreover, using a log softmax over a softmax function means yielding log probabilities over probabilities thus leading to nice information theoretic interpretations. Equation A.13 and figure A.3b represent the log-softmax operation.

$$f(x_i) = x_i - \log \sum_{j=0}^{K} e^{x_j}, \quad \forall i = 0, 1, ..., K \quad \& \quad x \in \mathbb{R}^K \tag{A.13}$$



(a) Softmax function (b) Log Softmax function (c) Output response of Softmax and Log Softmax functions

Figure A.3: Softmax & Log Softmax non-linearities

Figure A.3c shows the output of both the softmax function and the log-softmax function to a random input. The softmax squashes the input to a range between $(0, 1)$ whereas the log softmax being linear just shifts the same input by a bias as shown in equation A.13. Thus, if the network has a large depth, it would be beneficial to use the log-softmax to prevent the vanishing gradient problem.

A.2.2. Objective functions

An objective function, also named cost function, is a method of determining how well does a neural network model the data during training. In other words, it is a measure of the deviation of the model prediction from the actual data. Thus, the cost function learns to reduce the error or loss in prediction with the help of an optimization process by tuning the model parameters. A list of some objective functions used during the course of this project is provided as follows:

L_1 & L_2 Norm

These are the most commonly used loss functions for regression based problems. The L_1 norm stands for the mean absolute error and L_2 loss is the mean square error between the predicted signal and the target signal. These loss functions are expressed as follows:

$$\begin{aligned} L_1(\hat{x}, x) &= \frac{1}{N} \sum_{i=0}^{N} |\hat{x}_i - x_i|, \\ L_2(\hat{x}, x) &= \frac{1}{N} \sum_{i=0}^{N} (\hat{x}_i - x_i)^2, \end{aligned} \tag{A.14}$$

where, $\hat{x}$ and x represent the predicated values and target variables respectively. N is the dimension of the vectors. The choice between these two regression metrics depends on the application. L_2 loss is more sensitive to the presence of outliers in the dataset and tends to fit the model closer to these outliers. Due to the use of a squared distance, this objective incurs a large loss if outliers are not mapped correctly. In contrast, an L_1 objective is more robust to the existence of outliers in the dataset since it relies on the absolute distance between two points. Hence, the L_1 can effectively map the data despite the presence of outliers.

Cross Entropy or Negative Log Likelihood Loss

Cross entropy loss is most commonly used for binary or multi-class classification problems. For classification problems, the last layer of the network is mapped to a probability distribution which is used for the cross entropy loss minimization. This objective is expressed as:

$$L(y, p) = -\sum_{i=0}^{C} y_i \log(p_i), \tag{A.15}$$

where, L represents the cross entropy loss which is a function of the class label y and the probability distribution at the output of the network denoted by p. C denotes the number of classes. The loss in equation A.15 is expressed in a unit called 'nats'. If the $\log$ is expressed to the base 2, then the loss is expressed in 'bits'. This loss is also known as the negative log likelihood loss.

A.2.3. Regularization

One of the problems that deep learning suffers from is 'overfitting' to the training dataset. This problem occurs more frequently in complicated networks with large number of learnable parameters. Regularization can be seen as a way of imposing a constraint on the model such that it does not overfit to training data and generalizes well to unseen inputs. There are multiple ways of regularizing a neural network and some of them have been highlighted as follows:

Penalty on Cost functions

Many regularization approaches are based on limiting the capacity of models by adding a penalty on the norm of the parameters or a constraint to the model's objective function described in segment A.2.2. Thus, the modified objective function is expressed as:

$$L(\theta, \alpha; X, y) = J(\theta; X, y) + \alpha(\Omega(\theta) - k), \tag{A.16}$$

where, L represents the regularized loss that is a function of the learnable parameters θ, hyper-parameter α, input features X and output labels y. The term $J(\theta, X, y)$ is the objective function as described in segment A.2.2 and $\Omega(\theta)$ is the regularization function that acts as a penalty on the objective function and k is a desired constant. Commonly used regularization functions are the L_1 or L_2 norm of the network parameters θ which constrain these weights to lie in the region limited by these functions. The L_1 norm on the weights encourages sparsity in the learned weights whereas the L_2 norm shrinks the weights towards 0, hence, also termed as weight decay [91]. Penalties can also be added to constrain the intermediate feature space at the hidden layers of the networks. When the training algorithm minimizes the regularized loss function, it decreases both the original objective on the training data as well as the constrained measure on the network parameters or intermediate features.

Data Augmentation

The best way to enforce generalization in machine learning models is by training it with more data [75]. In practice, the amount of data available is limited and a way around this it to artificially create fake data and add it to the training set. A general practice is to either scale, translate or rotate in case of images or inject random noise with a carefully tuned magnitude in case of speech. Adding noise to hidden layers in the network leads to dataset augmentation at multiple levels of abstraction. Although, such transformations must be applied carefully ensuring not to change the correct class.

Normalization

Applying normalization layers after every hidden layer can also lead to regularization effects. For instance, the most commonly used batch normalization [92] technique normalizes the input features with the mean and standard deviation computed per dimension over the mini-batches. Batch normalization is said to be analogous to noise injection in the sense that it multiplies each hidden unit by a random value at each step. In this case, the random value is the inverse of the standard deviation because different examples are randomly chosen for inclusion in the mini-batch at each step hence, the standard deviation fluctuates randomly. Batch normalization also subtracts a random value, i.e., the mean of the mini-batch from each hidden unit. Both of these sources of noise at every layer causes the network to learn to be robust to a lot of variation in its input. A similar interpretation can be applied for group normalization [93] where normalization is applied over a group of channels in a convolutional layer as well as for instance normalization [94] where normalization is applied to separately for each object in a mini-batch.

A.2.4. Fully-connected Neural Networks

Fully-connected neural networks also called Multilayer Perceptrons (MLPs) belong to the class of deep feed forward networks. These are the most quintessential deep learning models. Before getting into the depth of fully-connected networks, the fundamentals of feed forward networks should be explained. The goal of a feed forward networks are to approximate a function f that maps an input x to a category y. A feedforward network defines a mapping $y = f(x;\theta)$ and learns the value of the parameters θ that result in the best function approximation. These models are called feed forward because information flows from the input x through the intermediate computations used to define the function f to the output y. There are no feedback connections from the output to the input. The class of feed forward neural networks also includes convolutional neural networks that are described in detail segment A.2.5. When feed forward networks include feedback connections, they are called recurrent neural networks as presented in segment A.2.6.

Feed forward neural networks are typically composed of many different functions that are connected in the form of a chain. For instance, $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. The subscript on the top indicates the layer number and the overall length of the chain determines the depth of the network. During a neural network training, $f(x)$ is driven so as the match the most optimal function denoted by $f^*(x)$. Each example x in the training set is accompanied with a label $y \approx f^*(x)$ in case of a supervised learning problem. Training examples specify the role of the output layer at each point x ,i.e., to produce a value that is as close as possible to y. The behaviour of intermediate layers, also called hidden layers, is learned by training the network so as to best implement an approximation of f^*. The dimensionality of each hidden layer determines the width of the model. Each unit in the layer is termed as a neuron since it receives input from many other such units and computes its own activation value using the non-linear transforms denoted by ϕ as described in segment A.2.1.

In fully-connected networks, each neuron in one layer is linked to every neuron in the subsequent layer. This link from one neuron to another is determined by the weight of the connection which is a learnable network parameter. Thus, the connection strengths from a vector of neurons in one hidden layer to another hidden layer is represented by a weight matrix W^l, where l stands for the layer number. This weight matrix is dot multiplied with the input feature vector x. As described earlier that every neuron consists of an activation function ϕ and the linear combination of the weights and features evaluates the activations of all neurons in that layer. Thus, a fully-connected network consists of a chain of such operations where repeated matrix-vector multiplications are interwoven with activation functions. These operations are expressed as:

$$f^l(x) = \phi(W^l_{m\times n}.x_n + b^l_m), \tag{A.17}$$

where, $f^l(x)$ is the m-dimensional output vector of a layer with an n-dimensional input feature vector denoted by x. $W^l_{m\times n}$ represents the weights of the layer l that have a dimension of $m \times n$. An m-dimensional bias vector b^l is also added to the matrix-vector product of the weights and the features. The weights W and bias b form the learnable parameters of the layer and are a part of the network parameters denoted by θ. The activation function is represented by ϕ.

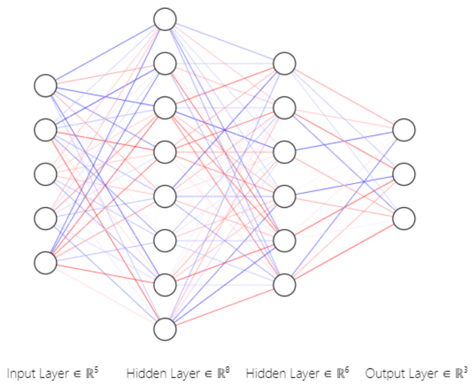


Figure A.4: A 4 layer fully connected network consisting of an input layer with 5 nodes, two hidden layers with 8 nodes and 6 nodes respectively and an output layer with 3 nodes.

Figure A.4 shows a fully-connected neural network that consists of 5 nodes in the input layer where a feature vector of length 5 is dot multiplied with the weight matrix represented by the links in the figure. Red links represent positive weight values whereas the blue ones are the negative values and their opacity suggests the strength of the weight values. Thus, the input is propagated through the two hidden layers where non-linear activation functions are applied. The final layer, also called the output layer, consists of 3 nodes for the output values which represent the predicted label y. The output layer is supplied to an objective function described in segment A.2.2. Thereafter, gradients are calculated with respect to the learnable parameters and they are backpropagated through every layer in the network so as to update the parameters till the objective function has converged to a minimum.

A.2.5. Convolutional Neural Networks

Convolutional Neural Networks, also known as, ConvNets or CNNs, belong to the class of feed-forward networks as described in segment A.2.4. Convolutional networks also consist of neurons that are controlled by learnable weights and biases. Each neuron receives inputs either from raw data or from previous layers, performs a dot product with the network weights followed by a non-linear unit. The whole network is expressed as a single differentiable score derived from the raw pixels of an image. They also have loss functions that compare the output of the last layer of the network with a set of pre-defined labels in the training set. Moreover, the methods mentioned above for loss functions (A.2.2) and regularization (A.2.3)) all apply to convolutional networks as well. So what makes them different from fully-connnected networks? Firstly, ConvNet architectures explicitly assume that the inputs are images which allows them to encode certain properties into the architecture. This makes the forward function more efficient with a vast reduction in the number of network parameters. Regular neural networks do not scale very well to images and hence the advent of convolutional networks has been a boon. It is also impractical to connect neurons to all the neurons in an image volume when only neurons surrounding a particular neuron are most relevant.

Convolutional Neural Networks take advantage of the fact that the input consists of images and their architecture is constrained accordingly. Unlike, fully connected networks, every layer of a convolutional network have neurons arranged in 3 dimensions i.e. width, height and depth. Here, height and width define the resolution of the image and depth stands for the number of channels. ConvNets mainly constitute of convolutional layers, pooling layers and optionally fully connected layers as well. In this section, the convolutional layer, which form the core building block of ConvNets, is described in more detail. The parameters of this layer consists of a set of learnable filters also called kernels. Every kernel is small spatially but extends through the full dimensions of the input volume. During the forward pass, each filter is slid or rather, more precisely, convolved across the width and height of the input and computes a dot product between the entities of the filter and the small local regions in the input. Eventually, a 2D activation map is created which captures the response of that filter at every spatial position in the input. Intuitively, the network learns filters that activate when they see some visual feature such as an edge of some orientation or a blob of some colour or some distinct patterns. An entire set of such filters are stacked along different channels at every layer and each of them after convolution produce a separate 2D activation map. This operation can be mathematically expressed as:

$$O^l(C_{out}) = \sum_{k=0}^{C_{in}-1} W^l(C_{out}, k) \circledast I(k) + b^l(C_{out}), \tag{A.18}$$

where, O^l stands for the output activation map at the l^{th} layer with C_{out} channels. The input features, I, at the k^{th} input channel are convoluted with the network weights, W^l, and summed across the number of input channels, C_{in}. The bias parameters b^l are also added at every output channel.

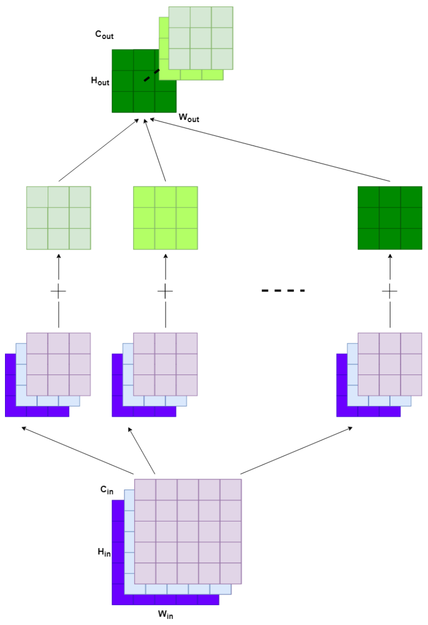
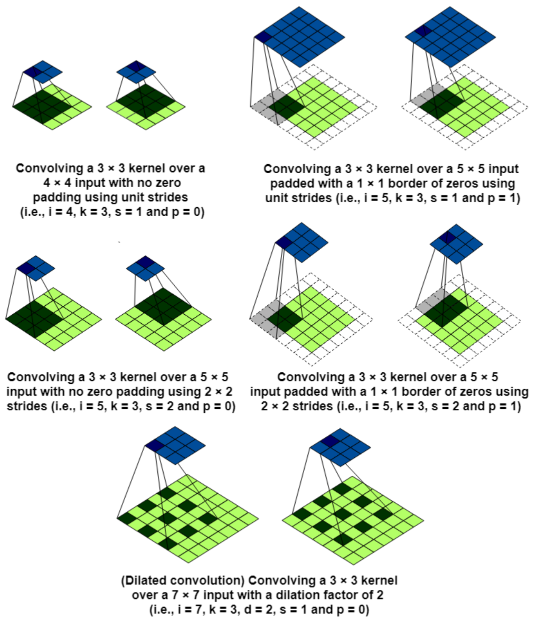


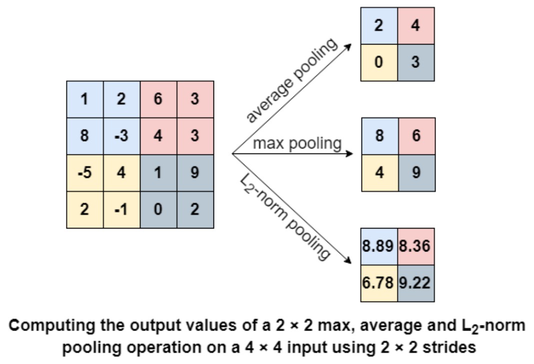
Figure A.5: An illustration of equation A.18 showing the convolution operation in a layer.

The structure of the network and the convolution operation are controlled by a set of hyper-parameters. The receptive field (kernel size), denoted by K, controls the spatial extent of the connectivity of a neuron along the entire depth of the input volume of size expressed as (C_{in}, H_{in}, W_{in}). Hyper-parameters such as number of channels, padding and stride define the output volume of the neurons. The output volume denoted by $(C_{out}, H_{out}, W_{out})$ corresponds to the number of filters or output channels and its spatial extent. Sometimes, it is convenient to pad the input volume size with zeros around the border. The size of zero-padding, denoted by P, allows to control the size of the output volume. It is most commonly used to preserve the spatial size of the input so that the width and height of the input and output are the same. Specifying the stride S allows to control the amount which with a filter slides across the input spatial size. A stride larger than 1 produces a smaller output volume spatially. So far the filters have been assumed to be contiguous in space, however, the introduction of the dilation D hyper-parameter allows the filter to have spaces between each cell. The use of dilation can increase the effective receptive field size. Figure A.6a depicts the effect of these hyper-parameters visually. The size of the output volume can also be calculated based on the formula given in equation A.19.

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times P[0] - D[0] \times (K[0] - 1) - 1}{S[0]} + 1 \right\rfloor,$$
$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times P[1] - D[1] \times (K[1] - 1) - 1}{S[1]} + 1 \right\rfloor. \tag{A.19}$$


Convolving a 3 × 3 kernel over a 4 × 4 input with no zero padding using unit strides (i.e., i = 4, k = 3, s = 1 and p = 0)
Convolving a 3 × 3 kernel over a 5 × 5 input padded with a 1 × 1 border of zeros using unit strides (i.e., i = 5, k = 3, s = 1 and p = 1)
Convolving a 3 × 3 kernel over a 5 × 5 input with no zero padding using 2 × 2 strides (i.e., i = 5, k = 3, s = 2 and p = 0)
Convolving a 3 × 3 kernel over a 5 × 5 input padded with a 1 × 1 border of zeros using 2 × 2 strides (i.e., i = 5, k = 3, s = 2 and p = 1)
(Dilated convolution) Convolving a 3 × 3 kernel over a 7 × 7 input with a dilation factor of 2 (i.e., i = 7, k = 3, d = 2, s = 1 and p = 0)

(a) Effect in the shape of the output when the hyper-parameters controlling the convolution operation are changed.


1 2 6 3
8 -3 4 3
-5 4 1 9
2 -1 0 2
average pooling
2 4
0 3
max pooling
8 6
4 9
L2-norm pooling
8.89 8.36
6.78 9.22
Computing the output values of a 2 × 2 max, average and L2-norm pooling operation on a 4 × 4 input using 2 × 2 strides

(b) Different types of Pooling operations

The convolution operation discussed earlier either preserves or reduces the spatial resolution, but, what if, there is a need for the transformation to be applied in the reverse sense while maintaining the connectivity patterns compatible with the said convolution. For instance, such a transformation could be useful in decoding layers of an auto-encoder or to project feature maps to a higher dimensional space. Thus, the need for a transposed convolutions or fractionally strided convolutions, arises [2]. The hyper-parameters in this case also remain the same as defined for the convolution operation. The dimensions of the output volume can be expressed as:

$$H_{out} = (H_{in} - 1) \times S[0] - 2 \times P[0] + D[0] \times (K[0] - 1) + 1,$$
$$W_{out} = (W_{in} - 1) \times S[1] - 2 \times P[1] + D[1] \times (K[1] - 1) + 1. \tag{A.20}$$

Sometimes pooling layers are also inserted in between two successive convolutional layers. A pooling layer reduces the spatial size of the input feature representation to reduce the amount of parameters and computation in the network. A pooling layer operates on every depth slice or channel of the input volume and resizes it spatially preserving the number of channels in the output volume. Pooling also works by sliding a window across the spatial content and applies a pooling function. There are different types of pooling operations such as max pooling, average pooling or L_2 norm pooling. Figure A.6b displays different types of pooling operations. Research has shown that these layers do not exhibit any added advantage and in some cases also ruins the performance of the network [95]. On the other hand, using strided convolutions instead of pooling turns out to be more beneficial.

Other variants of ConvNets have also been introduced such as highway networks and residual neural networks. These networks introduce skip connections that traverse from the input of the layer to its

[2]Note that, in a decoder setting, the transposed convolution does not guarantee to recover the input as it is not defined as the inverse of a convolution, but rather just returns a feature map that has same spatial dimension as the input

output, thus providing an additional path for the data and gradients to flow. This is in stark contrast to the traditional strictly sequential pipeline. In highway networks, these skip connections are controlled by learnable gating functions [96]. Whereas, in residual networks the skip connection is equivalent to applying an identity transform from the input to the output [97]. A study claims that these short-cut connections are advantageous because they solve the vanishing gradients problem in deep networks [97]. Thus, skip connections can facilitate in designing deeper networks with large number of layers.

A.2.6. Recurrent Neural Networks

In contrast to the Feed-Forward Networks, Recurrent Networks receive not just the input at the current time step but also what they have perceived previously in time. It can be claimed that Recurrent Networks gain inspiration from how humans think and base their understandings from previous context as well. Traditional Neural networks fail to do so and Recurrent networks address this issue. Hence, they are neural networks with loops within them allowing information to persist and passed from one time step to the next. This loop in Recurrent Networks can be unfolded into a chain of multiple copies of the same network up to a pre-defined sequence length such that each network passes information from one to the successive network as shown in figure A.7. Unfolding leads to 'parameter sharing' which is better than using different parameters at different time-steps and helps in generalizing better. Such a structure makes recurrent neural networks appropriate for modeling sequential data such as language, speech and time-series data. Recurrent Networks can be formulated as:

$$h_t = \tanh(W * [h_{t-1}, x_t] + b), \tag{A.21}$$

where h_t is the current hidden state which is a non-linear function of the hidden state at the previous time-step h_{t-1}, the current context x_t and network parameters i.e. weights and biases represented as W and b respectively. The network learns to use a kind of a summary of the past sequence of inputs as expressed in the hidden state vector.

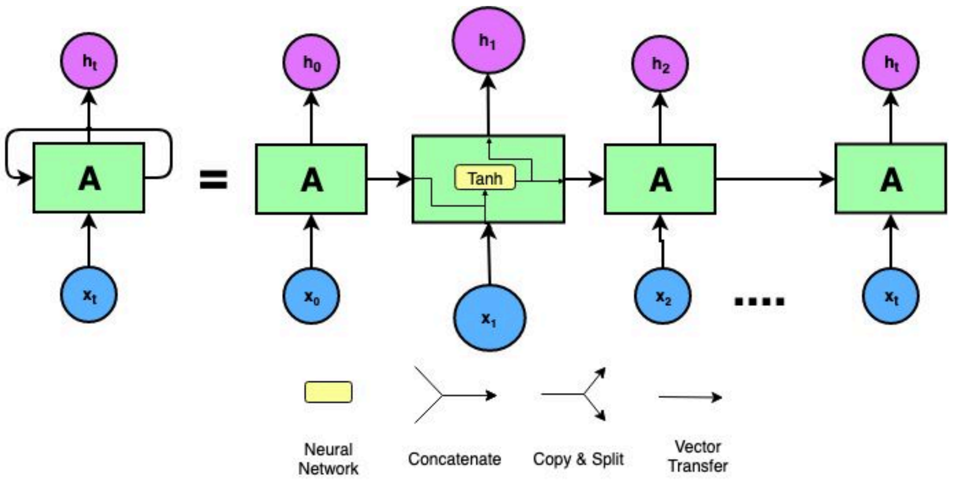


Figure A.7: The unfolding of an RNN layer and internal operations within an RNN block

For updating the network weights, the gradients are backpropagated through time which is obtained by applying the chain rule of derivatives on an unfolded representation of the network. However, as the sequence length i.e. the number of unfoldings increases, the gradients become smaller and smaller in magnitude and are unable to sufficiently update the network parameters. This problem is known as the vanishing gradient problem. Thus, these networks are good at capturing short-term dependencies as compared to long-term dependencies in sequential data. In theory, Recurrent Networks are absolutely capable of capturing long term dependencies, but due to the vanishing gradient problem, their performance is limited. This gave rise to Long Short Term Memory (LSTM) networks and Gated Recurrent Units (GRUs).

LSTM's [36] are special kinds of recurrent networks that are designed in a way that makes them capable of learning long-term dependencies. They have been widely used on a large variety of sequence

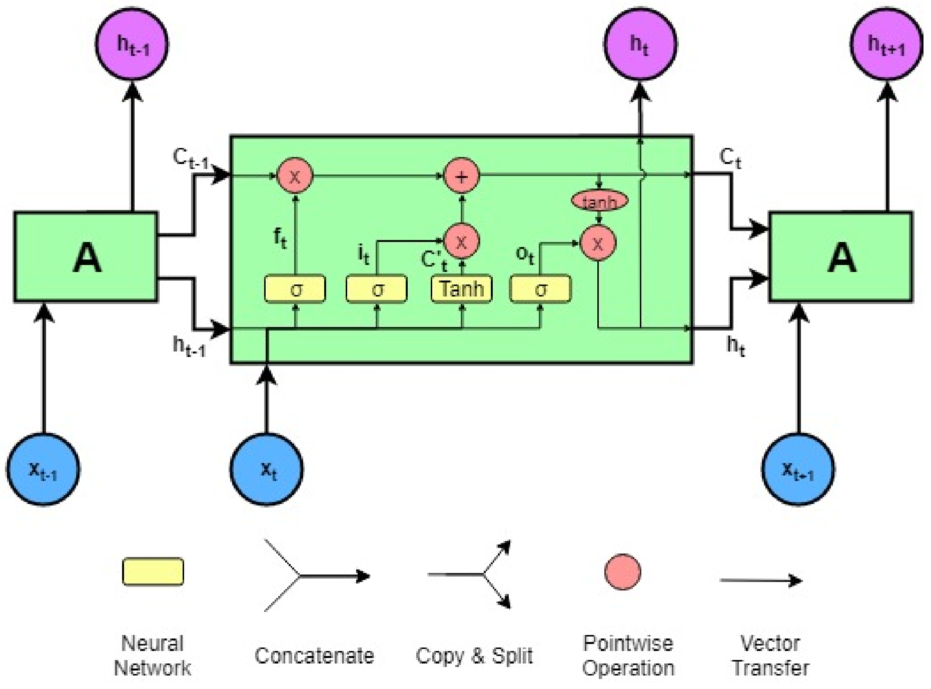


Figure A.8: The unfolding of an LSTM layer and internal operations within an LSTM block

modeling problems and have led to an improvement in performance as compared to the traditional recurrent networks. LSTM's are also expressed in a chain of repeating modules but their internal structure is different as illustrated in figure A.8. An LSTM network can be formulated as:

$$\begin{aligned} f_t &= \sigma(W_f.[h_{t-1}, x_t] + b_f), \\ i_t &= \sigma(W_i.[h_{t-1}, x_t] + b_i), \\ \tilde{C}_t &= \tanh(W_C.[h_{t-1}, x_t] + b_C), \\ C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t, \\ o_t &= \sigma(W_o.[h_{t-1}, x_t] + b_o), \\ h_t &= o_t \circ \tanh(C_t), \end{aligned} \tag{A.22}$$

where, in addition to the hidden state h, a new variable known as the cell state C is also introduced. This cell state makes LSTMs capable of regulating the flow of information with the help of gating units σ. The sigmoid layers output values within the range $[0, 1]$ which are applied point-wise to determine the amount of input x_t and/or previous content h_{t-1} to be let through. Thus, the term f_t represents the 'forget gate' which decides the amount of information to be neglected while updating the cell state. The gate i_t represents the 'input gate' that decides the amount of new information to be stored within the cell state. This new information is a vector of new candidate values $\tilde{C}_t$ that are obtained by applying a $tanh$ operation on the current input x_t and the previous hidden state h_{t-1}. After deciding what content has to be stored and neglected, the current cell state C_t is updated by discarding some portion of the previous cell state C_{t-1} controlled by f_t and adding new information $\tilde{C}_t$ controlled by i_t. Finally, the output i.e. the updated hidden state h_t is controlled by the 'output gate' o_t that permits only the decided information to be supplied to the layer at the next time-step. This cycle continues till the amount of unfolding of the LSTM netowrk. The network parameters i.e. the weights and biases at different gates are represented by symbols W and b with their respective subscripts indicating the gate operations. While backpropagating through time, the updates of these network parameters are also decided by the respective gating functions.

Another variant of gated recurrent units are GRU's [37]. There are two major differences between LSTM and GRU networks; first is that the hidden state and the cell state are combined. Thus, the forget

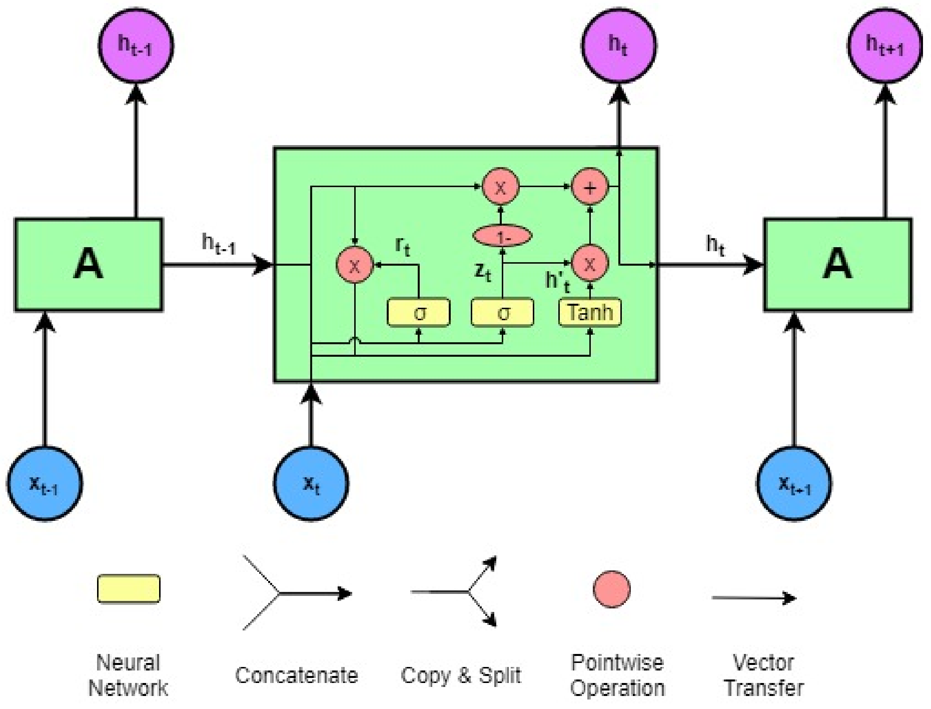


Figure A.9: The unfolding of a GRU layer and internal operations within a GRU block

and input gate layers in LSTM that were responsible to update the cell state now control the hidden state directly using the update gate and reset gate layers. These changes enable GRUs to control the information flow from the previous activation when computing the new, candidate activation. Second, is the absence of the output gate which allows a particular unit to expose its full content to the subsequent units within the layer without any control. The chain structure representation and internal operations within a GRU network are illustrated in figure A.9. A GRU network can be expressed as:

$$\begin{aligned} z_t &= \sigma(W_z.[h_{t-1}, x_t] + b_z), \\ r_t &= \sigma(W_r.[h_{t-1}, x_t] + b_r), \\ \widetilde{h}_t &= \tanh(W.[r_t \circ h_{t-1}, x_t] + b), \\ h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \widetilde{h}_t, \end{aligned} \tag{A.23}$$

where, h is the hidden state that represents the summary of the outputs from the current and past time-steps. The information flow within the hidden state in GRUs is regulated with the help of gating units σ. The sigmoid layers output values within the range $[0, 1]$ which are applied point-wise to determine the amount of input x_t and/or previous content h_{t-1} to be let through. The 'reset gate', r_t, controls the amount of past information to be neglected. The hidden state from previous time-steps h_{t-1} is altered by the reset gate such that it only keeps the relevant information. The current memory content $\widetilde{h}_t$ is calculated by applying a $tanh$ operation on the altered hidden state and the current input. The 'update gate', z_t, regulates the amount of past information to be retained and current information to be incorporated. This cycle continues till the amount of unfolding of the GRU netowrk. The network parameters i.e. the weights and biases at different gates are represented by symbols W and b with their respective subscripts indicating the gate operations. While backpropagating through time, the updates of these network parameters are also decided by the respective gating functions.

The training procedure for a Recurrent Neural network is also similar to that of Feed forward and Convolutional networks where an appropriate objective function (described in section A.2.2) suitable for the task is selected and minimized using a relevant optimizer. Training gated recurrent networks such as LSTM's and GRU's is more stable than the originally proposed RNNs because they suffer from the vanishing and in rare cases exploding gradient problems. Thus, gated recurrent networks are being widely used now-a-days for sequence modeling problems. Choosing between LSTM's and GRU's is a tough choice. However, empirical results indicate that GRU's perform slightly better than LSTM's in most cases and also require lesser computations [98].

A.2.7. Generative Modeling with Neural Networks

In generative modeling, a joint conditional probability distribution is modelled using different types of neural network architectures as described in segments A.2.4, A.2.5 and A.2.6. The joint probability of a data sample can be factorized as a product of conditional probabilities as follows:

$$p(x) = \prod_{t=1}^{T} p(x_t|x_{t-1}, x_{t-2}, ..., x_1, C), \tag{A.24}$$

where, the input data is denoted by x which can be either an image, audio waveform or embodiment of text information. C stands for the additional context input which acts as a conditioning input variable. For instance, while modeling audio, the speaker identity can be fed as an extra input to produce speech with the selected speakers voice. Similarly, for text-to-speech, extra information about text can be provided. The ideal approach to modeling conditional distributions $p(x_t|x_{t-1}, x_{t-2}, ..., x_1)$ over the individual samples would be to use mixture models such as Gaussian mixture models or mixture density networks [99]. However, recall that in segment A.2.1, the softmax transform is used mainly for mapping a feature vector to a probability distribution. The softmax distribution tends to perform better even when the data is implicitly continuous (as in the case of image pixel intensities or audio sample values) [19] [20]. The categorical distribution obtained from a softmax is more flexible and can more easily model arbitrary distributions because it makes no assumptions about their shape. After training such a network architecture to learn the distribution of data, samples can be generated by randomly sampling from the distribution.

Another way of generative modelling was introduced by Goodfellow, *et al.* known as Generative Adversarial Networks (GANs). The GAN architecture involves two sub-modules namely, a generator model that generates new examples and a discriminator model that classifies whether the example presented to it comes from the training data domain (real) or generated by the generator (fake). The concept of GANs comes from a game theoretic scenario in which the generator competes against anadversary. The generator network produces samples $x = g(z; \theta^{(g)})$, where z is the latent random variable. Whereas the discriminator emits a probability value given by $d(x, \theta^{(d)})$ indicating the probability that x is a real training example rather than a fake sample drawn from the generator model. The objective function of a GAN is formulated as:

$$\min_{\theta^{(g)}} \max_{\theta^{(g)}} \mathbb{E}_{X\sim P_D} \log(d(x)) + \mathbb{E}_{z\sim P_z} \log(1 - d(g(z))) \tag{A.25}$$

This drives the discriminator to learn to correctly classify samples as real or fake. Simultaneously, the generator attempts to foolthe classifier into believing its samples are real. At convergence, the samples generated by the generator model are indistinguishable from the real data. The discriminator may then be discarded. Training GANs can be very difficult in practice. Though there exist many solutions for stabilization of GAN learning, it still remains an open problem. It is also possible to train conditional GANs where the generator receives input not only from the latent space z but an additional known vector h.

B
Extra Experiments & Analysis

Most relevant experiments and their results have been discussed in chapters 4, 5 and 6. In this Appendix, some extra experiments, their outcomes and inferences have been discussed.

B.1. Experiments for Speech Synthesis

Here, the experiments and results associated with the WaveRNN synthesizer will be mentioned.

B.1.1. Train with Noisy MFCC context

The aim of this experiment was to verify if noisy MFCC features can provide suitable context for generation of clean speech. From chapter 3, it was concluded that MFCC features are not very robust to addition of noise and should be refrained from use. However, as per the time-line of the project, this experiment was conducted before the analysis revealed otherwise. Rather, it went the other way round during implementation, where the results obtained from this and a few other experiments led to an in depth analysis of different possible features. The reason to begin experimenting with MFCC features was inspired from literature in speech recognition that mention MFCC features to contain linguistic information in speech []. On this basis, it was believed that MFCC features would provide appropriate context. The configuration for this experiment has been mentioned in table C.1.

After training was complete, the MFCC features extracted from the noisy speech present in the test set are provided as context to the fully trained WaveRNN synthesizer. Based on this context, the synthesizer generates audio samples in an auto-regressive fashion as described in section 4.2. The results obtained are illustrated in figures B.1 and B.2 where, the waveforms of the clean, noisy and generated speech along with their respective spectrograms have been displayed. The generated speech waveform looks completely different from that of clean speech as expected from a generative model. Since, most objective evaluation metrics rely on waveform comparison, these metrics can fail to evaluate the quality of generated speech. Hence, only subjective evaluation can be provided in this case. On listening to the generated audio file from context provided by noisy MFCC features, the following observations were made:

1. Free from noise in most cases, irrespective of SNR as no noise characteristics were heard
2. In some segments, it blabbers and minces with the information being conveyed
3. The intonation (tone) and pitch keeps changing frequently within the same speech file, as a result, the voice jumps from male to female sounding talkers and vice versa
4. Unvoiced speech is slightly unclear
5. Some glitches and artifacts present during no speech activity or onsets/offsets but no distortions heard during speech activity

Some of these observations are also visible in the spectrogram of generated speech. For generated spectrograms in both figures B.1 and B.2, some sudden transitions from high energy to low energy and

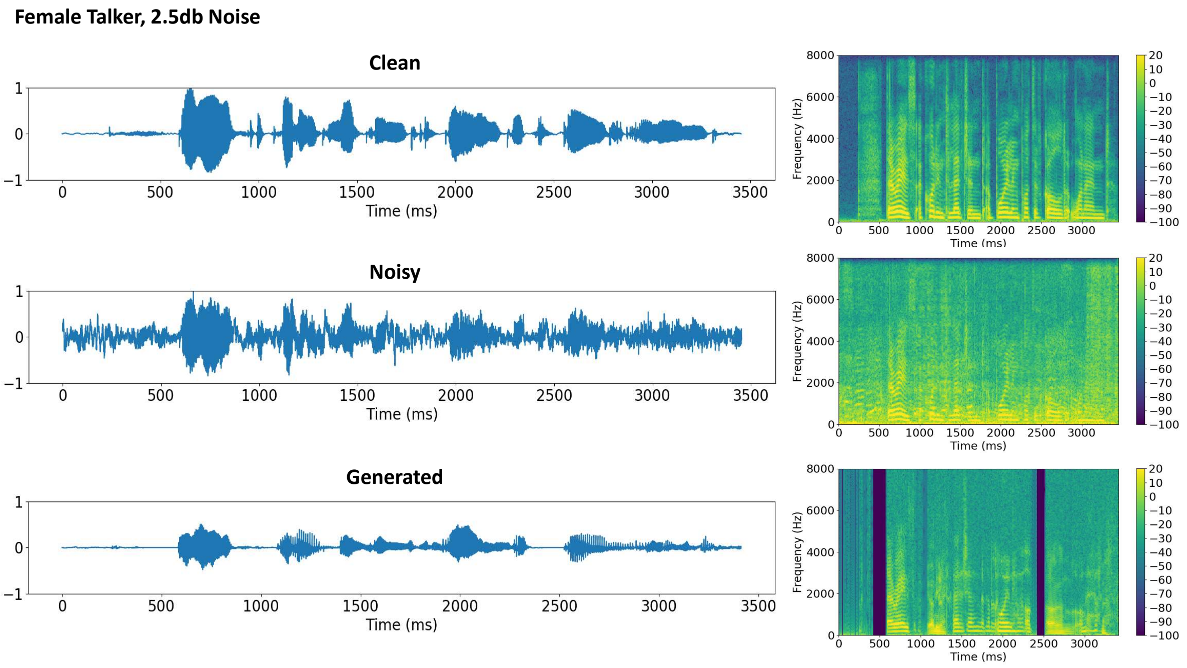


Figure B.1: **Top**: Clean speech waveform and its spectrogram for a female talker. **Middle**: Cafe noise added at 2.5dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Generated speech waveform and its spectrogram.

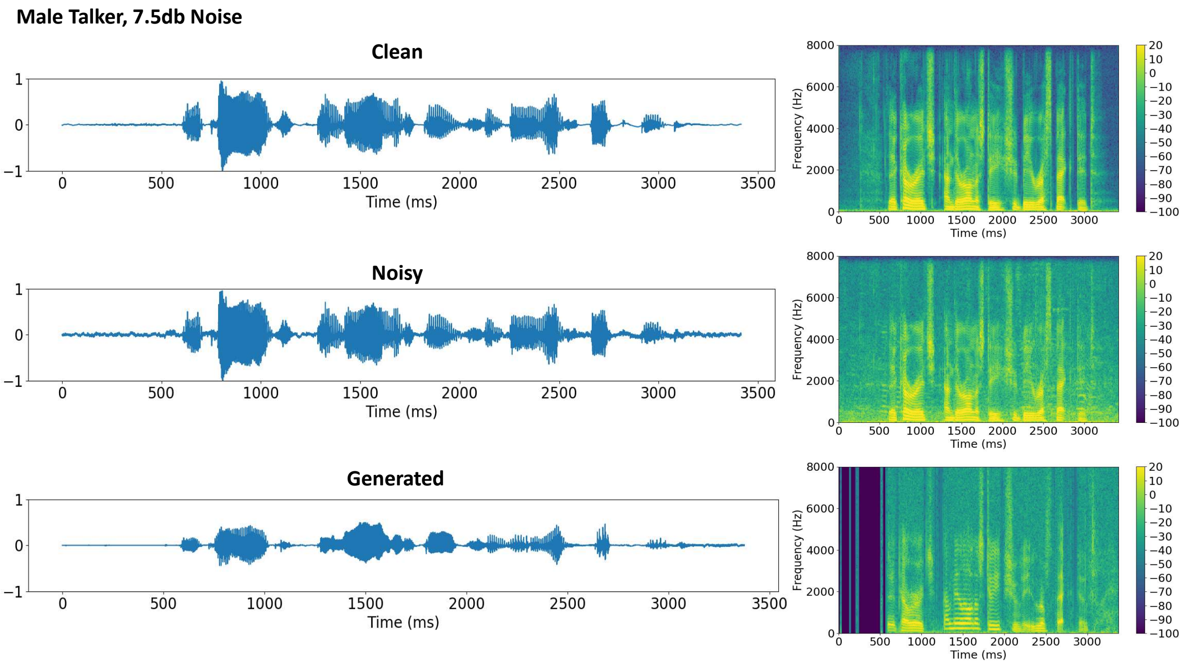


Figure B.2: **Top**: Clean speech waveform and its spectrogram for a male talker. **Middle**: Public Square noise added at 7.5dB SNR to form noisy speech waveform and its spectrogram. **Bottom**: Generated speech waveform and its spectrogram.

vice versa are observed along the time segments. These discontinuities correspond to sudden glitches and artifacts that are heard in generated speech. Additionally, the frequent voice change from male to female talkers is also visible in the spectrogram. For example, in figure B.2, for the spectrogram obtained from generated speech, it is clearly seen that the spacing between harmonics present in between time segments from 600ms and 1000ms is lesser than those between time segments from

1200ms to 1700ms. This represents a change from a male voice to a female voice. On comparing this with the spectrogram obtained from clean speech, the spacing between harmonics remains similar throughout the entire speech segment. Change in intonation is also visible, for example, in figure B.1 within the time segment from 600ms to 1000ms, the shape of the bands in the generated spectrogram is different from that in the clean spectrogram. Thus, the formants are located differently as a result the gradient changes over time and this indicates the change in tone.

The quality of generated speech sounds clean during speech activity but intelligibility in speech is totally lacking. The reasons attributed to such behaviour could be due to a smaller look back period i.e. the sequence length. Currently, the look back period is 1000 samples as mentioned in table C.1 which is roughly equal to 63ms. If the sequence length would be longer, it could capture more long term dependencies in speech and maybe reproduce more intelligible speech. Additionally, due to the presence of noise, perhaps, the network cannot capture the pitch information. Thus, the effect of noise must be reduced so that the network can capture more speech characteristics.

B.1.2. Increase Sequence Length and Reduce number of Mel Bands

This experiment is in continuation with noisy MFCC features. Based on the observations from the previous experiments, it was suspected that perhaps, the sequence length was not long enough to capture the long term dependencies in audio. On comparing with other methods such as the Speech Enhancement WaveNet [13] that has a receptive field of 4096 samples i.e. 256ms or the Speech Enhancement GAN [12] that takes 16384 samples i.e. more than 1 second of speech as input to its generator network, the look back period for the experiment in section B.1.1 seems rather too short. However, it has to be kept in mind that the sequence length cannot be as large as the the other techniques because the training time and complexity also increase with it. Hence, the aim of this experiment is to observe the effect of increasing the sequence length of the Gated Recurrent Unit (GRU) layer.

Additionally, there was another conjecture about the number of Mel bands in the Mel filter bank being large. It is because, if there are more number of bands in the filter bank, then the bandwidth of each band becomes less. When the Mel filter bank is applied on the spectrogram as described in chapter 3 section 3.4, a filter takes a weighted average over the spectral samples that fall within the bandwidth. Given the fact, that these filters are being applied on noisy features, a weighted average applied on these noisy features would tend to reduce the noise by a certain amount and hence the uncertainty in the input. If the bandwidth of a filter is small then the weighted average is calculated over a small number of samples and hence, the suppression in noise would be less. Thus, it is hypothesized that a smaller number of Mel bands in the filter bank would lead to certain improvement in quality. Hence, an experiment will also be conducted to prove this hypothesis. The configuration for this experiment as stated in table C.2 shows that the sequence length has now been changed to 1920 samples which is 120ms and the number of Mel bands is 40.

After training was complete, the MFCC features extracted from the noisy speech present in the test set are provided as context to the fully trained WaveRNN synthesizer. Based on this context, the synthesizer generates audio samples in an auto-regressive fashion as described in section 4.2. The results obtained are illustrated in figure B.3, where the spectrograms obtained from clean, noisy and generated speech for both male and female talker cases are shown. On listening to the audio files, certain observations were made as follows:

1. Completely free from noise, irrespective of SNR as no noise characteristics were heard
2. Abrupt discontinuities are no more present and the generated speech sounds free from artifacts and distortions
3. No improvement in intelligibility as generated speech still sounds mumbled and consonants are not reproduced correctly
4. The pitch and intonation of the speaker are different from the original clean speech example but the change in pitch and intonation has reduced considerably

These characteristics described are also observed in the spectrograms of generated speech shown in figure B.3. There are no more abrupt discontinuities observed in the spectrograms of generated speech

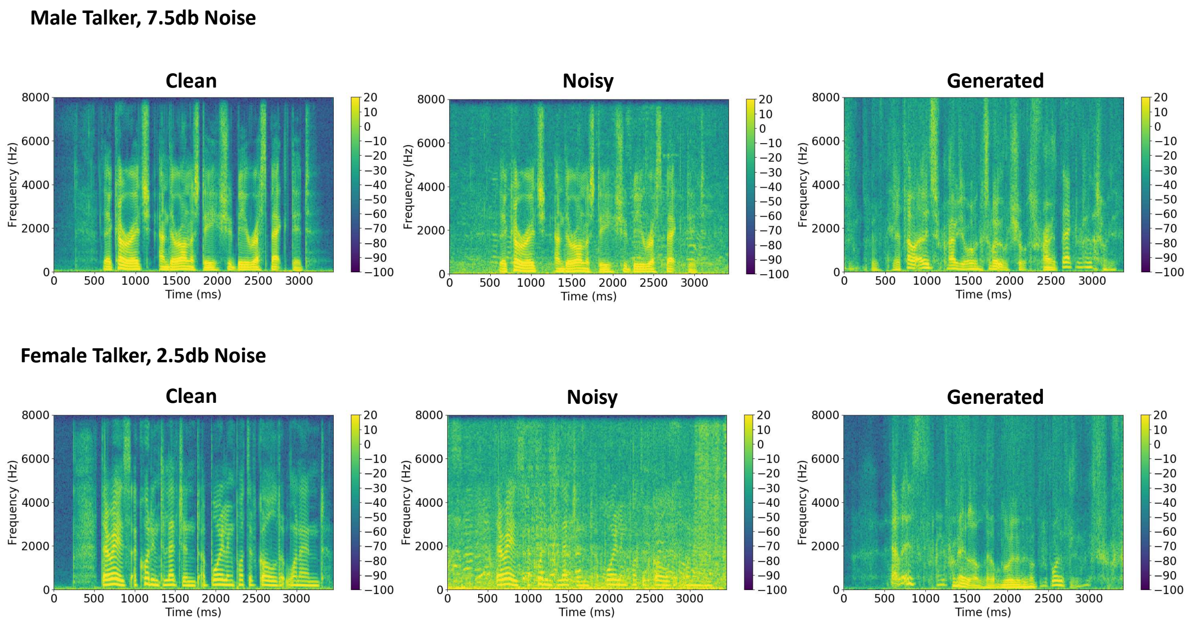


Figure B.3: **Top Left**: Spectrogram of clean speech for male talker. **Top Middle**: Spectrogram for speech corrupted with Public square noise at 7.5db SNR. **Top Right**: Spectrogram for generated speech of the example. **Bottom Left**: Spectrogram of clean speech for female talker. **Bottom Middle**: Spectrogram for speech corrupted by Cafe noise at 2.5db SNR. **Bottom Right**: Spectrogram for generated speech of the example.

for both examples. On comparing the spectrogram of clean and generated speech, the harmonics are spaced differently which show that the pitch of generated speech is different from the original. Change in voice still occurs but it is less frequent now. The tone of the speaker is also different from the reference clean speech and this is visible in the spectrogram from the location of the formants. The reason for such behaviour could be because the noise in the context is interfering with the speech information and hence the network cannot capture and learn those characteristics. Perhaps, MFCC features are severely affected by the presence of noise and this was confirmed by the studies in the domain of speech recognition where MFCC features are used abundantly []. Hence, there is a need to find features that are not as severely affected by noise as MFCC features.

B.1.3. Reduce sampling frequency of audio

The main purpose of conducting this experiment was mainly to reduce the training time of the network. As mentioned in experiment 4.5.2, the network was trained for approximately one week. Within the time span of this project, it did not seem feasible to wait one week for new results. Moreover, taking this step would allow to conduct more experiments and at a faster rate. Hence, the sampling rate of audio was reduced from 16kHz to 8kHz. It was speculated that by taking this measure, the training time would be a quadruple of the training time for experiment 4.5.2. Thus, the goal of this experiment was to solely reduce the training time. The configuration for this experiment is mentioned in table C.5. When the sampling rate is halved, the sequence length and the frequency range in the spectrogram are also halved. But while calculating the Mel spectrograms, the number of Mel bands was left the same. As a result, the training time was only halved and not quadrupled.

Ideally, since the frequency range is halved using fewer Mel bands would have sufficed. In figure B.4, different points are used to represent. Point B shows that when the sampling frequency is halved, using 60 Mel bands is enough to cover the frequency range from $0 - 4$ kHz, but still 80 Mel bands were used while training as depicted by point C. The usage of 80 Mel bands in the input context for speech sampled at 8 kHz led to sufficiently high quality speech generation. Hence, the curve can be extrapolated to point D for the case of speech with 16 kHz sampling rate. Hence, the number of bands used must be at least 105 bands. Thus, for new experiments with 16 kHz samping rate, 128 Mel bands

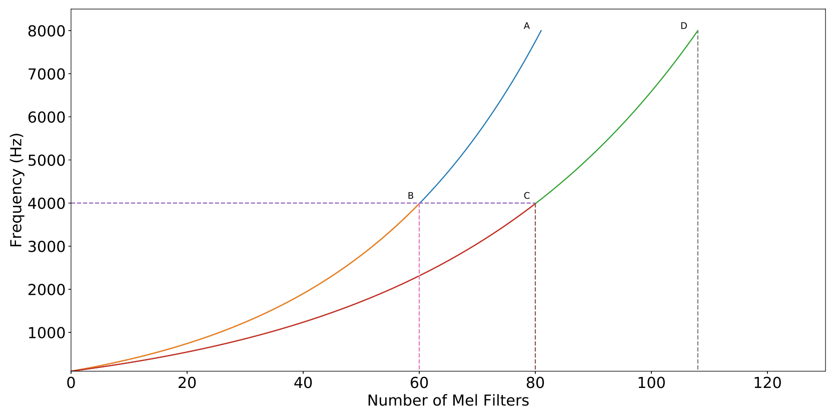


Figure B.4: Variation in center frequencies of Mel filter banks with number of filters

were used.

B.2. Experiments for Denoising Features

Here, the experiments and results associated with the denoising methods will be mentioned.

B.2.1. U-Net: Effect of using a 1-D kernel and no padding for spatial reduction

This experiment was conducted as a first attempt to setup a U-Net based design for the purpose of denoising the noisy Mel spectrogram features. As described in segment 5.1.1, the U-Net design is a fully convolutional architecture. In this configuration, as enlisted in table C.6, it is illustrated that this network incorporates an encoder and a decoder structure that consists of three 'Down Conv Unit' and 'Up Conv Unit' sub-modules each respectively. The convolution and transposed convolution operations share the same hyper-parameter value assignment to match the dimensionality of the input to the output of this network. In this case, the convolution kernel is applied only along the frequency dimension. The reason behind such a choice is motivated by the usage of spectral features that contain most meaningful information along the frequency dimension. Hence, the kernel should also learn to capture the essence of denoising by operating in the frequency domain. Additionally, to reduce the spatial dimensionality of the Mel spectrogram, no zero padding has been applied to the input or intermediate features. As a result, the application of convolution decreases the size of the features along the frequency dimension and the number of time segments are preserved due to the shape of the kernel. Note that, despite there being a reduction in spatial resolution of the features, it cannot be constituted as a bottleneck because the number of feature channels at the end of the encoder is large. This has been depicted in figure B.5.

B.2.2. U-Net: Effect of using a 2-D kernel and no spatial reduction

Inferences from the results of the previous experiment B.2.1, the use of a 1-D kernel did lead to significant noise reduction in the frequency bands, however, there were quite some observed discontinuities and abrupt changes along the time dimension. These could lead to unwanted artifacts. Thus, in order to make the transitions a bit more smooth along the time segments, the usage of a 2-D kernel is proposed. Moreover, it was realized that the reduction in spatial resolution is just because the boundary elements of the features were neglected due to the convolution operation in the network when zero padding is not applied. Hence, the true essence of the features may not be captured in the reduced space because the learnable filters would not traverse along the entire feature space despite the large number of channels. As a result, zero padding was applied in this experiment. Thus, there was no reduction in the spatial resolution at the output of any layer. The aim of this experiment was to understand the effect of using a 2-D kernel and no spatial reduction. The configuration used for training has been mentioned in table C.7. This has been depicted in figure B.6.

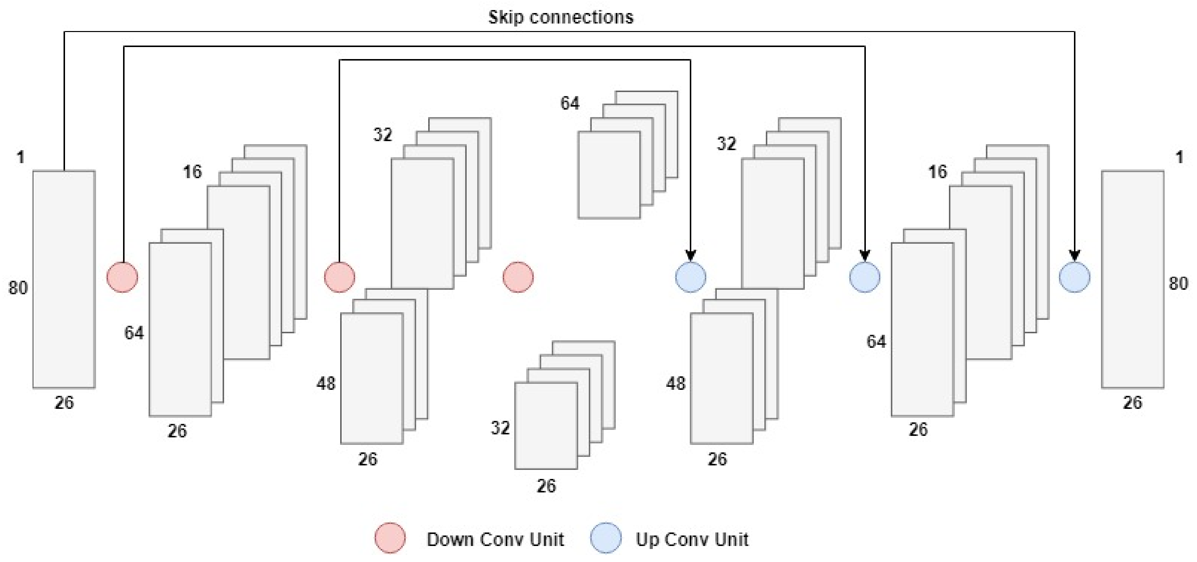


Figure B.5: Information flow in the U-Net architecture

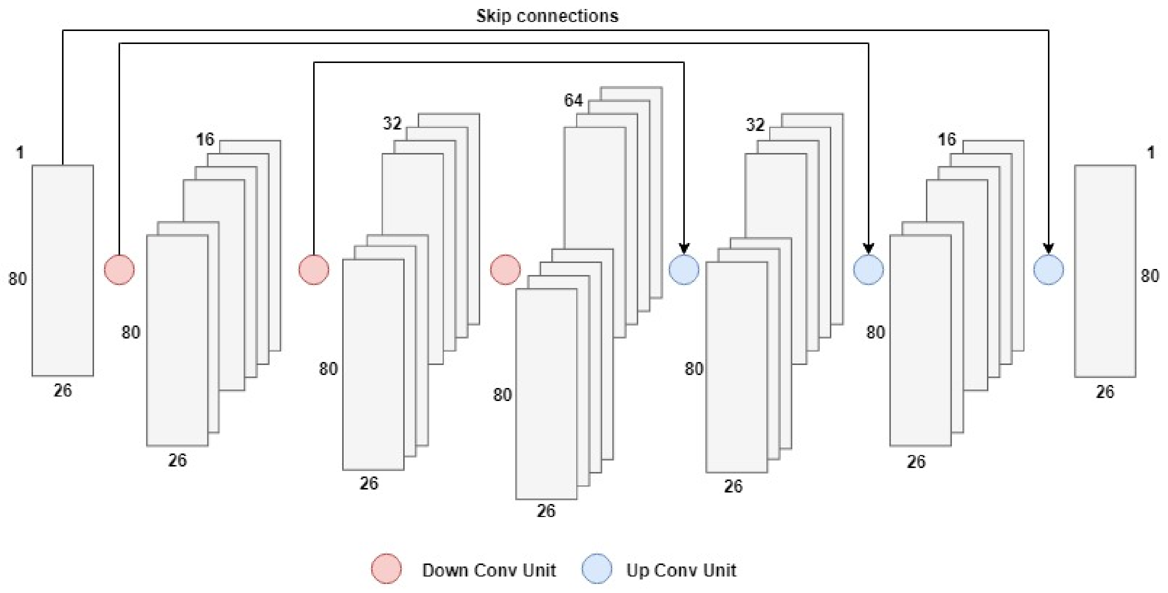


Figure B.6: Information flow in the U-Net architecture

B.3. Word Error Rate - Proposed System

This section shows the details of calculating the WER on the text responses from the clean, noisy and generated speech.

Table B.1: Speech Intelligibility based on Word Error Rate for selected examples from the test dataset with 4 types of noises added at 4 different SNRs.

Noise Level (dB)	Noise Type	Example		Speech-to-text Response	Word Error Rate (WER)		Comments
					wrt Reference	wrt Clean	
2.5	Public Square	p232_023	Reference	if the red of the second bow falls upon the green of the first the result is to give a bow with an abnormally wide yellow band since red and green light when mixed form yellow			The error in the text response for generated speech is more as compared to that for noisy speech. This shows a decrease in intelligibility.
			Clean	if the red of **a** second bow falls upon the green **at** the first the result is to give a bow with an abnormally wide yellow band since red and green light when mixed form yellow	2/36		
			Noisy	if the right of **a** second **phone calls** upon the green **at** the first the result is to give a bow with an abnormally wide yellow band since red and green light when mixed from yellow	6/36	4/36	
			WaveRNN	if **that is** the second **photo** falls upon the green of the first the result is to give a **bull** with an abnormally wide **yelp and its** red and green light when **next one** yellow	10/36	12/36	
2.5	Living	p257_054	Reference	we were surprised to see the photograph			The error in the text response for generated speech is less as compared to that for noisy speech. This shows a slight increase in intelligibility though the semantics have changed totally.
			Clean	**i was** surprised to see the photograph	2/7		
			Noisy	**whats the price of ticket prices**	7/7	7/7	
			WaveRNN	**what a surprise** to see the **question**	4/7	4/7	
7.5	Public Square	p232_103	Reference	we recognize the important role of golf in attracting visitors			There is a minor improvement in the error for generated speech as compared to that for noisy speech. This shows in improvement in intelligibility but the meaning of the sentence has changed.
			Clean	we recognize the important role of golf in attracting visitors	0/10		
			Noisy	recognize **seeing both control** of golf in attracting **visit**	5/10	5/10	
			WaveRNN	recognize the important role of **contracting** visitors	4/10	4/10	
7.5	Living	p232_013	Reference	some have accepted it as a miracle without physical explanation			The generated speech has no error when compared with the clean and reference text responses, thus showing an improvement in intelligibility.
			Clean	some have accepted it as a miracle without physical explanation	0/10		
			Noisy	**somehow** accepted it as a miracle without physical	3/10	3/10	
			WaveRNN	some have accepted it as a miracle without physical explanation	0/10	0/10	
7.5	Public Square	p257_101	Reference	however the intensive care unit at the southern general hospital was full			The WER does not show any improvement here. In fact the error rates for all clean, noisy and generated speech are the same. Hence, a valid comparison cannot be made.
			Clean	**i was at** the intensive care unit the southern general hospital was full	4/12		
			Noisy	however the intensive care unit **7th** general hospital **is** full	4/12	6/13	
			WaveRNN	however the intensive care unit the southern general **hospitals home**	4/12	6/13	
12.5	Cafe	p257_369	Reference	there is a solution she believes			The metric shows a slight improvement in the error rate of generated speech as compared to noisy speech.
			Clean	**theres** a solution she believes	2/6		
			Noisy	**theres** a solution **tree please**	4/6	2/5	
			WaveRNN	**theres** a solution she **please**	3/6	1/5	
12.5	Living	p232_054	Reference	several other pupils and staff were seriously injured in the accident			The WER for the noisy and the clean speech wrt the reference text are the same since their text response is also the same. However, the text response of the generated speech is slightly closer to the reference and further away from the clean. Hence, a minor improvement is observed.
			Clean	several **of the** pupils and staff **was** seriously injured in the accident	3/11		
			Noisy	several **of the** pupils and staff **was** seriously injured in the accident	3/11	0/12	
			WaveRNN	several other **peoples** and staff **was** seriously injured in the accident	2/11	3/12	
17.5	Cafe	p232_029	Reference	their courage and their honesty should be respected			There is no improvement in intelligibility as suggested by the metric.
			Clean	their courage and their honesty should be respected	0/8		
			Noisy	**acreage** and their honesty should be respected	2/8	2/8	
			WaveRNN	**acreage** and their honesty should be respected	2/8	2/8	
17.5	Cafe	p257_127	Reference	his son has been travelling with the tartan army for years			The error rate for text obtained from clean speech wrt to the reference is significant. The metric shows an overall decrease in intelligibility for generated speech.
			Clean	his son has been traveling with the **top nami** for **use**	4/11		
			Noisy	**hey** son its been traveling with the **top money** for **you**	6/11	4/11	
			WaveRNN	**your** son has been traveling with **a todd lonely on** for **he is**	8/11	7/11	

C
Experiment Configurations

This appendix contains configurations for the conducted experiments.

C.1. Configurations for Synthesizer experiments

The configurations for the experiments conducted with the WaveRNN synthesizer are mentioned here.

C.1.1. WaveRNN with noisy MFCC as context

Table C.1: Configuration for training WaveRNN synthesizer with noisy MFCC as context

	Parameter Name	Format	Value
Network Hyper-parameters			
	GRU units	R	512
	Fully Connected layer 1 units	F	512
	Fully Connected layer 2 units	P	256
	Upsample factors	$[U_0, U1, ..., Un]$	[5, 5, 8]
	Sequence Length	S_L	1000
	Batch size	b	64
	Epochs	e	2000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Noisy MFCC
	Mel bands	M	80
	MFCC bands	mf	40
	Mel window	w	9
	Sampling Frequency	sr	16000
	Window Length		400
	Hop Length	h	200
	FFT size	N	512

C.1.2. Increase sequence length of WaveRNN input

Table C.2: Configuration for training WaveRNN synthesizer for an increased sequence length and reduced Mel bands

	Parameter Name	Format	Value
Network Hyper-parameters			
	GRU units	R	512
	Fully Connected layer 1 units	F	512
	Fully Connected layer 2 units	P	256
	Upsample factors	$[U_0, U1, ..., Un]$	[5, 8, 8]
	Sequence Length	S_L	1920
	Batch size	b	64
	Epochs	e	2000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Noisy MFCC
	Mel bands	M	40
	MFCC bands	mf	40
	Mel window	w	10
	Sampling Frequency	sr	16000
	Window Length		640
	Hop Length	h	320
	FFT size	N	1024

C.1.3. WaveRNN with noisy Mel spectrogram as context

Table C.3: Configuration for training WaveRNN synthesizer with noisy Mel spectrogram

	Parameter Name	Format	Value
Network Hyper-parameters			
	GRU units	R	512
	Fully Connected layer 1 units	F	512
	Fully Connected layer 2 units	P	256
	Upsample factors	$[U_0, U1, ..., Un]$	[5, 8, 8]
	Sequence Length	S_L	1920
	Batch size	b	64
	Epochs	e	2000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Noisy Mel Spectrogram
	Mel bands	M	40
	Mel window	w	10
	Sampling Frequency	sr	16000
	Window Length		640
	Hop Length	h	320
	FFT size	N	1024

C.1.4. WaveRNN with clean Mel spectrogram as context

Table C.4: Configuration for training WaveRNN synthesizer with clean Mel spectrogram

	Parameter Name	Format	Value
Network Hyper-parameters			
	GRU units	R	512
	Fully Connected layer 1 units	F	512
	Fully Connected layer 2 units	P	256
	Upsample factors	$[U_0, U1, ..., Un]$	[5, 8, 8]
	Sequence Length	S_L	1920
	Batch size	b	64
	Epochs	e	6000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Clean Mel Spectrogram & Delta Features
	Mel bands	M	80
	Mel window	w	10
	Sampling Frequency	sr	16000
	Window Length		640
	Hop Length	h	320
	FFT size	N	1024

C.1.5. Reduce sampling frequency

Table C.5: Configuration for training WaveRNN synthesizer for 8kHz speech

	Parameter Name	Format	Value
Network Hyper-parameters			
	GRU units	R	512
	Fully Connected layer 1 units	F	512
	Fully Connected layer 2 units	P	256
	Upsample factors	$[U_0, U1, ..., Un]$	[5, 8, 8]
	Sequence Length	S_L	960
	Batch size	b	64
	Epochs	e	2000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Clean Mel Spectrogram & Delta Features
	Mel bands	M	80
	Mel window	w	10
	Sampling Frequency	sr	8000
	Window Length		320
	Hop Length	h	160
	FFT size	N	512

C.2. Configurations for Denoising experiments

The configurations for the experiments conducted with the Denoising architcetures are mentioned here.

C.2.1. U-Net architecture with 1-D kernels and spatial reduction

Table C.6: Setup and Configuration for the U-Net architecture with 1-D kernels and spatial reduction

	Parameter Name	Format	Values
Network Hyper-parameters			
	Convolution Kernel size	$[k_f, k_t]$	[17, 1]
	Encoder Channels	$[C_1, C_2, ..., C_n]$	[1, 16, 32, 64]
	Decoder Channels	$[C_n, C_{n-1}, ..., C_1]$	[64, 32, 16, 1]
	Stride	$[s_f, s_t]$	[1, 1]
	Padding	$[p_f, p_t]$	[0, 0]
	Batch size	b	64
	Epochs	e	2000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Noisy Mel Spectrogram
	Mel bands	M	80
	Mel Window	w	26
	Sampling frequency	sr	8000
	Window length		320
	Hop length		160
	FFT size	N	512

C.2.2. U-Net architecture with 2-D kernels and no spatial reduction

Table C.7: Setup and Configuration for the U-Net architecture with 2-D kernels and no spatial reduction

	Parameter Name	Format	Values
Network Hyper-parameters			
	Convolution Kernel size	$[k_f, k_t]$	[15, 3]
	Encoder Channels	$[C_1, C_2, ..., C_n]$	[1, 16, 32, 64]
	Decoder Channels	$[C_n, C_{n-1}, ..., C_1]$	[64, 32, 16, 1]
	Stride	$[s_f, s_t]$	[1, 1]
	Padding	$[p_f, p_t]$	[7, 1]
	Batch size	b	64
	Epochs	e	2000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Noisy Mel Spectrogram & Delta Features
	Mel bands	M	80
	Mel Window	w	26
	Sampling frequency	sr	8000
	Window length		320
	Hop length		160
	FFT size	N	512

C.2.3. U-Net architecture with 2-D kernels and strided convolutions

Table C.8: Setup and Configuration for the U-Net architecture with strided convolutions

	Parameter Name	Format	Values
Network Hyper-parameters			
	Convolution Kernel size	$[k_f, k_t]$	[15, 3]
	Encoder Channels	$[C_1, C_2, ..., C_n]$	[1, 16, 32, 64, 128]
	Decoder Channels	$[C_n, C_{n-1}, ..., C_1]$	[128, 64, 32, 16, 1]
	Stride	$[s_f, s_t]$	[2, 1] for alternate channels
	Padding	$[p_f, p_t]$	[7, 1]
	Batch size	b	64
	Epochs	e	4000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Noisy Mel Spectrogram & Delta Features
	Mel bands	M	80
	Mel Window	w	26
	Sampling frequency	sr	8000
	Window length		320
	Hop length		160
	FFT size	N	512

C.2.4. VCAE with Dense Bottleneck

Table C.9: Configuration for the VCAE architecture with Dense Bottleneck

	Parameter Name	Format	Values
Network Hyper-parameters			
	Convolution Kernel size	$[k_f, k_t]$	[15, 3]
	Encoder Channels	$[C_1, C_2, ..., C_n]$	[1, 16, 32, 32, 64]
	Decoder Channels	$[C_n, C_{n-1}, ..., C_1]$	[64, 32, 32, 16, 1]
	Stride	$[s_f, s_t]$	[2, 1] for all channels
	Padding	$[p_f, p_t]$	[7, 1]
	Type of Bottleneck		Dense
	Dense Bottleneck layers	$[n_1, ..., n_L, ..., n1]$	[16640, 1024, 16640]
	Variance constraint	v	1024
	Standard deviation of noise	σ	0.02
	Batch size	b	64
	Epochs	e	4000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Noisy Mel Spectrogram & Delta Features
	Mel bands	M	80
	Mel Window	w	26
	Sampling frequency	sr	8000
	Window length		320
	Hop length		160
	FFT size	N	512

C.2.5. Fully convolutional VCAE

Table C.10: Setup and Configuration for the fully convolutional VCAE architecture

	Parameter Name	Format	Values
Network Hyper-parameters			
	Convolution Kernel size	$[k_f, k_t]$	[15, 3]
	Encoder Channels	$[C_1, C_2, ..., C_n]$	[1, 16, 32, 64, 16, 4]
	Decoder Channels	$[C_n, C_{n-1}, ..., C_1]$	[4, 16, 64, 32, 16, 1]
	Stride	$[s_f, s_t]$	[2, 1] for all channnels
	Padding	$[p_f, p_t]$	[7, 1]
	Type of Bottleneck		Convolutional
	Dense Bottleneck layers	$[n_1, ..., n_L, ..., n1]$	-
	Variance constraint	v	520
	Standard deviation of noise	σ	0.02
	Batch size	b	64
	Epochs	e	4000
	Learning rate	lr	1e-4
Feature configurations			
	Feature		Mel Spectrogram & Delta Features
	Mel bands	M	80
	Mel Window	w	26
	Sampling frequency	sr	8000
	Window length		320
	Hop length		160
	FFT size	N	512

C.2.6. Final System

Table C.11: Setup and Configuration for the final speech enhancement system.

	Parameter Name	Format	Value
Network Hyper-parameters			
WaveRNN	GRU units	R	512
	Fully Connected layer 1 units	F	512
	Fully Connected layer 2 units	P	256
	Upsample factors	$[U_0, U1, ..., Un]$	[5, 8, 8]
	Sequence Length	S_L	1920
U-Net	Encoder Channels	$[C_1, C2, ..., Cn]$	[1, 16, 32, 64, 128]
	Decoder Channels	$[C_n, Cn-1, ..., C1]$	[128, 64, 32, 16, 1]
	Convolution Kernel size	$[k_f, k_t]$	[15, 3]
	Stride	$[s_f, s_t]$	[2, 1] for all channels
	Padding	$[p_f, p_t]$	[7, 1]
	Batch size	b	64
	Epochs	e	6000 (pre-training) 12000 (fine-tuning)
	Learning rate	lr	1e-4 (pre-training, first 4000 epochs) 5e-5 (pre-training, after 4000 epochs) 1e-4 (fine-tuning, first 8000 epochs) 5e-5(fine-tuning, after 8000 epochs)
Feature configurations			
	Feature		Noisy Delta Features
	Mel bands	M	128
	Mel window	w	10
	Sampling Frequency	sr	16000
	Window Length		640
	Hop Length	h	320
	FFT size	N	1024

D
Hardware Specifications

This chapter includes specifications of the hardware used for training and testing the proposed systems.

Table D.1: GPU Specifications used for training and testing.

GPU Model	NVIDIA GeForce RTX 2080 Ti
CUDA Cores	4352
Clock Frequency	1635 MHz
Memory Interface	352-bit
Memory Capacity	11 GB
Memory Bandwidth	616 GB/s
Power	250 W
FP16 (half) performance	26.90 TFLOPS
FP32 (float) performance	420.2 GFLOPS