Adaptive Microphone Array Systems With Neural Network Applications

Jazmine Marisol Covarrubias
University of Texas at El Paso

ADAPTIVE MICROPHONE ARRAY SYSTEMS WITH NEURAL NETWORK APPLICATIONS

JAZMINE MARISOL COVARRUBIAS

Master's Program in Computer Engineering

APPROVED:

Rodrigo Romero, Ph.D., Chair

Sergio Cabrera, Ph.D.

Natalia Villanueva-Rosales, Ph.D.

Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School

ADAPTIVE MICROPHONE ARRAY SYSTEMS WITH NEURAL NETWORK APPLICATIONS

by

JAZMINE M. COVARRUBIAS,B.E

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

December 2019

Acknowledgements

I would like to express my greatest appreciation to my committee chair, Dr. Rodrigo A. Romero, for his guidance throughout the development process of this thesis. I also want to thank the rest of my committee members: Dr. Sergio D. Cabrera and Dr. Natalia Villanueva Rosales for their insightful comments and questions. Additionally I want to thank Mr. George Salazar, P.E., ESEP ,LSM-IEEE, Human-Computer Interface Technical Discipline Lead, NASA/JSC, for providing the initial motivation and requirements for my research work. I also want to give my gratitude to the Texas Instruments Foundation Endowed Scholarship for supporting this research. Last but not least, I want to thank my family for their love and support and Raul Ontiveros for helping me through this journey.

Abstract

A microphone array integrated with a neural network framework is proposed to enhance and optimize speech signals derived from environments prone to noise and room reflections that cause reverberation. Microphone arrays provide a way to capture spatial acoustic information for extracting voice input from ambient noise. In this study, we utilize and analyze established signal processing methods combined with different neural network architectures to achieve denoised and dereverberated speech signal results that are comparable with their clean, anechoic versions. The first stage of the proposed system involves using datasets containing anechoic speech recordings of speech utterances and convolving them with a collection of different room impulse responses to simulate reverberation. Noisy speech signals are also added to the clean speech to simulate noisy environments. These datasets are used for training and testing speech enhancement methods of our combined system. The goal of this work is to develop an approach to process and remove reverberation present in voice signals to achieve adaptability and optimal signal-to-noise ratio for any stationary and non-stationary interference including white noise, impulse noise, and reverberations. CNN and FCNN neural network architectures are structured to optimize learning or improve other performance metrics based on attributes such as layer connections, types of layers, number of neurons in each layer, activation functions, and cost functions. While preserving signal integrity, this work aims to enhance the quality and intelligibility of the extracted speech. Our experimental results show that, while more computationally intensive, the CNN outperformed the FCNN in training and audible quality.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

As deep space crewed endeavors become more venturous, the complexity of spacecraft systems will need to evolve to minimize the dependency on Earth-based mission control centers with their convenient, low-latency voice-based technical support while transitioning into fully stand-alone, self-sufficient systems. Communications from within a spacecraft or spacesuit, both for intra- and extravehicular activities, are essential for carrying out missions as well as maintaining the overall wellbeing of astronauts. Speech signals in space environments may present some unique processing problems due to environmental noise. For instance, environmental parameters, such as reflective surface types and geometries, can degrade speech signals. While methods for denoising speech signals such as linear filtering, adaptive filtering, frequency-domain noise cancellation, and smoothing algorithms have been developed, dealing with reverberations has proven more difficult. One method of reducing reverberation from voice signals is to use beamforming with a microphone array, which can capture signals with different angles of reflection in a room, to clean out a signal from ambient noise or any other interference. However, this may not lend itself to dynamic changes in the environment such as significant variations of speaker position, amount and type of reflective surfaces, and geometry. Recent approaches for reducing reverberation effects rely on advanced signal processing techniques and machine learning – especially, deep-learning techniques to accommodate a large variety of dynamic changes in the environment – including back- and front-end techniques such as feature enhancement using mapping-based and masking-based methods. Therefore, *is it possible to develop an adaptable system to deliver intelligible speech in reverberant environments?*

1.1 Problem Statement

The goal of this work is to develop an approach to process and remove reverberations that are present in voice signals from a microphone array that can achieve adaptability and optimal signal-to-noise ratio (SNR) for any stationary and non-stationary interference including white noise, impulse noise, and reverberations. A microphone array consists of multiple omnidirectional/directional microphones that capture and transmit speech signals from different directions. These signals can then be processed through signal processing techniques such as filtering and masking or transformed as when using STFT to get magnitude and phase representations of important speech features. Subsequently, these features can be vectorized to train neural networks to produce clean signals from noisy or reverberant signals. There is currently a wide variety of effective noise removal methods in use and so the focus of this research will solely be on the removal of reverb which presents a range of challenges. Neural network architectures can be structured to optimize learning or improve other performance metrics based on attributes such as layer connections, types of layers, number of neurons in each layer, activation functions, and cost functions. While it is important to keep the integrity of the signal intact, enhancing the quality and intelligibility of the extracted speech are the main focus of this work. The main types of interference that can degrade a speech signal are reverberation and background noise. Both types of interference can cause inaccuracies in speech processing due to echoes and variance in coloration introduced into the source signals [2].

1.2 Processing Approach

The presented approach uses neural networks to process microphone array output to improve system scalability and adaptability for source distance and variations of environmental noise levels. Additionally, signal processing techniques are implemented to capture dominant feature representations for training and validating the selected neural networks. MATLAB [7] is used for signal processing and deep learning experiments with the Audio System Toolbox, the Signal Processing Toolbox, and the Deep Learning Toolbox. Networks were trained using a dataset containing thousands of recordings of short sentences spoken with different accents by different age groups of male and female voices. The dataset is derived from the Mozilla Common Voice dataset [8], which is a collection of 48 kHz recordings, downsampled to 8kHz to reduce the computational load. As shown in Figure [1], the target and predictor signals input to the neural network represent the magnitude spectra of the noisy and clean speech signals. These signals are generated by applying a Short-Time Fourier transform (STFT) with a window length of 256 samples, which are reduced to 129 by removing samples corresponding to negative frequencies. The predictor input has 8 STFT vectors and the target input contains a single vector to enable computing the STFT output estimate based on the current noisy STFT and the overlap of 7 previous noisy STFT vectors. A tall array is used to speed up transformations for extracting features for all the speech files in the datastore. A fully connected neural network and a convolutional neural network are used to process these tall arrays as 129 x 8 x 1 input images for training with 1% reserved for validation to prevent overfitting. The output of the networks is the magnitude spectrum of the denoised voice signal.

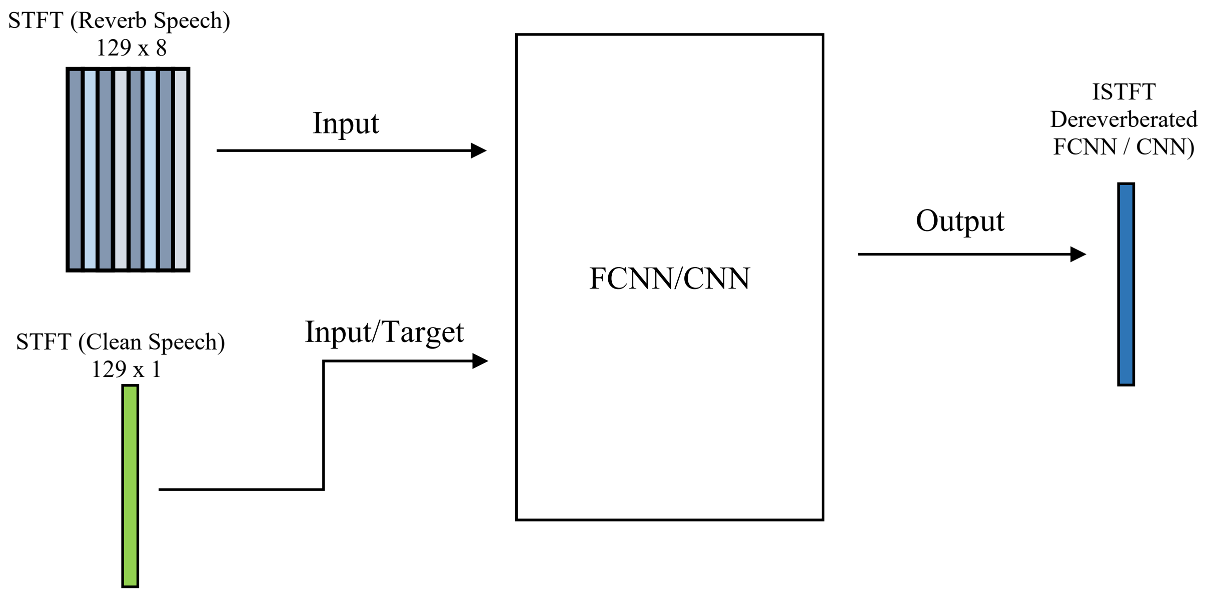

Figure 1.1: Representation of clean and reverberated STFTs as inputs to FCNN/CNN and dereverberated vector output.

1.3 Related Work

Ever since communication systems and technologies have been in existence, many advancements have been made to continuously improve speech recognition and intelligibility. Some areas of research involve utilizing different signal processing techniques and machine learning to address challenges posed by environments that may degrade the intelligibility of speech and impact reconstruction quality. The following articles are representative of recent work where neural networks are at the core of different methods of extracting speech from noisy signals, specifically those distorted by reverberant components. The first article uses a deep neural network (DNN) based on a spectral mapping and trained using inputs with optimal frame shift and varying acoustic context window sizes. This DNN has linear activation functions in the output layer. The second article focuses in on feature mapping techniques. The third article talks about using estimation of a complex ideal ratio mask in real and imaginary domains to enhance reverberant speech. Finally, the fourth article presents an overview of recent work on environmentally robust speech recognition systems.

1.3.1 A Reverberation-Time-Aware Approach to Speech Dereverberation Based on Deep Neural Networks," IEEE/ACM Trans. (B. Wu, K. Li, M. Yang, and C.-H. Lee)

Wu et al. investigate the use of a frame shift and context window aware approach to observe their effects on DNNs recovery of anechoic speech from reverberant environments. The authors argue that training DNNs using STFT to generate spectrograms from anechoic and noisy speech signals for feature extraction will only generate results for a fixed temporal resolution and will not generalize to different types of reverberant reflections. Instead, they suggest training DNNs with one type of frame shift and expansion at a time, as they claim that both affect the quality of the results. They used the perceptual evaluation of speech quality (PESQ) scores to determine which frame shift, denoted as (R), and frame expansion, denoted as (N), or acoustic context to use for RT60s.

The authors trained a DNN with anechoic and reverberant speech to dereverberate speech by extracting the phase of the reverberant speech and reconstructing from an estimated spectral magnitude. Their DNN uses sigmoid activation and min-max normalization. Instead of using spectral mapping learning to predict the log spectral magnitude of a dereverberated signal, they attempt to improve this method by using a linear output in the output layer and globally normalized target features while also including optimal frame shift and context window sizing parameters. They select their optimal frame shifts and context windowing parameters using a as reference a nonlinear DNN based regression model. Their DNN improves the estimated restoration of anechoic spectrogram by using a reverberation time aware DNN (RTA-DNN) to capture the frame-wise effects of temporal correlation in different reverberant environments and the acoustic context window size for speech continuity enhancement. The TIMIT dataset, which has 4620 training utterances, was used to train the network. Testing parameters were a 6x3 meter room, loudspeaker position (2, 3, 1.5 meters), microphone position (4, 1, 2 meters), and 10 RIRs with improved image source method in the 0.1–1.0s RT60 range. For this research, the Edinburg dataset of anechoic speech will be used to train the networks along with creating reverberant versions by convolving them with impulse responses.

1.3.2 Learning Spectral Mapping for Speech Dereverberation and Denoising IEE (Woods).

Han et al. point out that simply using an inverse filter is not sufficient for dereverberation. The inverse filter itself must be estimated which does not tend to be replicated successfully in realistic applications. Instead of doing a direct inverse STFT to reconstruct the signal in the time domain, which can result in obtaining a different magnitude spectrum than intended, they use an algorithm that iterates through the STFT by replacing the phase of the noisy phase with its inverse STFT to find the closest match to the provided magnitude spectrum. They use a DNN that produces estimates of clean speech log spectrum magnitudes. Their network implementation is based on feature extraction, model training, and post-processing. Feature extraction uses a STFT window of 20ms with a frame shift of 10ms and 320 points with 161 frequency bins. They use neighboring frames of spectral features to account for temporal optimization. The output of the network is intended to be the spectrogram of anechoic speech at current frame. This frame is represented by a 161-feature vector with log magnitudes of each frequency. Their neural network is trained using the 161x11 units of the log magnitude spectrum of noisy signals and mean squared error for optimization. The network has three hidden layers with 1600 neurons each. Inputs are normalized to zero mean and unit variance and outputs are normalized to values between 0 and 1. Rectified linear unit (RELU) is the activation function for the hidden layers and sigmoid functions for the output layer. Weights do not have a standardized initial value; rather, they are randomized. Backpropagation handles 512 samples per mini batch with stochastic gradient descent. They use several methods to test the intelligibility of their results including frequency-weighted segmental speech to-noise ratio (SNRfw), the short-time intelligibility measure (STOI), and finally the perceptual evaluation of speech quality (PESQ) with a (-05 to 4.5) scoring range. They experiment with the number of hidden layers and frame window sizes, which affect the DNN's performance. Their simulated room parameters for measuring dereverberation performance are 10m x 7m x 3m with T60 conditions of 0.3s, 0.6s, and 0.9s tested with two different simulated room impulse responses (RIRs). Their training set consists of 1200 utterances from 200 anechoic versions provided by a single female speaker convolved with the two RIRs and each of the reverberation times. The test is a set of 60 reverberant utterances with 20 utterances each convolved with one RIR and three T60s. For cross-validation, another set of 10 utterances combined with all three T60s and RIRs was developed. Their method was tested against three other dereverberation methods which employ a binary masking method, estimated inverse filters, spectral subtraction, and finally an ideal binary masking method as a baseline for binary masking systems. Their results show that their proposed method had an average improvement in the SNR and STOI scores as well as PESQ scores when it came to long reverberation times. They did some more testing to observe the proposed DNNs ability to generalize by changing up the T60s of the RIRs, evaluating with the TIMIT corpus, and using real recorded RIRs.

1.3.3 Training Deep Neural Networks for Reverberation Robust Speech Recognition (M. Ritter et al., M. Müller, S. Stüker, F. Metze, A. Waibel)

Authors train DNNs to deal with multiple reverberating environments by using multiple RIRs in combination with clean speech and various noise sources. The run their experiments for dealing with reverb utilizing the Janus recognition toolkit and IBIS decoder. Their DNNs are implemented using the Theano python tool library. The dataset used for training was from the TED-LIUM corpus combined with 10 hours of noise. Also, they used 658 RIRs derived from different sources along with RIRs generated with “Room Impulse Response Generator” tool by E. Habets. The evaluation set consisted of TED talks recorded during the International Workshop on Spoken Language Translation (IWSLT) 2013. Their proposed DNN had five pre-trained hidden layers with 1200 neurons, each with the sigmoid activation function, followed by a softmax output layer. For this research the proposed DNN consists of an image input layer with two hidden layers with 1024 neurons each followed by a batch normalization and RELU layer and an output fully connected layer followed by a regression layer.

1.3.4 Time Frequency Masking in the Complex Domain for Speech Dereverberation and Denoising.

A complex ideal ratio mask (cIRM) is used for learning spectral mapping from reverberant speech signals. For every time frame, features are extracted using different methods such as amplitude modulation spectrogram (AMS), relative spectral transform and perceptual linear prediction (RASTA-PLP), mel-frequency cepstral coefficients (MFCC), and gammatone filterbank energies. Neighboring frames are incorporated to account for temporal information. The proposed cIRM acts as an inverse filter equal to the ratio of the real and imaginary components of the target anechoic speech and the reverberant speech, which enhances both the magnitude and the phase of reverberant speech. A hyperbolic tangent function is used to compress the real and imaginary values for supervised learning. DNN inputs are normalized to zero mean and unit variance. Input features from the feature vector are filtered using the auto-regressive moving average (ARMA). The activation function for the hidden layers is the rectified linear unit function

(ReLU). There are two output layers that handle the real and imaginary components using linear activation functions.

Chapter 2: Theoretical Background

2.1 Sound, Acoustics, and Noise

Reverberation is the accumulation of sound reflections from surfaces in a closed environment. A listener in such an environment would hear a combination of sound traveling directly from a source and reverberation, i.e., the sum of all reflections off reflective surfaces. An important parameter to characterize reverberation is reverberation time (RT), which is how long it takes for reverberant sound energy to decrease 60 dB from its original level after its driving sound signal ends. Reverberation is in large part shaped by the geometry of an enclosure and the reflective properties of the surfaces it contains. There can be many reflections consisting of decaying attenuated copies of the source at a time.

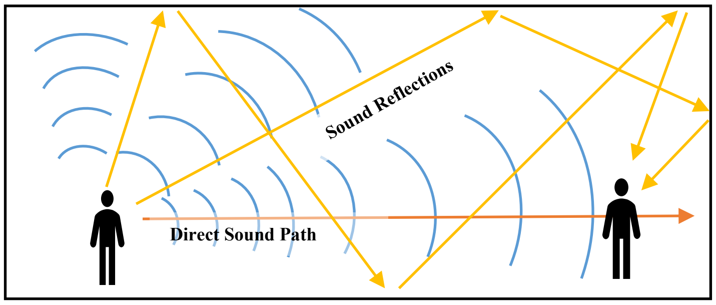

Figure 2.1: Reverberation effects from environment

In this study we will use convolution reverberation, which means taking an anechoic speech file and convolving it with the impulse response of different types of rooms. There are different variables to consider when using a type of reverberation. Typical variables include depth (room size), as bigger rooms have longer lasting reflection; decay (RT60), which measures the amount of time it takes for the reflected signal to die down; predelay, which is the time from initial source emission to the point earliest reflections are heard; diffusion, which describes the complexity of the environment in terms of its number of different surfaces and the different reflections they can produce; and the wet or dry mix, where reverb signals are the wet mix and the dry signal is the anechoic speech.

The impulse response is the frequency response of a given environment. A convolution of an impulse response and an anechoic speech signal replicates the reverberation effects the room would have when exposed to such a signal. The impulse response is an observation of a system reaction to an impulse input. Noise can take many forms that can be the cause of any unwanted disturbances in a particular signal. Noise can be uniform and non-uniform (impulse)

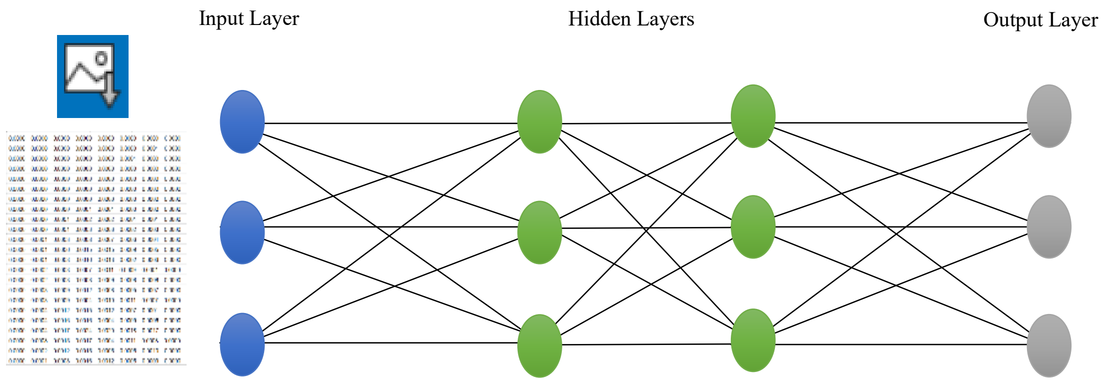

Figure 2.2: Fully connected neural network architecture

2.2 Deep Learning

Neural network architectures considered for this research are a fully connected neural network (FCNN) and a convolutional neural network (CNN). Both are supervised regression models that converge as close to the target as possible. The FCNN typically has different layers, each containing an interconnected network of neurons which represent a node holding a certain value. The inputs to the FCNN are normalized to speed up learning and getting passed along the two hidden networks consisting of FCNNs.

Neurons in an FCNN have connections between them with weighted values that are multiplied by the neuron value while adding a bias value when present. Each hidden layer is followed by a batch normalization layer which takes care of normalizing each output from each hidden network and a ReLU layer that performs regression. If the output does not match the targeted value then the network performs backpropagation to calculate the error gradient (loss). The error itself is calculated using the RMSE that will used by the ADAM optimizer. This process is repeated until we get values that are the same as or closely match the targeted values. Perceptrons are represented as neurons. Architectures tested in this research are as follows: fully connected layer input, batch normalization layer, then a ReLU layer.

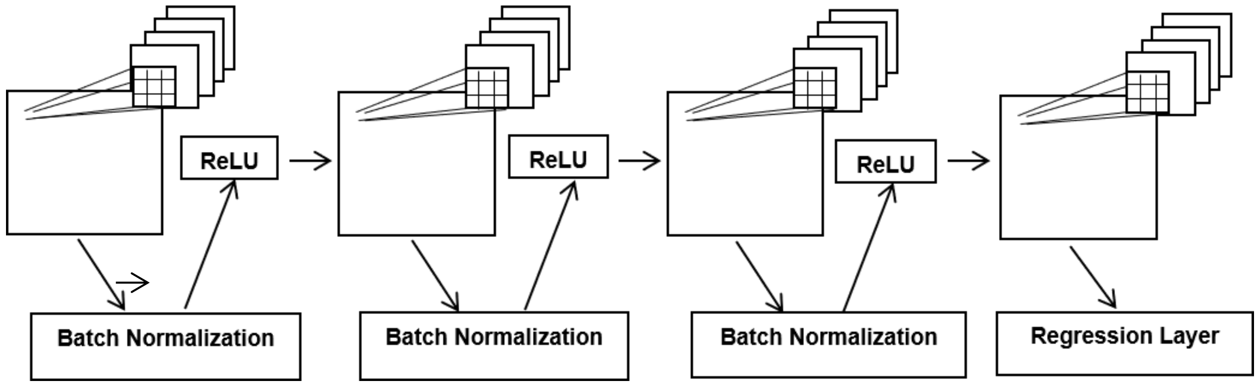

Figure 2.3: Convolutional neural network architecture

Convolutional neural networks used for this work have an architecture with an input layer that performs normalization on the dataset. The rest of the neural network consists of purely 2D convolutional layers. Each of these layers applies a horizontally and vertically sliding filter that performs convolutions on the inputs. The purpose of these filters is to compute the dot product of the weights and the input and adding the bias term. Convolutional neural networks can vary in their architecture and typically contain max-pooling layers and up-sampling. In a similar fashion to its fully connected counterpart, each convolutional layer is also followed by a batch normalization layer and a ReLU layer.

Chapter 3: Methodology

3.1 Approach

The approach followed for this work combines signal processing methods with different neural network architectures to achieve denoised and de-reverberated speech that is then compared to its anechoic version to assess benefits of processing. The first stage of the approach uses datasets containing clean, i.e., anechoic speech recordings of speech utterances and convolves them with a collection of different room impulse responses (RIR) to simulate reverb. Noise speech signals are also added to the clean speech to simulate noisy environments. These datasets are used for training and testing speech enhancement methods being evaluated. MATLAB Version 2019b [7] apps and toolboxes were used for various signal processing computations such as generating speech spectrograms for neural network inputs. This version was necessary to take advantage of tools for handling deep neural networks.

Utilized building blocks include code for denoising speech from the Mozilla dataset, which is a public dataset created with contributions from people who donate a recording of their voices dictating sentences. Although this dataset is extensive, it was not completely adequate to train a network with clean signals and simulated reverberant signals, as a great amount of recordings in the dataset already contain a variety of audibly reverberant surroundings. While many research articles use the TIMIT dataset [9] for training neural networks to suppress reverberation, access to this dataset requires a membership costing $200. To accomplish the same goal without incurring on that cost, the Edinburgh dataset [10] with anechoic recordings of people dictating sentences was utilized for this work. To create a dataset with reverberant versions of the Edinburgh dataset, the AIR dataset [11], which contains a collection of impulse responses of different environments ranging from classrooms to stair wells and concert halls was utilized. Creating the reverberation dataset consisted in convolving the anechoic signals with different impulse response signals. Other ways to simulate reverberation include using the reverb function provided in MATLAB [7], but the choice of impulse response to use provided more control over the simulation. First, a set of clean and reverberant signals were used to train a neural network without adding noise to observe dereverberation performance with an architecture meant for denoising. Two neural networks were used: one with fully connected layers only and another one with a mix of convolutional and fully connected layers.

3.2 Fully Connected Network

The fully connected network is available as a function in MATLAB [7] where certain parameters can be specified and changed systematically. The FCCN used in this code has the following parameters: the input layer consists of fully connected layers which automatically determine the input size during training and the output size is specified as a parameter of the number of neurons in the layer. This is done using the imageInputLayer function which inputs 2-D images into a network and applies data normalization [7]. The normalization is zero centered, which means it subtracts the mean calculated during training. The default setting is 'auto' – If the training option is false and you specify any of the normalization statistics (mean, standard deviation, min, or max), then normalize over the dimensions matching the statistics. Otherwise, recalculate the statistics at training time and apply channel-wise normalization [7]. The standard deviation, the minimum and maximum value for rescaling are calculated during training.

There are other parameters to consider when using this architecture, such as the function to initialize weights (default is the Glorot function which independently samples from the uniform distribution with zero mean and variance → 2/(inputSize + outputSize). The function to initialize bias is set to zeros, the layer weights are specified as a matrix and are learnable parameters. In this case the trainNetwork function uses the weightInitalizer function mentioned earlier and it has an outputSize by inputSize matrix. Layer biases are also learnable parameters and use the specifications in the bias initializer function. The learning rate factor for the weights is set to 1.0, which is multiplied by the global learning rate specified in the code as “initialLearnRate” which is 1.0e-5. The bias learn rate factor is also multiplied by the global learning rate for determining the learning rate for the biases in a layer. This code uses L2 regularization, which is also referred to as “Ridge regression,” to help prevent overfitting. This factor is set to 1.0. Ridge regression adds a square magnitude of the coefficient as a penalty term to the loss function [13] Regularization is used to avoid overfitting by penalizing high valued regression coefficients. [14] It also reduces parameters and simplifies the model. There is an L2 regularization factor for the biases that defaults to 0.0. This layer only accepts one input and one output by default. The training options can be customized and include the following parameters: MaxEpochs is set to 3 epochs; thus, the network does 3 passes through the training data [8]. Training sequences are shuffled at the beginning of every epoch. The minibatch size is set to 128 in order to evaluate our 128 training signals at the same time. The validation function evaluates the predictors and the targets. The validation frequency is set to calculate the mean square error every epoch. This code used the adaptive moment estimation solver (ADAM).

The fully connected network is trained using the parameters specified in the training options and architecture earlier in the code. The function trainNetwork(X, Y, Layers, Options) uses these parameters to train the network as follows: X is a 2D image data input with dimensions h x w x c x N which correspond to the height, width, # of channels of the images, and number of images, respectively; Y is the responses; Layers is an input argument that returns a series network with all layers connected sequentially; and Options is an input argument that takes all the specifications from the training options portion of the code.

The code counts the number of weights [7], biases the weight learn rate factor which uses the Glorot function as an input layer that takes the size of the 129x8 feature vector and is followed by two hidden fully connected layers containing 1024 neurons. Each hidden layer is followed by a ReLU layer and a batch normalization layer, which normalizes each input channel across a minibatch [7] This makes training faster and more robust. This layer takes the activations from each channel and normalizes them by subtracting the minibatch mean and dividing them by the minibatch standard deviation [7]. The input is then shifted by the layer by BETA(learnable offset) and the learnable scale factor. The batch normalization layer is set to the default parameters. The ReLU layer follows the batch normalization layer and first, it applies a threshold to each input element setting any value less than zero set to zero as follows:

$$f(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases}$$

and second, it calculates the learn rate with a piecewise schedule to decrease the learning rate by 0.9 after every epoch.

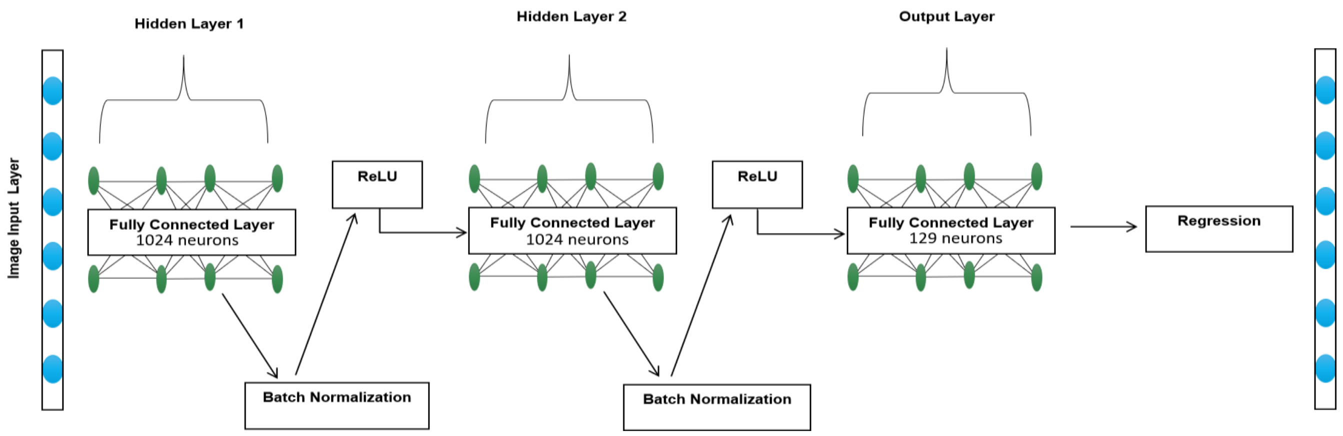

Figure 3.1: Implementation of fully connected neural network (FCNN)

3.3 Convolutional Neural Network

The convolutional neural network (CNN) acts as a filter to the input by applying convolution and addition of the bias term. The filters are only applied in the frequency domain. Filter width for the time domain is eight for the first layer and one for the rest of the layers. There are seven convolutional layers in total, six of which are followed by a batch normalization layer, then a ReLU layer and one is followed by a regression layer. For CNNT processing, the input is taken as an image input layer specified by the parameters numFeatures and numSegments. The 2D convolution layer function takes the following parameters: filter size, number of filters, and name-value pair arguments for specifying zero padding for the layer input and the stride filters. The stride format is [a b] where a is how many units it will be traversing in the vertical direction and b for the horizontal direction. This network was used with the default padding, which has a size calculated at training or prediction time to adjust the output size to the input size when the stride equals 1. For stride larger than 1, the output size is set to the ceiling of inputSize/stride, where inputSize is the input height or the width and the stride is in the corresponding dimension. The same amount of padding is added to the top, bottom, left, and right if possible. When the vertical padding is odd, extra padding is added to the bottom. Also, for odd horizontal padding, extra padding is added to the right. Filters applied to the input image will generate feature maps.

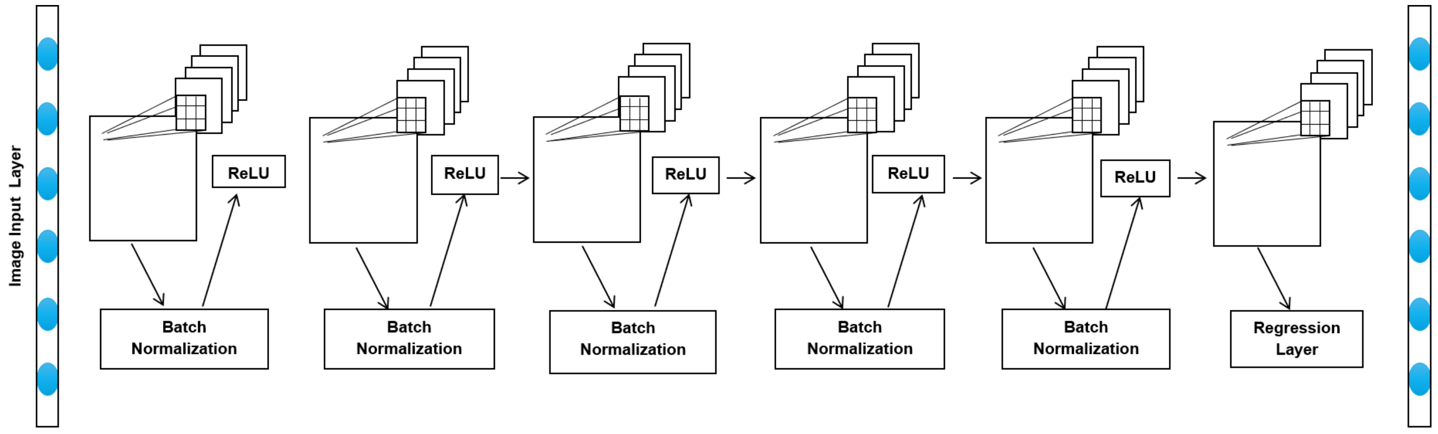

Figure 3.2 Implementation of convolutional neural network (CNN)

3.4 Testing Method

To train this neural network, we load in the datastore containing the validation speech files, shuffle, and follow the algorithm shown in Figure 6. The predict function is used to dereverberate (and potentially denoise) the signals. predict(Mdl, X) returns a vector of predicted class labels based on the trained discriminant analysis classification model [7]. Mdl is the classification discriminant model and X is predictor data to be classified. Root-mean-squared-error (RMSE) was used for both networks.

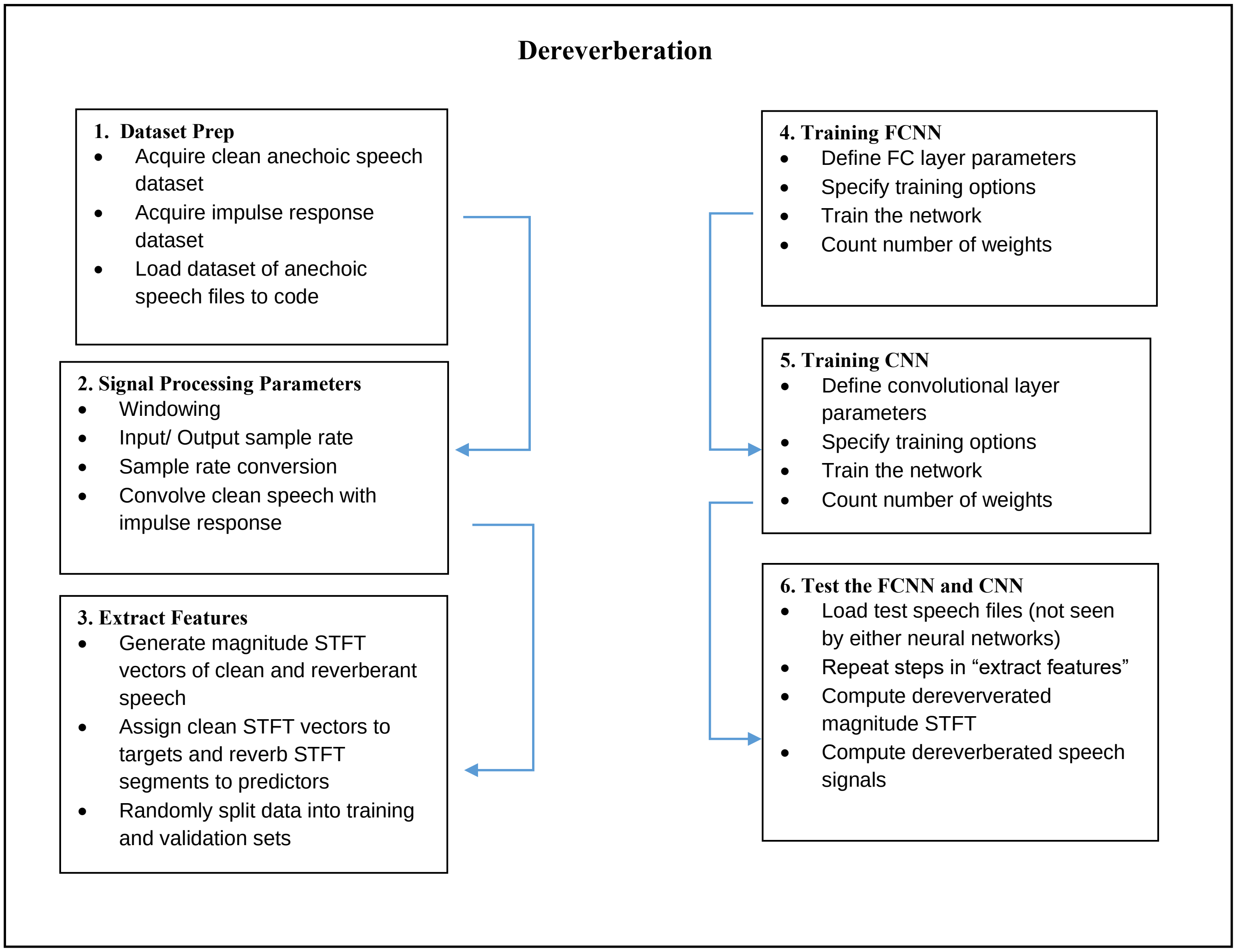
Dereverberation
1. Dataset Prep
Acquire clean anechoic speech dataset
Acquire impulse response dataset
Load dataset of anechoic speech files to code
2. Signal Processing Parameters
Windowing
Input/ Output sample rate
Sample rate conversion
Convolve clean speech with impulse response
3. Extract Features
Generate magnitude STFT vectors of clean and reverberant speech
Assign clean STFT vectors to targets and reverb STFT segments to predictors
Randomly split data into training and validation sets
4. Training FCNN
Define FC layer parameters
Specify training options
Train the network
Count number of weights
5. Training CNN
Define convolutional layer parameters
Specify training options
Train the network
Count number of weights
6. Test the FCNN and CNN
Load test speech files (not seen by either neural networks)
Repeat steps in "extract features"
Compute dereververated magnitude STFT
Compute dereverberated speech signals

Figure 3.3 Implementation of convolutional neural network (CNN)

Chapter 4: Experiments

The datasets were prepared for training the neural networks. The Edinburgh dataset [10] contained sets of speech files meant for training and testing speech enhancement methods. The set consists of speech uttered by 56 English-speaking speakers with a variety of accents. There are about 400 sentences spoken by each speaker and all dictations were recorded with a 96kHz sampling frequency with 24 bits, which was converted to 16 bits and downsampled to 48kHz. These were recorded from an omnidirectional microphone in a hemi-anechoic chamber of the University of Edinburgh. This anechoic clean speech dataset was augmented to include reverberation in various versions. Each of the clean speech recordings were convolved with the Aachen Impulse Response (AIR) database to produce the reverberated versions of the clean speech. The AIR database is a collection of room impulse responses derived from lecture rooms, stairwells, concert halls, etc., that can be easily imported to MATLAB [7]. Once the clean and reverberated signals are produced, they are assigned as target and predictor inputs respectively. The combined total of clean speech and reverberated speech is 1000 speech signals which are later used to train the neural networks. During training, the FCNN and CNN use the target input to adjust learning.

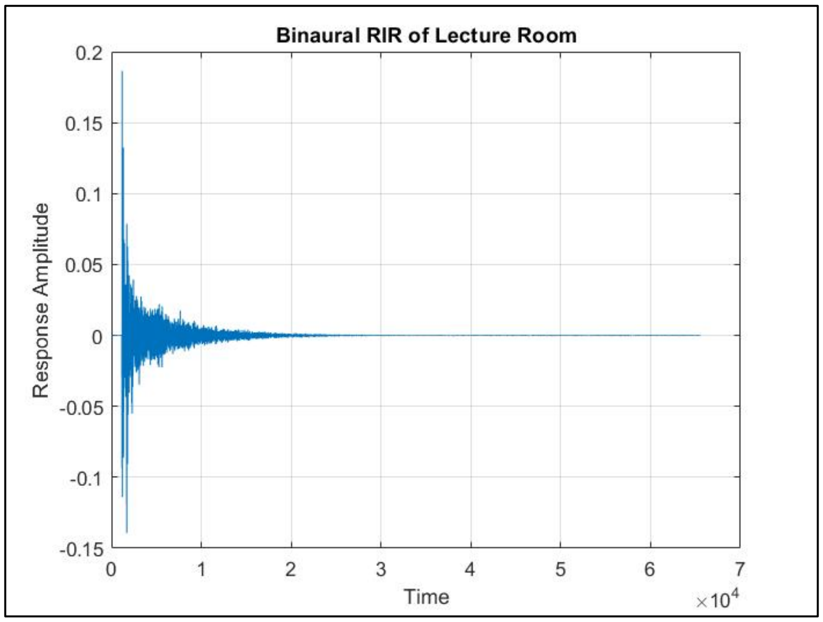

Figure 4.1 Room Impulse Response

Chapter 5: Results

The results of the FCNN are shown in Figure 8 and Table 1. The RMSE shows the standard deviation of the prediction errors and the loss shows how the prediction performed in estimating the expected outcome. The time it took to take all of the speech files to apply the sample rate conversion, convolve them with a room impulse response, and assign them to targets and predictors was completed in 2 minutes 33 seconds. Estimation of the gradient and update of network parameters were performed in each iteration. The training progress smooths out the training accuracy with a smoothing algorithm to make it easier to spot trends. The training loss depicts the loss on each minibatch and its smoothed version and the loss on the validation set [7]. The number of weights in the fully connected network totaled 2237440.

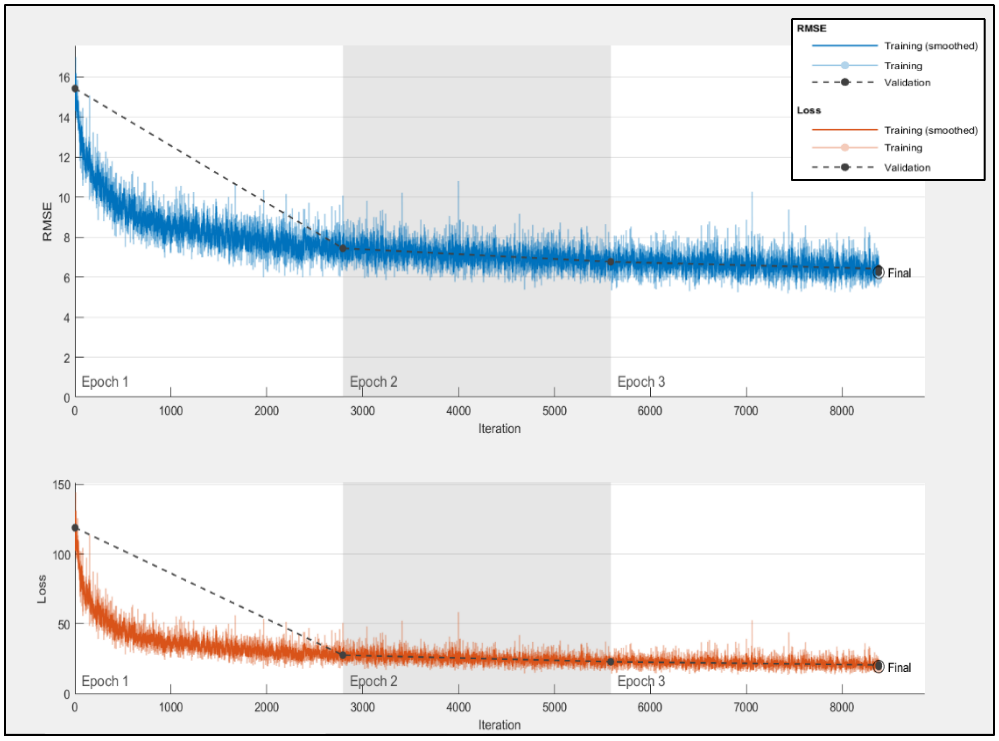

Figure 5.1 Training progress of fully connected neural network

Table 1.1: FCNN

Results	
Validation RMSE	6.2355
Training Time	
Elapsed Time	6min 26sec
Training Cycle	
Iterations per epoch	2793
Maximum iterations	8379
Validation	
Frequency	2793 iterations
Other Information	
Hardware Resource	Single CPU
Learning rate schedule	Piecewise
Learning rate	8.1e-06
Weights	2237440
Epochs	3

The results of the CNN are shown in Figure 9 and Table 2. The time it took to take all of the speech files, apply the sample rate conversion, convolve with a room impulse response, and assign them to targets and predictors was completed in 2 minutes 33 seconds. Like in the FCNN case, estimation of the gradient and update of network parameters were performed in each iteration.

Table 1.2: CNN

Results	
Validation RMSE	7.4374
Training Time	
Elapsed Time	23min 5sec
Training Cycle	
Iterations per epoch	2793
Maximum iterations	8379
Validation	
Frequency	2793 iterations
Other Information	
Hardware Resource	Single CPU
Learning rate schedule	Piecewise
Learning rate	8.1e-06
Weights	31812
Epochs	3

Again, the training progress smooths out the training accuracy with a smoothing algorithm to make it easier to spot trends. The training loss depicts the loss on each minibatch and its

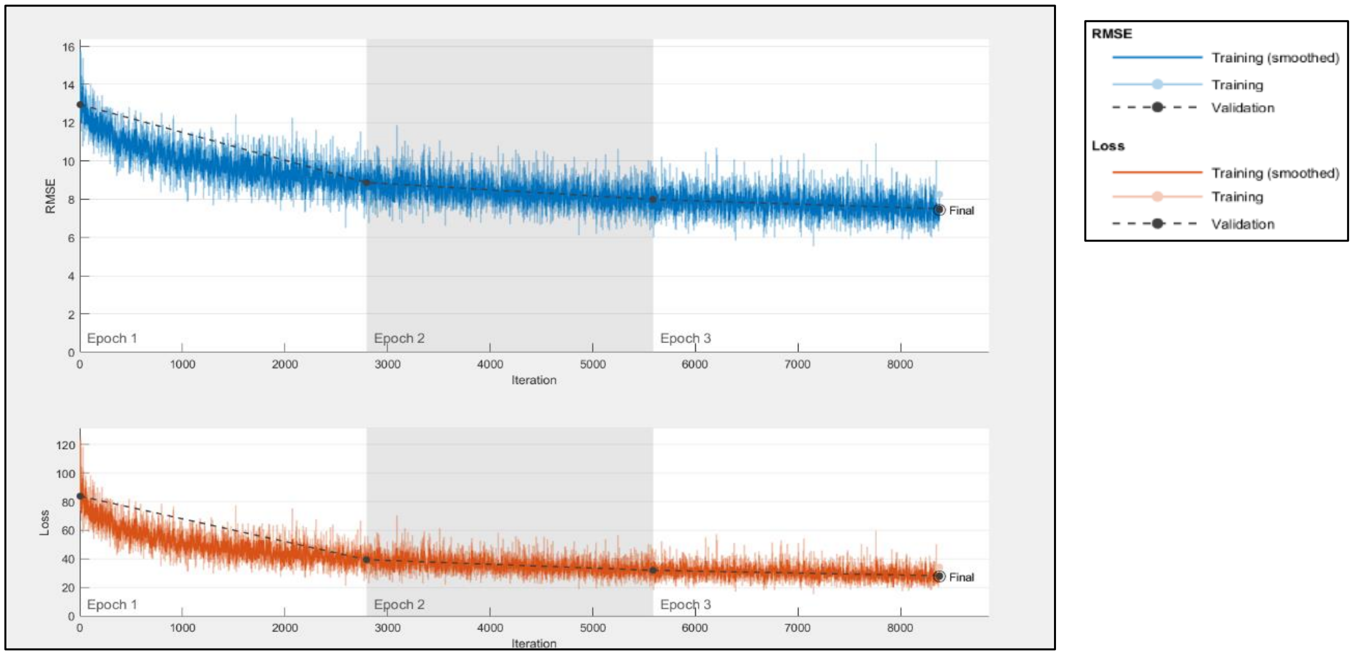

Figure 5.2 Training progress of convolutional neural network

smoothed version and the loss on the validation set [7]. The number of weights in the convolutional network totaled to 31812. Figure 10 includes visualizations of the resulting performance between FCNN and CNN.

5.1 Conclusions

The experimental setup shows that overall, the CNN performed better in terms of intelligible quality and generating fewer weights than its FCNN counterpart although the CNN was more computationally intensive. These results hold true to the CNN's reputation for its superior image recognition and classification. In conclusion it is shown that intelligible speech can be best produced by the using the presented CNN.

5.2 Future Work

For future work we can make more experiments utilizing architectures with different combinations of layers to observe what will work best for dereverberation. Additionally, using more testing metrics such as the Frequency weighted segmental SNR (fwSNR), short time

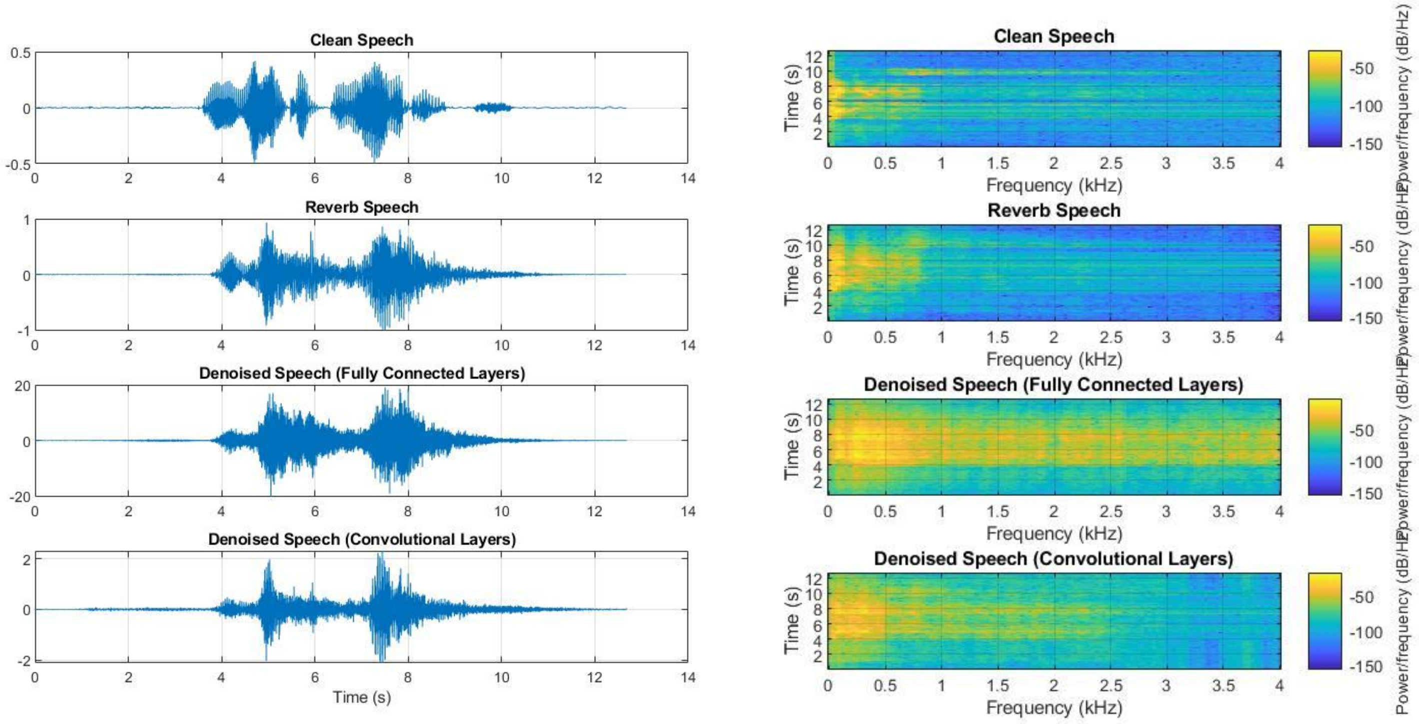

Figure 5.3 Graph and Spectrograms of clean, noisy (reverberated), denoised with FCNN and CNN

objective intelligibility (STOI), and perceptual evaluation of speech quality (PESQ) for scoring on speech intelligibility. More computational capabilities will also be required to allow for training with much larger datasets to achieve improved optimization and learning parameters.

References

[1] B. Wu, K. Li, M. Yang, and C.-H. Lee, “A Reverberation-Time-Aware Approach to Speech Dereverberation Based on Deep Neural Networks,” IEEE/ACM Trans. Audio Speech Lang. Process., vol. 25, no. 1, pp. 102–111, Jan. 2017.

[2] K. Han, Y. Wang, DeL. Wang, W. S. Woods, I. Merks, and Tao Zhang, “Learning Spectral Mapping for Speech Dereverberation and Denoising,” IEEE/ACM Trans. Audio Speech Lang. Process., vol. 23, no. 6, pp. 982–992, Jun. 2015.

[3] D. S. Williamson and D. Wang, “Time-Frequency Masking in the Complex Domain for Speech Dereverberation and Denoising,” IEEE/ACM Trans. Audio Speech Lang. Process., vol. 25, no. 7, pp. 1492–1501, Jul. 2017.

[4] Z. Zhang, J. Geiger, J. Pohjalainen, A. E.-D. Mousa, W. Jin, and B. Schuller, “Deep Learning for Environmentally Robust Speech Recognition: An Overview of Recent Developments,” arXiv:1705.10874 [cs], Sep. 2018.

[5] M. Ritter, M. Müller, S. Stüker, F. Metze, and A. Waibel, “Training Deep Neural Networks for Reverberation Robust Speech Recognition,” p. 5.

[6] P. Agrawal and S. Ganapathy, “Modulation Filter Learning Using Deep Variational Networks for Robust Speech Recognition,” IEEE J. Sel. Top. Signal Process., vol. 13, no. 2, pp. 244–253, May 2019.

[7] MATLAB ver. R2019a MathWorks, (2012). Signal Toolbox and Audio Processing Toolbox: User's Guide (R2019b). Retrieved December, 2018

[8] “Common Voice by Mozilla,” Common Voice. [Online]. Available: https://voice.mozilla.org/en/about. [Accessed: 02-Dec-2018].

[9] John S. Garofolo, et al., TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[10] C. Valentini-Botinhao. (2016). Reverberant speech database for training speech dereverberation algorithms and TTS models, 2016 [dataset]. University of Edinburgh. https://doi.org/10.7488/ds/1425.

[11] S. R. Park, J. W. Lee, "A Fully Convolutional Neural Network for Speech Enhancement", INTERSPEECH, 2017.
[12] M. Jeub (2019). AIR Database, (https://www.mathworks.com/matlabcentral/fileexchange/29073-air-database), MATLAB Central File Exchange. Retrieved December 6, 2019.

[13] A. Nagpal, “L1 and L2 Regularization Methods,” Medium, 14-Oct-2017. [Online]. Available: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c. [Accessed: 15-Jun-2019].

[14] A. Nagpal, “L1 and L2 Regularization Methods,” Medium, 14-Oct-2017. [Online]. Available: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c. [Accessed: 15-Jun-2019].

Vita

Jazmine Covarrubias is an IEEE, HKN, and RAS student member. She has received her Bachelor's in Electrical Engineering from The University of Texas at El Paso in spring 2018 and will receive her Master's in Computer Engineering in fall 2019. As an undergraduate and graduate student, her research and projects have explored areas in machine learning and embedded systems.

Contact Information : covarrubiasjaz05@gmail.com