Improving Keywords Spotting Performance in Noise with Augmented Dataset from Vocoded Speech and Speech Denoising

Ruohao Li

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2021

Committee:

Kaibao Nie, Chair

Sunwoong S Kim

Dong Si

Program Authorized to Offer Degree:

Electrical Engineering

University of Washington

Abstract

Improving Keywords Spotting Performance in Noise with Augmented Dataset from Vocoded Speech and Speech Denoising

Ruohao Li

Chair of the Supervisory Committee:

Kaibao Nie

Electrical Engineering

As more electronic devices have an on-device Keywords Spotting (KWS) system, producing and deploying trained models for keyword(s) detection is becoming more demanding. The dataset preparation process is one of the most challenging and tedious tasks in KWS. It requires a significant amount of time to obtain raw or segmented audio speeches. In this thesis, we first proposed a data augmentation strategy using a speech vocoder to generate vocoded speech at different numbers of channels artificially. Such a strategy can artificially increase the dataset size by at least two-fold, depending on the use case. With the new features introduced by the different number of channels of the vocoded speeches, a convolutional neural network (CNN) KWS system trained with the augmented dataset from vocoded speech showed promising improvement evaluated at +10 dB SNR noisy condition. The same results were confirmed in implementation on a microcontroller and proved using vocoded speech in data augmentation is the potential to improve KWS on microcontrollers. We further proposed a neural-network-based speech denoising system using the Weighted

Overlap-Add (WOLA) algorithm for feature extraction for more efficient processing. The proposed speech denoising system uses regression between a noisy speech and a clean speech and converts noisy speech (as input) into clean speech (as output). Thus, the input of the proposed KWS system will be relatively clean speech. Furthermore, by changing the training target to vocoded speech, such a speech denoising system can convert noisy speech (as input) into vocoded speech (as output). The combination of speech denoising and vocoded speech in data augmentation achieved relatively high accuracy when evaluated at +10 dB SNR noisy condition.

Abbreviations

Keywords Spotting …….. (KWS)

Convolutional neural network …….. (CNN)

Weighted Overlap-Add …….. (WOLA)

Hidden Markov Model …….. (HMM)

Deep Neural Network …….. (DNN)

Neural Network …….. (NN)

Continuous Interleaved Sampling …….. (CIS)

Denoising autoencoder …….. (DAE)

Deep denoising autoencoder …….. (DDAE)

Discrete Cosine Transform …….. (DCT)

Mel-frequency cepstral coefficients …….. (MFCC)

Band-pass filter …….. (BPF)

Low-pass filter …….. (LPF)

Root-Mean-Square …….. (RMS)

Signal-to-noise ratio …….. (SNR)

Digital signal processor …….. (DSP)

Table of Contents

List of Table

List of figures

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr. Kaibao Nie, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. I would also like to thank Dr. Dong Si and Dr. Sunwoong S Kim for your great help in providing suggestions during my research and feedback for the improvement of this thesis.

Chapter 1. Introduction

1.1 Problem Description and Motivation

With the rapid development of deep-learning-based KWS technologies, speech-interacted electronics are becoming increasingly popular, for example, Amazon Echo, Google Home, and Apple Home. The personal assistants behind those devices, such as Amazon's Alexa, Google Now, and Apple's Siri, are all utilizing KWS to interact with users. Such a KWS system aims to detect a pre-defined keyword or a set of keywords in a live stream of audio to enable a hands-free experience for users. Also, a KWS system can be potentially implemented in hearing devices such as hearing aids and cochlear implants to allow voice-activated adjustment of map parameters, e.g., volume and sensitivity settings. However, for practical problems, the creation of a dataset for the KWS system is long and tedious. In order to have high recognition performance with desired keyword(s), the use of large datasets is required since it helps in preventing overfitting in complex prediction methods. Moreover, in the case of recognizing keyword(s) in noisy environments, a fair number of the noisy keyword(s) should be included in the datasets to introduce different features for classification, which further increases the time when preparing datasets. The data augmentation technique has the potential to increase the size of the dataset artificially.

On the other hand, environmental noise is detrimental when using the KWS system in hearing devices. Users will be using the hearing device in various types of environments. Noisy voice as input to the KWS system will introduce unseen features that may result in a low detection rate.

Figure 1. The overall system proposed in this thesis.

As shown in Figure 1, in this thesis, we first created a speech vocoder to generate vocoded speech for dataset augmentation with adjustable spectral resolutions (different number of channels) that will give different levels of distortion. The smaller number of channels, the more distortion will be introduced in vocoded speeches. We built the KWS system by adapting a CNN from Zhang et al. [33], and trained it with Google's Speech Commands dataset [2] with selected 12 classes (10 keywords + background noise + unknown), and verified that it has high accuracy when classifying those 12 classes (best performance is ~95%). Then, we used the trained CNN to classify vocoded speeches at different numbers of channels varying from 1 – 32. The goal was to investigate how a trained KWS system can recognize vocoded speech. Data from human subjects [3-6] have shown that, and 4-8 channels are sufficient for the human to reach a high level of performance in recognizing sentences or phonemes with vocoded speech. Another experiment we performed is mixing vocoded speeches with original speeches from Google's Speech Commands dataset, creating an augmented dataset, and then trained it with the same CNN. We hypothesize that training with the augmented dataset can potentially improve its performance in classifying noisy keywords in noise (e.g., SNR = 10dB) since vocoded speeches introduce additional acoustic features in distorted speeches [37].

After creating the KWS system and the augmented dataset from vocoded speech, we built a neural-network-based speech denoising system using the WOLA for feature extraction to clean the noisy inputs to a KWS system as the second stage (the orange block in Figure 1). The core concept of the speech denoising system is to compute a regression between a noisy speech and a clean speech in the frequency domain, then convert noisy speech into clean speech using regression function.

Lastly, we implemented a similar KWS system in Arduino Nano 33 BLE Sense, including necessary preprocessing modifications and the trained CNN since the targeted hardware -- Arduino Nano 33 BLE Sense has limited computational power. The results showed that an augmented dataset from vocoded speech could be used in microcontroller implementation while maintaining similar performance.

1.2 Background

1.2.1 Keywords Spotting

A typical KWS system consists of a feature extractor and a classifier. As shown in Figure 2, the feature extractor will extract features from the input speech signal. Then the speech features will be fed into a classifier for classification and output probabilities for each output class. Back in the 1970s, traditional speech recognition technologies for KWS use HMMs as a classifier with limited vocabularies [20]. As computational power increased and costs decreased, more advanced KWS systems were developed, mostly based on HMMs. The 2000s saw the rise of NNs and were soon deployed in KWS. Early systems used HMMs alongside NNs, but in recent years the focus has shifted to primarily using NNs. Today, KWS platforms have mostly shifted from HMMs to NNs, using either DNNs or CNNs as a classifier.

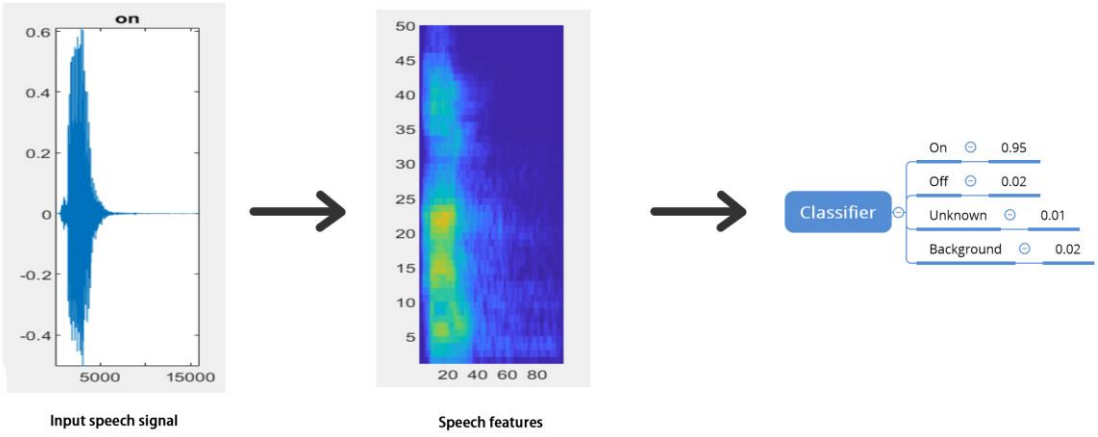

Figure 2: An example of a KWS system.

Chen et al. [1] reported a DNN based KWS system had 45% relative improvement in recognizing keywords and 39% relative improvement in the presence of babble noise with respect to a competitive HMM-based system. However, Sainath et al. [21] showed a trained CNN-based KWS system has a 27-44% relative improvement in false reject rate compared

to a DNN based KWS system. Such CNN not only performs better than a DNN but also has far fewer parameters. In this thesis, we used a CNN-based KWS system for all later chapters since CNNs have been proven to be better than DNNs in KWS.

In order to increase the KWS system's performance, a noise remover is often added to the KWS system to increase its compatibility when dealing with noise corrupted input speeches. The noise remover can be an additional part added to the KWS system. Huang et al. [22] presented a far-field KWS algorithm with a denoising frontend inspired by the cocktail party effect that can selectively focus the hearing on one particular conversation and resulting in a significant improvement in strong background noise. On the other hand, the noise remover can be a strategy of adding noise corrupted samples during the training stage. A common and easiest way is mixing white noise and clean speeches to artificially generate noisy speeches. For a more robust system, real noises should be used in order to cover as many real-world scenarios as possible. For example, dishwashing noises, traffic noises, and wind noises should be used when generating noisy speeches.

While previously limited to specific devices, today, KWS is available on most computer platforms, from home computers to smartphones. The most notable example is Apple's Siri, recognizing and answering questions from the user. Most major electronic brands have their own KWS systems, such as Apple's Siri and Microsoft's Cortana. KWS is still being used for customer service, voice-controlled applications and is a significant part of the "smart home" concept. Moreover, KWS can be used in microcontrollers. Zhang et al. [33] showed it is possible to use KWS in microcontrollers when satisfying memory and compute requirements without sacrificing accuracy. In this thesis, we tested a trained KWS in an Arduino Nano 33 BLE Sense and kept the same performance simulated in the desktop.

1.2.2 Data Augmentation

As mentioned previously, in order to have a noise-robust KWS system, lots of noisy speeches from various noisy environments should be included in the dataset. Pete Warden from Google Brain[2] described preparing a dataset of 105,829 utterances of 35 words as challenging and expensive in collecting and verifying audio samples. Therefore, personal users or student researchers may not have the ability to collect a large dataset for their own purposes in training a noise-robust KWS system. However, data augmentation may have the ability to solve the problem. It can artificially increase the amount of data in the dataset by adding newly created synthetic data from existing data through signal processing or modified copies of already existing data. The early work from Chang et al. [10] used speaker modification techniques [11] in a speaker-independent keyword spotting task by increasing the size of the training set. More recent works [12-15] also used data augmentation for Automatic KWS. These works mainly take advantage of special audio datasets transformations for speeches such as vocal tract length perturbation. Also, data augmentation has been used to increase the quantity of training data, avoid overfitting and increase the robustness of models [16-17], while Raju et al. [18] demonstrated improved performance in the living room with ambient noise from household appliances when mixing training datasets with music and TV/movie audio.

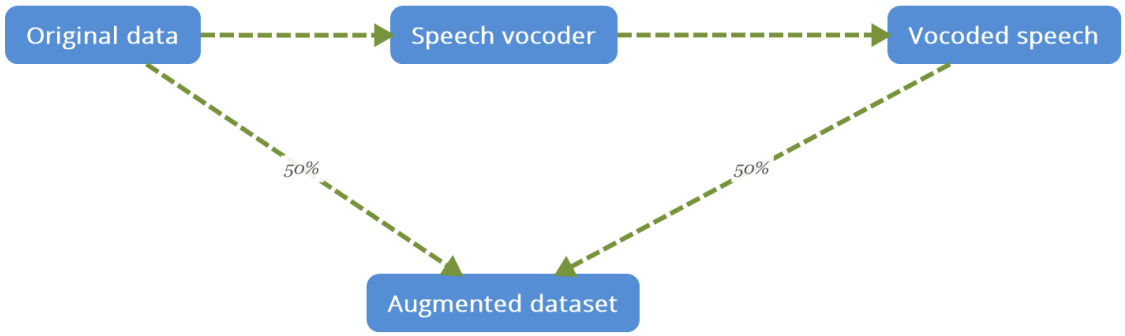

Figure 3: Process of generating augmented dataset from speech vocoder.

In this thesis, as shown in Figure 3, we proposed a data augmentation strategy by adding modified copies of already existing data using a proposed speech vocoder with the

different number of channels that will be introduced in the next section, the output of the speech vocoder (vocoded speeches) will be distorted speeches that are similar to noisy speeches. For all later chapters, we used a 1:1 ratio in data augmentation, which means there will be half modified copies from already existing data and half original data inside the augmented dataset. According to the ratio in data augmentation, the number of total samples in the augmented dataset will be doubled compared to the original dataset.

Since vocoded speeches are similar to noisy speeches, adding vocoded speeches into the dataset during training will potentially adding noise features and increasing noise robustness. Moreover, with adjustable spectral resolutions, vocoded speeches have the ability to modify the noise features depending on the number of channels used in the speech vocoder. Therefore, the data augmentation strategy using the proposed speech vocoder may increase the noise robustness of the KWS system in a simple and economical way.

1.2.3 Speech Vocoder

The speech vocoder we created in this thesis was inspired by CIS signal processing strategies found in cochlear implants [36], and we will discuss it in detail in Chapter 2. The first vocoder was developed in the 1920s, showing that an electromechanical device with relatively few controllable parameters could produce intelligible speech [7]. Decades ago, speech vocoder was widely used in music and filmmaking. Artists and musicians used a vocoder to artificially synthesize desired voice or sound effects in particular needs [8]. Recently, Manu et al. [9] demonstrated the vocoder-based modifications for speech data augmentation for NNs when performing the estimation of the noise-robust fundamental frequency (F0), showing the best-performing proposed method greatly outperformed the compared F0 estimation methods in terms of noise robustness.

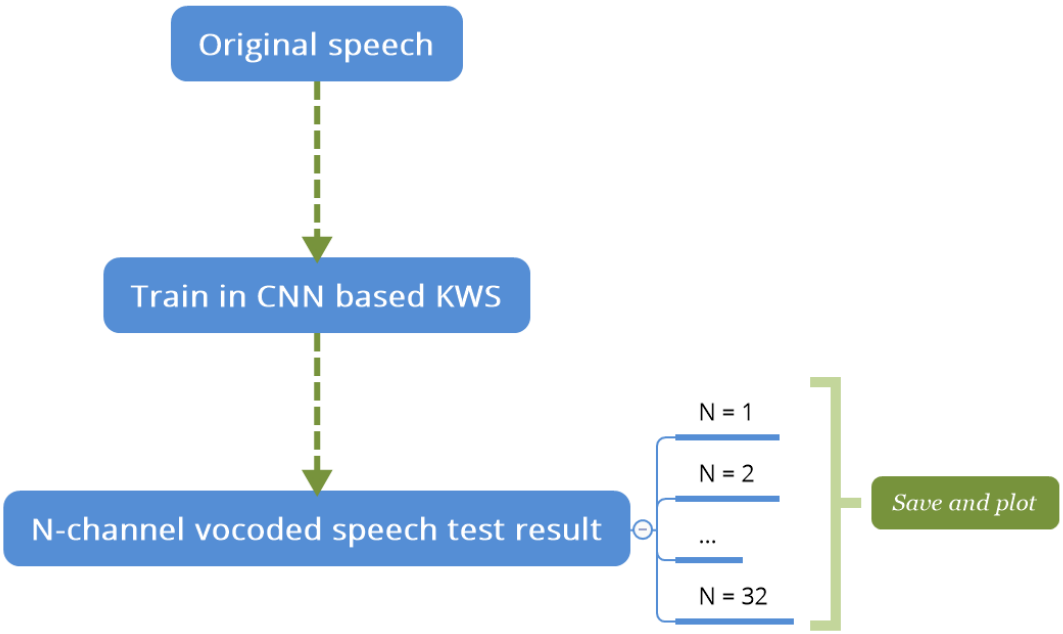

Figure 4: Test flowchart for KWS trained with vocoded speeches from channels 1-32 with clean speeches.

When hearing the sound from the CIS simulation, we found that it sounded strange and sometimes sounded like noise in some cases. Then, we created a speech vocoder inspired by it in MATLAB and discovered that it would create different levels of distortions depending on the different number of channels. The smaller number of channel(s), the more distortions will be introduced. For example, 1-channel vocoded speeches will sound like background noises, 32-channel vocoded speeches will sound similar to clean speeches, and 64-channel vocoded speeches will sound like clean speeches. We hypothesis vocoded speeches may be useful in the KWS system and prepared an experiment shown in Figure 4., we tested KWS systems trained with clean speeches and tested with N-channel vocoded speeches. Since 32-channel vocoded speeches are similar to clean speeches but with distortions introduced, a test when classifying 32-channel vocoded speeches will potentially have a better recognition rate; a test when classifying 1-channel vocoded speeches will have the worst recognition rate.

On the other hand, shown in Figure 5. we created vocoded speeches from channel 1-32 using the proposed speech vocoder and mixing vocoded speeches with an original dataset with the ratio of 1:1 to form 32 augmented datasets in total for training KWS systems. The trained KWS systems will be tested using SNR = 10 dB noisy speeches. Since vocoded speeches are distorted speeches (noisy features introduced), the trained KWS systems will be able to handle noisy speeches in some way and perform better compared to the KWS system trained without using vocoded speeches.

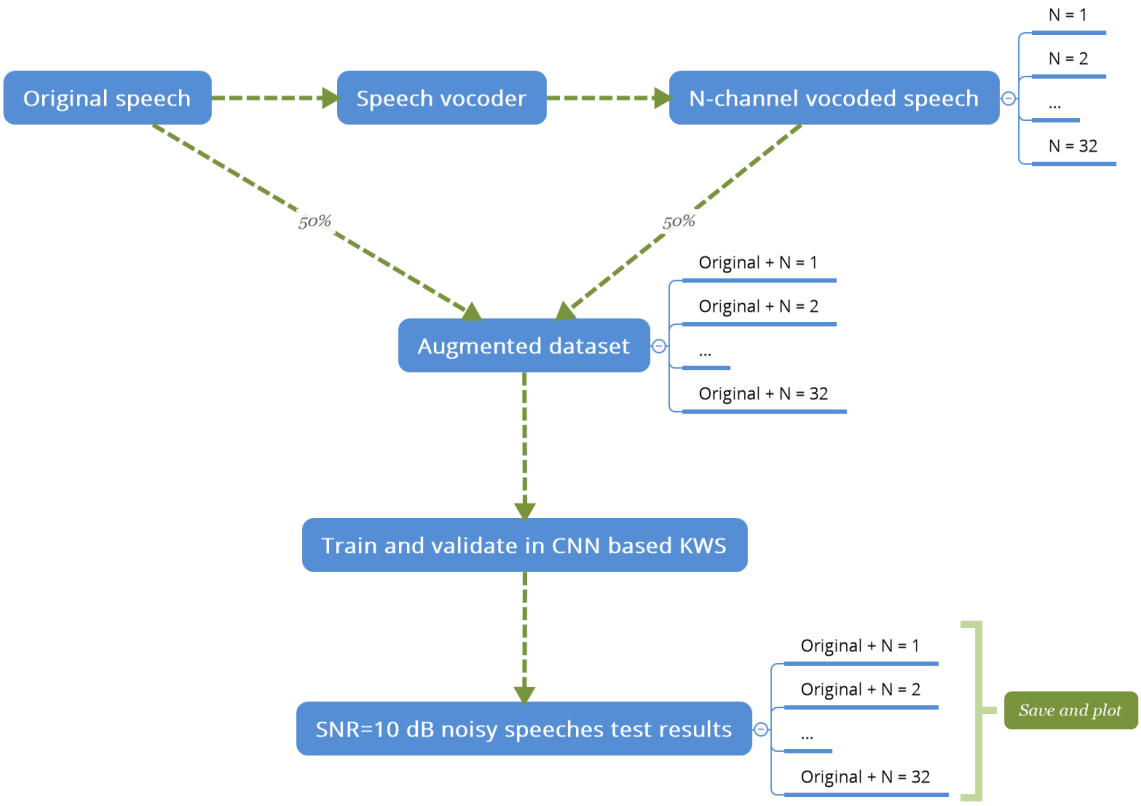

Figure 5: Test flowchart for KWS trained with the vocoded augmented dataset from channel 1-32 with SNR = 10dB noisy speeches.

1.2.4 Speech Denoising

As mentioned in previous section 1.2.1, the noise remover in the KWS system can be an additional part added to it. Zeghidour et al. [23] introduced Wavesplit as an end-to-end source (speech) separation system that can separate clean speech and background noise from input speech. Wavesplit can also be used in other domains, such as separating fetal and maternal heart rates from a single abdominal electrocardiogram. Wang et al. [24] provided an overview of deep learning-based supervised speech separation algorithms, such as speech enhancement, speaker separation, speech dereverberation, and multi-microphone techniques. Lu et al. [25] showed a denoising process in learning the DAE and proposed a new DDAE that outperformed with a minimum mean square error-based speech enhancement algorithm. The DDAE can also be used in hearing devices to improve Intelligibility. Lai et al. [26] showed a DDAE-based noise reduction could potentially be integrated into a cochlear implant (CI) speech processor to help CI users reduce incoming noise under noisy conditions.

In this thesis, we adapted speech denoising algorithms from [27, 28], using WOLA feature extraction to build two separate speech denoising systems, a DNN based speech denoising system and a CNN speech denoising system before the KWS system, just like shown in Figure 1. By computing a regression between noisy speeches and clean speeches in the frequency domain, the trained DNN or CNN can convert noisy speech into clean speeches in the frequency domain and then apply phases after getting clean speeches in the frequency domain. Figure 6. shows how noisy speech can be converted into clean speech or vocoded speech. During the training stage, if the target is set to clean speech (using clean speeches during training), the DNN or CNN will compute the regression that has the ability to convert noisy speech into clean speech. Same for vocoded speech, if the target is set to N-channel vocoded speech (using N-channel vocoded speeches during training), the DNN or CNN will compute the regression that has the ability to convert noisy speeches into N-channel vocoded speeches.

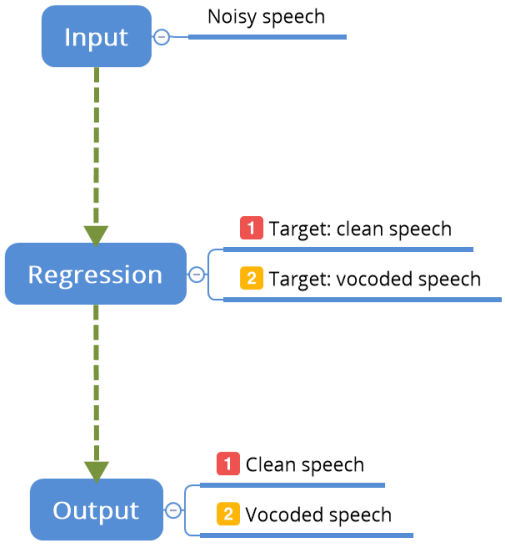

Figure 6: Flowchart of noisy speech into target speech.

1.2.5 Feature Extraction

In this thesis, we tested two feature extractors -- MFCC and WOLA in different scenarios. MFCC was first used in the KWS system due to its popularity in deep learning-based speech recognition projects. However, WOLA was tested in a later stage for system optimization since WOLA is efficient for analyzing continuous audio signals, while Ahadi et al. [29] showed using WOLA filter bank for feature extraction in frontend processing for their KWS system. WOLA processing is also readily available in hearing devices and mobile devices. Their result showed that WOLA has equally or slightly better performance than other well-known feature extraction techniques like MFCC, especially in lower SNR.

1.2.5.1 Mel Frequency Cepstral Coefficients (MFCC)

MFCC values describe the Mel-frequency cepstrum, represent the time-based power spectrum of a signal. MFCC values are calculated by taking the logarithm of the Mel-weighted power data to better resemble the signals perceived by the human ear and then performing a DCT to reduce the impact of low-energy components. MFCC is commonly

used human-engineered speech features in deep learning-based speech recognition that are adapted from traditional speech processing techniques [30]. It is able to produce robust features in noisy signals and has good error reduction [31]. When lots of projects and researchers using MFCC, the MFCC's built-in function has been implemented in lots of platforms that can be used easily.

1.2.5.2 Weighted Overlap-Add algorithm

WOLA is widely used in analyzing audio signals. It has two main parts: WOLA analysis and WOLA synthesis. WOLA analysis uses an overlapping window to transform a time-domain signal into input blocks. And then, those time-domain input blocks are further transformed to give a short-time Fourier spectrum. In short, its main function is to transform a finite number of overlapped blocks of the time-domain signal into the frequency domain. WOLA synthesis is the reverse of WOLA analysis. It applies inverse Fourier transformation to the segments of the short-time Fourier spectrum and sums the results with overlaps to obtain an audio signal in the time domain. Its main characteristic is to transform frequency spectrum data back to a finite length of the time-domain signal. More details of WOLA analysis and synthesis will be discussed in Chapter 3. WOLA analyses the time domain signal block by block, making it an ideal method for real-time processing devices. Compared to the FFT method, WOLA is more suitable for implementation on a microcontroller with fewer distortions created between frames, especially for low-power devices. The low delay goal is achieved with the help of WOLA's block processing feature, which doesn't involve long FFT or IFFT operations. Besides the great accuracy, WOLA also provides additional benefits such as efficient implementation, easy application of enhancement algorithms, and perfect or near-perfect reconstruction property.

1.2.6 Implementation on Microcontroller

Since the use of vocoded speeches in data augmentation can potentially increase the KWS system's performance in noisy environments. We hypothesis such a benefit can be expanded into real-world scenarios through implementation on a microcontroller into a microcontroller – e.g., Arduino Nano 33 BLE Sense (Clock 64MHz, Flash 1MB, RAM 256KB). Since such a microcontroller has power and compute constraints, some modifications in reducing feature extractions and CNN architecture are made to satisfy hardware limitations, and more detail will be discussed in Chapter 5.

1.3 Thesis Structure

- This chapter, Chapter 1, introduces the problem, some background information in regards to the history and concepts of keyword spotting, data augmentation, speech denoising, and speech preprocessing.
- Chapter 2 covers the process of preprocessing, generating vocoded speeches in different channels, preparing different kinds of the dataset, and training with the proposed KWS system.
- Chapter 3 covers speech denoising by using both CNN and a fully connected NN.
- Chapter 4 covers how to combine the best KWS system trained proposed in Chapter 2 with the speech denoising system proposed in Chapter 3.
- Chapter 5 covers the processes of transferring the KWS system proposed in Chapter 2 into an Arduino Nano 33 BLE Sense, including necessary modification in MFCC and trained CNN.
- Chapter 6 presents the conclusions and discussions of this thesis.

Chapter 2. Data Augmentation with Vocoded Speeches

This chapter covers the process of preprocessing, generating vocoded speeches in different channels, preparing different kinds of the dataset, and training with the proposed KWS system.

2.1. Overview of the Proposed KWS System

The block diagram of the KWS system used in this thesis is shown in Figure 7. First, audio samples will be processed by a vocoder to generate vocoded speeches. We mixed audio samples and vocoded speeches with a ratio of 1:1 to form an augmented training set. Then, in the feature extraction module, 50 MFCC features were computed every 30 ms with a 10 ms frameshift at 512 FFT length. The extracted speech feature matrix was fed into a CNN, which is adapted from [33]. The classifier can generate the probabilities for the output classes for training or validation.

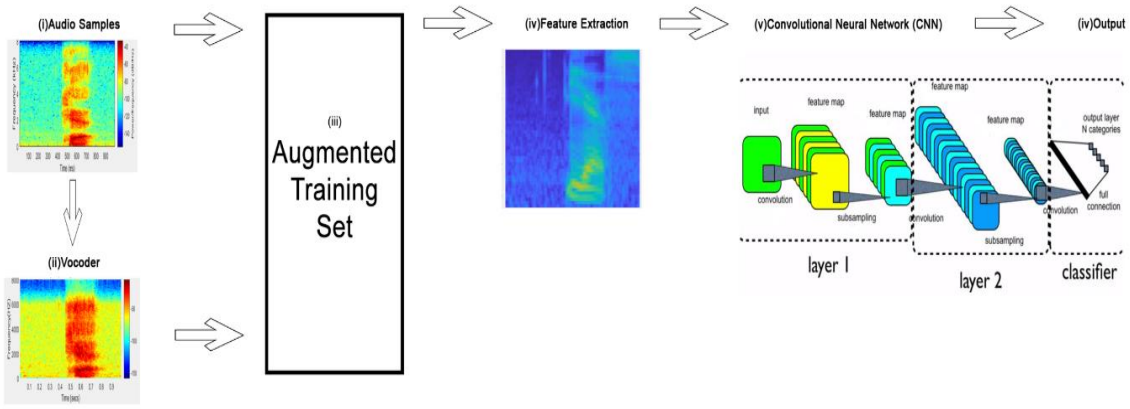

Figure 7. Block diagram of the KWS system. From left to right: (i) Input audio samples and the spectrogram of a keyword “cat.” (ii) the spectrogram of a vocoded keyword “cat.” (iii) The augmented training set contains both original audio samples and vocoded speeches with a ratio of 1:1, e.g., if the total number of input audio samples is 100, then the

augmented training set will have 200 samples in total, with 100 samples from the vocoded speeches. (iv) Feature extraction from the keyword “cat.” (v) CNN. (vi) Output probability for classification.

2.2 Model Implementation

2.2.1. Speech Vocoder

The Speech Vocoder we used in this thesis was inspired by the acoustic simulation algorithms for cochlear implants as described in [37], which are commonly used to simulate the sound perceived by cochlear implant subjects. The vocoder has five major steps to process speech signals, as shown in Figure 8. The first step is to use a number of BPFs (BPF) (N >= 1) to filter the input signal with logarithmically equal frequency bands. Since the frequency of human voice is primarily under 5 kHz, an LPF is commonly applied before BPF. In this thesis, the frequency range for vocoding is from 80 Hz to 6 kHz. The second step is to apply a full-wave rectifier in each channel after the BPF to extract temporal envelopes from each channel. The third step is to apply a 300Hz LPF in each channel after the full-wave rectifier to filter out temporal fine structures higher than the upper limit of envelope frequency. Two different modulations were used to synthesize vocoded sounds, a sine-wave modulation that produces robotic sound and a white noise modulation that produces more human-like sounds. The last step is to sum all signals from N channels after modulation. All vocoded sounds were further normalized by calculating their RMS, which is required for the generated vocoded speeches because the amplitude of the vocoded speeches typically dropped by around 60% compared to the input speech signal.

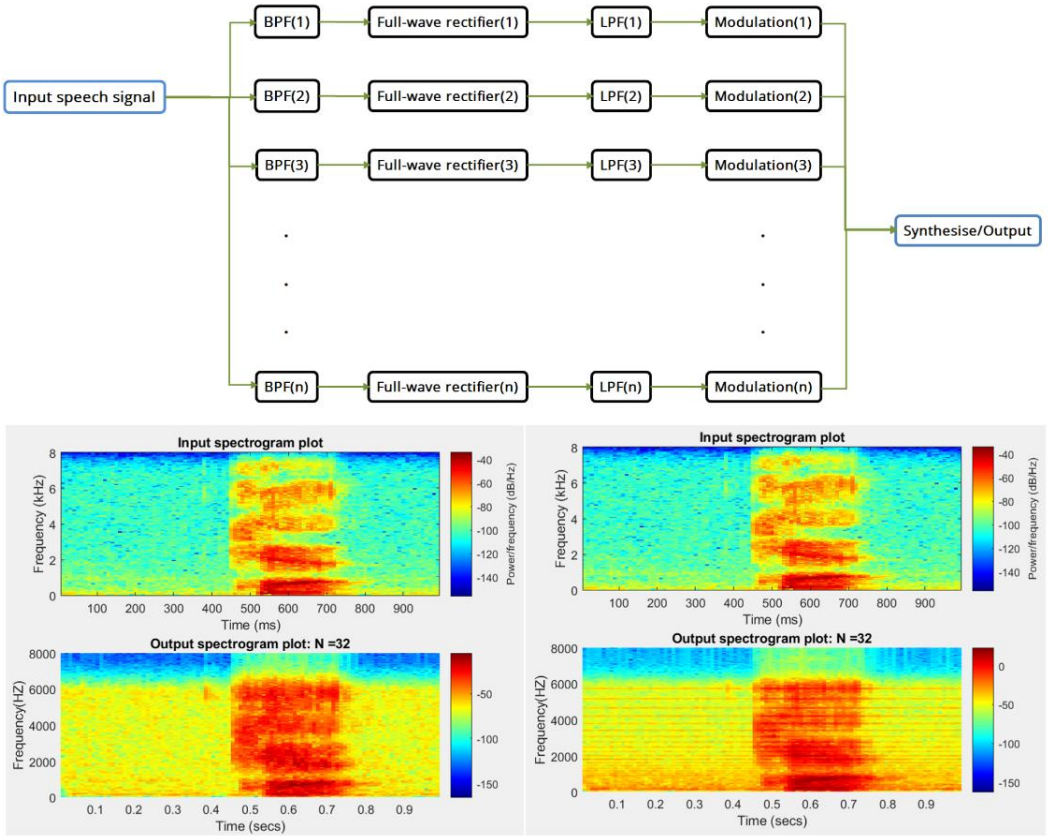

Figure 8. shows the flowchart of an n-channel Speech Vocoder. The spectrogram of “cat” with white noise modulation (left), the spectrogram of “cat” with sine modulation (right). The spectrogram of “cat” with white noise modulation has smoother transitions.

2.2.2. CNN Architecture

In Figure 9, we adapted the CNN architectures from [33] with increased convolutional layers to five to deal with larger feature sets. After feature extraction, the KWS system produces 4900 (50x98) feature matrixes for one-second of audio, while [33] only has 1960 (49×40) features. Each convolutional layer is followed by a batch normalization layer, a ReLU layer, and a max-pooling layer to reduce feature maps. By adding a final global max-pooling layer, the number of parameters required in the fully-connected layer will be significantly reduced. To prevent the network from overfitting, a 0.2 of the dropout was added to the input to the last fully-connected layer.

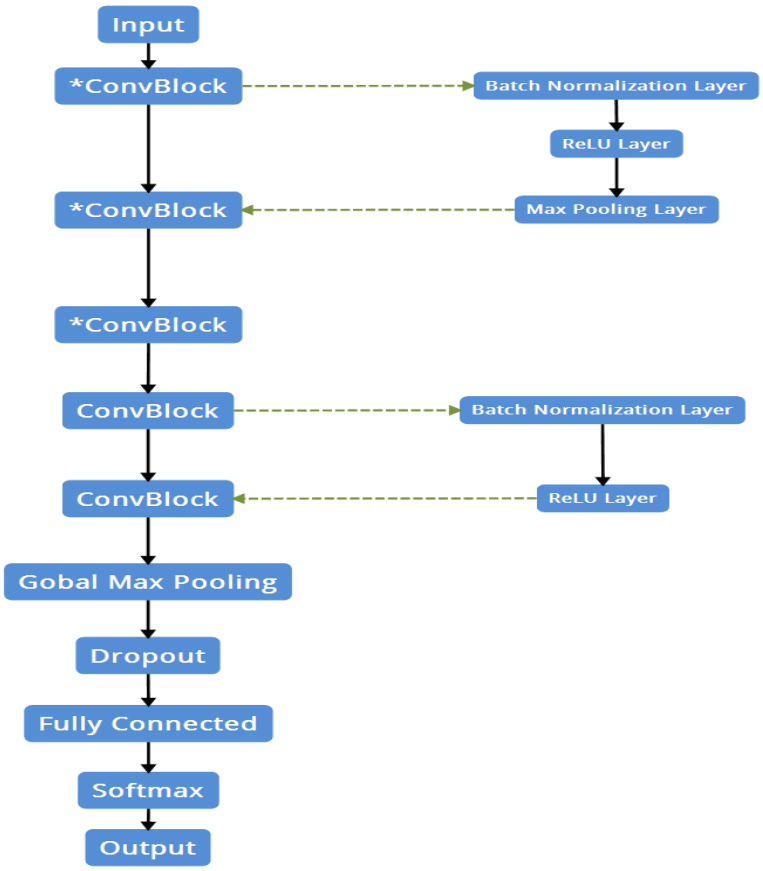

Figure 9. Overall, CNN architecture was used in this study. Each *ConvBlock is followed by a batch normalization layer, a ReLU layer, and a max-pooling layer. Each ConvBlock is followed by a batch normalization layer and a ReLU layer. Global max pooling is followed by the second ConvBlock.

The KWS system is trained for 40 epochs with the Adam optimizer [33] with a mini-batch size of 128 and an initial learning rate of 1e-4 to reduce the learning rate by a factor of 10 after 30 epochs; the model with the highest validation accuracy is saved to evaluate accuracy on the test dataset.

2.3 Data Augmentation with Vocoded Speeches

2.3.1. Training on the Speech Commands Dataset

We trained and evaluated our model using Google's Speech Commands dataset [2], an established dataset for benchmarking KWS systems. The dataset consists of 65k one-second-long speech audio of 30 different keywords spoken by 1881 different speakers. The CNN proposed in this paper is trained to classify the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with unknown and background noise. Background noise data is generated from the background noise category inside the dataset and segmented into one-second-long audios. Thus 4500 background noise data samples will be generated. The remaining 20 keywords from the dataset are labeled as "unknown." However, there are many more samples for the "unknown" that leads to an imbalance in the dataset. Therefore, we only take 15% of the "Unknown" from the remaining 20 keywords. The original dataset is split into training, validation, and test sets in the ratio of 8:1:1 while making sure that the audio clips from the same person stay in the same set.

2.3.2. Experimental Setup for Vocoded Speech Dataset

The Vocoded Speech Dataset is generated from the same Google's Speech Commands dataset [2] by using a speech vocoder with the same dataset split configuration as introduced previously. We generated six vocoded speech datasets at different numbers of channels: n = 1, 2, 4, 8, 16, 32. Since the characteristic of 2-channel vocoded speech is similar to the 1-channel vocoded speech, we dropped the 2-channel vocoded speech when evaluating the performance of vocoded speech with numbers of channels: n = 1, 4, 8, 16, 32 by training with the original dataset, and validating with vocoded speech with the proposed KWS. On the other hand, one and 2-channel vocoded speech are highly distorted sounds, so we dropped them when evaluating the performance of vocoded speech in noise with conditions at the numbers of channels: n = 4, 8, 16, 32 by training with 50% original

dataset and 50% vocoded speech with various channels, and validating with SNR = 10 dB noise corrupted original dataset in the proposed CNN.

2.3.3. Results

The base model of CNN using the original dataset in both training and validation achieved the desired accuracy of 94.38% on the Google's Speech Commands dataset. We also tested this result with k=6 cross-validation, and the result has nearly no change. Figure 10 shows the performance of vocoded speech tested with the trained CNN (orange curve) and the sentence recognition performance from human subjects (blue curve) listening to vocoded sounds from a previous study [5], as the number of channels for vocoding is increased. It is worth noticing that the performance of KWS for vocoded speech is in a similar trend to sentence understanding performance from human subjects. Better recognition performance is achieved when the number of channels increases. The number of channels required to achieve reasonable performance is around 8 for the trained KWS system. However, the performance of sentence understanding from human subjects does not tend to change much from 5-channel to 16-channel. The performance at N = 32 shows that 32-channels vocoded speech is very similar to the original dataset, which is 92.96%. 8-32 channels of vocoded speech can be considered as augmented samples for training KWS or automatic KWS systems. The dataset augmentation method may have the potential to significantly expand the training dataset by multiple times.

By comparing confusion matrixes in Figure 11 and Figure 12, it is noticeable that when classifying "unknown," the KWS trained with 32-channels vocoded speech has 25% less accurate than the KWS trained with 100% original dataset. Such drawback has a close relationship with the distortion introduced from vocoded speech. For example, shown in Figure 12, the KWS trained with 32-channels vocoded speech falsely predicted the keyword "no" as the keyword "unknown" 28 times. As shown in Figure 11, the KWS trained with 100% original dataset only falsely predicted the keyword "no" as the keyword "unknown" two times.

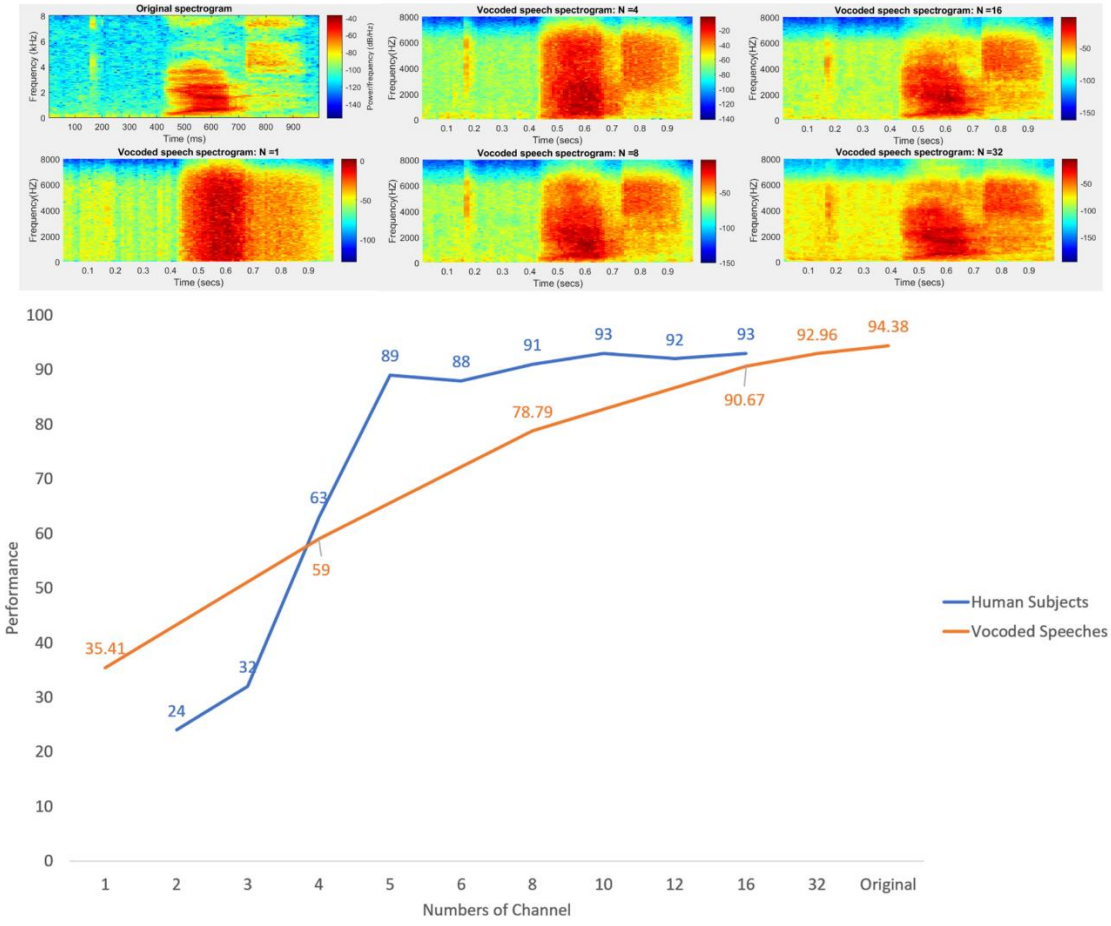

Figure 10. The top six spectrograms show the difference between original speech and various channels of vocoded speech from the keyword “yes.” The orange curve indicates the recognition performance (correct percentage) of the trained CNN as the number of channels is increased from 1 to 32 tested with the vocoded speeches. The CNN was trained with the Original Dataset and validated with the vocoded speeches. The “Original” in the “Numbers of channels” means trained with the Original Dataset and validated with the Original Dataset. The blue curve indicates the recognition performance of the human subjects with different channels from 2 to 16.

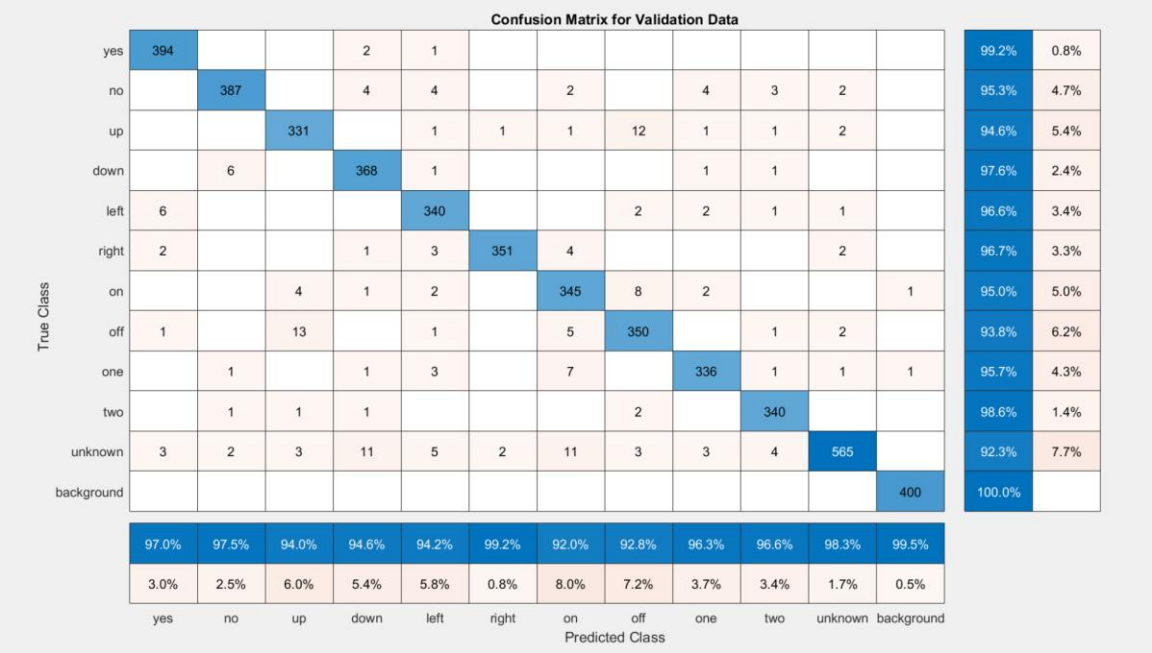

Figure 11. Confusion Matrix of trained KWS with 100% original dataset ("Original" in Figure 5), when classifying the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with unknown and background noise.

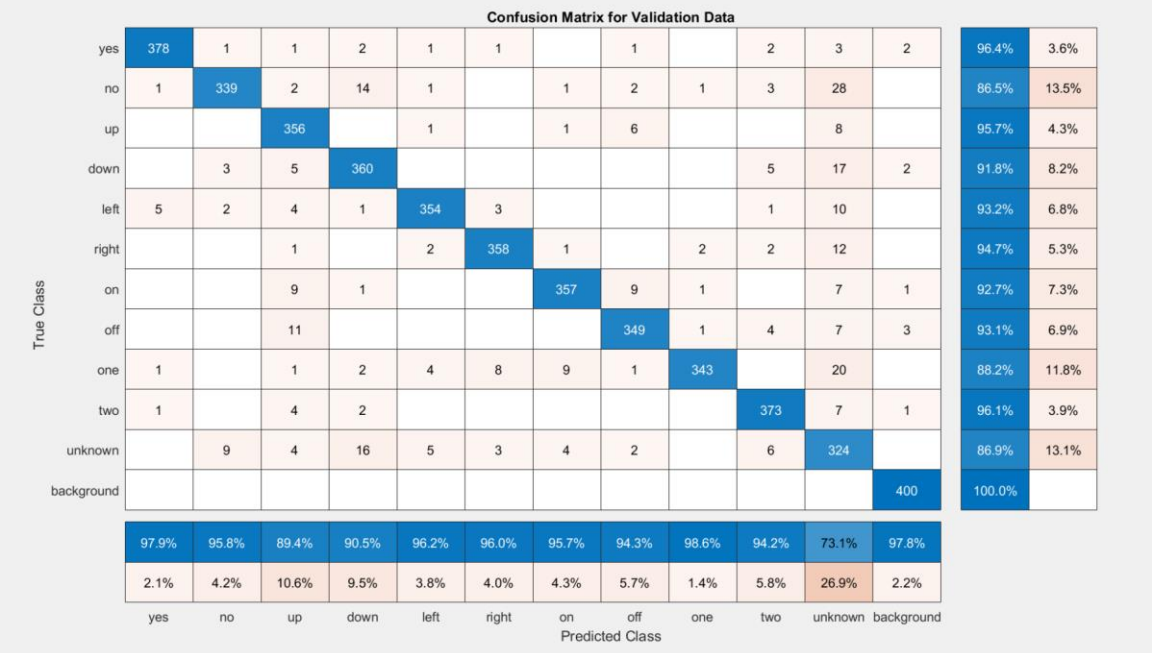

Figure 12. Confusion Matrix of trained KWS with 32-channels vocoded speech (“32” in Figure 5), when classifying the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with unknown and background noise.

Figure 13 summarizes the performance of vocoded speech in SNR = 10 dB noise corruption with numbers of channels: n = 4, 8, 16, 32 as compared to how the original dataset performed in the same condition. The original dataset has lower performance compared to vocoded speech with the number of channels at n = 4, 8, 16, and 32. And also, the performance of KWS with vocoded speech improves when the number of channels in the vocoder increases.

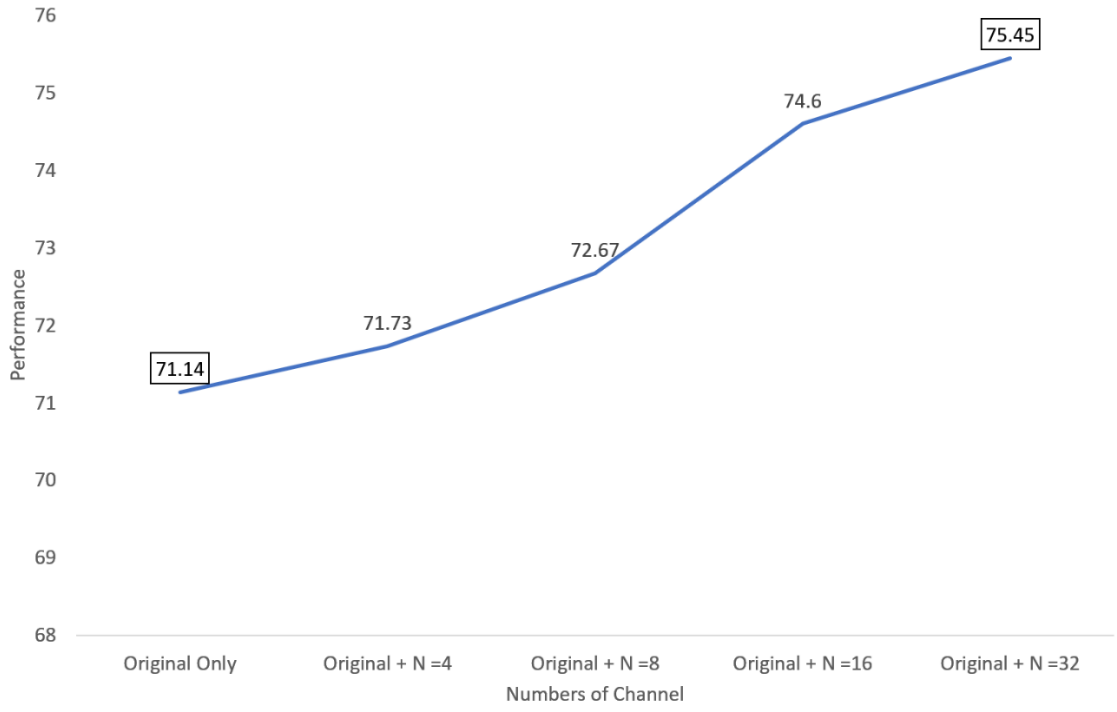

Figure 13. performance curve on noise corrupted audio speech at SNR=10 dB tested with the original data set and the augmented data sets. From left to right, “Original Only” means training with 100% original dataset, “Original + N =4” means training with the

augmented dataset (50% original dataset plus 50% 4-channel vocoded speech), and the same mixing strategy for N=8, N=16, and N=32.

By comparing confusion matrixes in Figure 14 and Figure 15, it is clear that when classifying “background,” the KWS trained with 100% original dataset has 5% less accuracy than the KWS trained with 50% original dataset plus 50% 32-channel vocoded speech. Since vocoded speech has a certain level of distortion, the augmented dataset from the mixture of 50% original plus 50% 32-channel vocoded speech contains features from noise. It has better adaptability when dealing with background noise. For example, as shown in Figure 14, the KWS trained with 100% original dataset falsely predicted the keyword “no” as keyword “background” 119 times. As shown in Figure 15, the KWS trained with 50% original dataset plus 50% 32-channel vocoded speech only falsely predicted keyword “no” as keyword “background” 77 times.

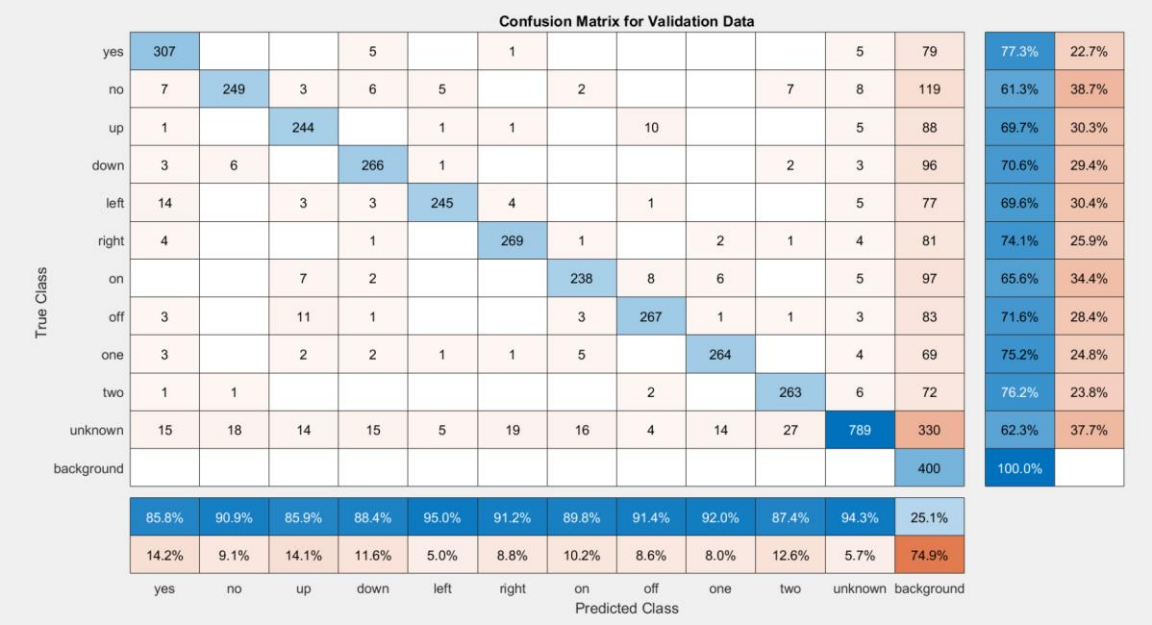

Figure 14. Confusion Matrix of trained KWS with 100% original dataset (“Original Only” in Figure 8), when classifying the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with unknown and background noise.

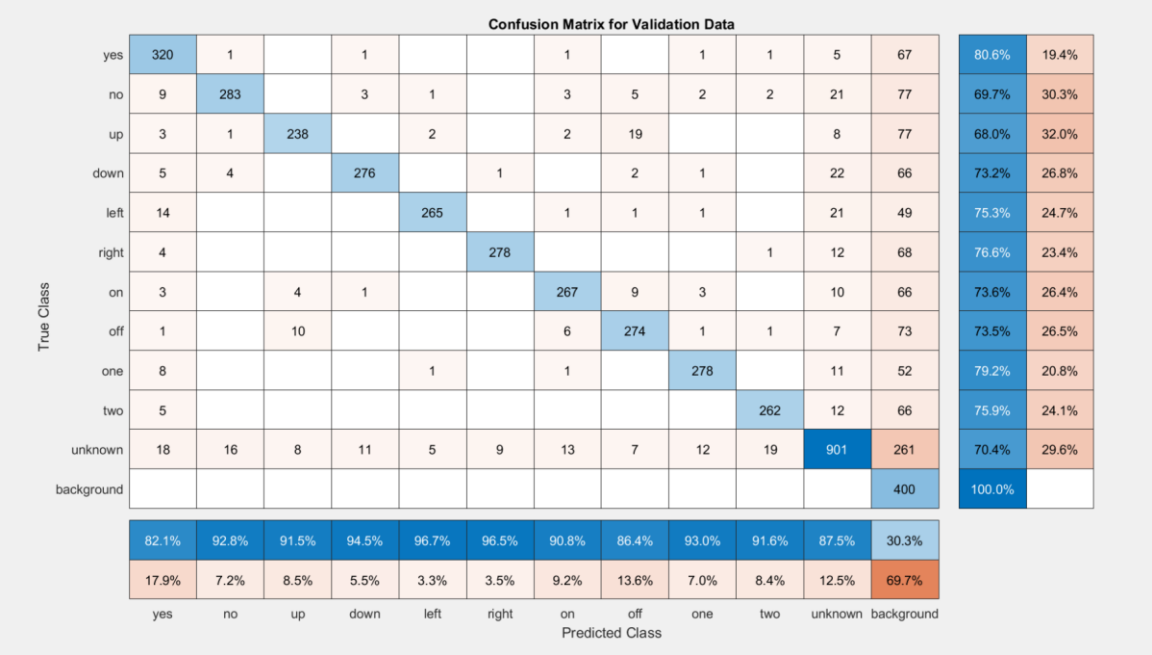

Figure 15. 32 Confusion Matrix of trained KWS with 50% original dataset plus 50% 32-channel vocoded speech (“Original + N = 32” in Figure 8), when classifying the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with unknown and background noise.

Chapter 3. Speech Denoising

This chapter covers the process of speech denoising by using both CNN and a fully connected network.

3.1. Structure of Speech Denoising

As shown in Figure 16, the process of denoising speech using NNs is divided into three steps: Preprocessing, Training, and Postprocessing. And, all three steps are implemented in MATLAB.

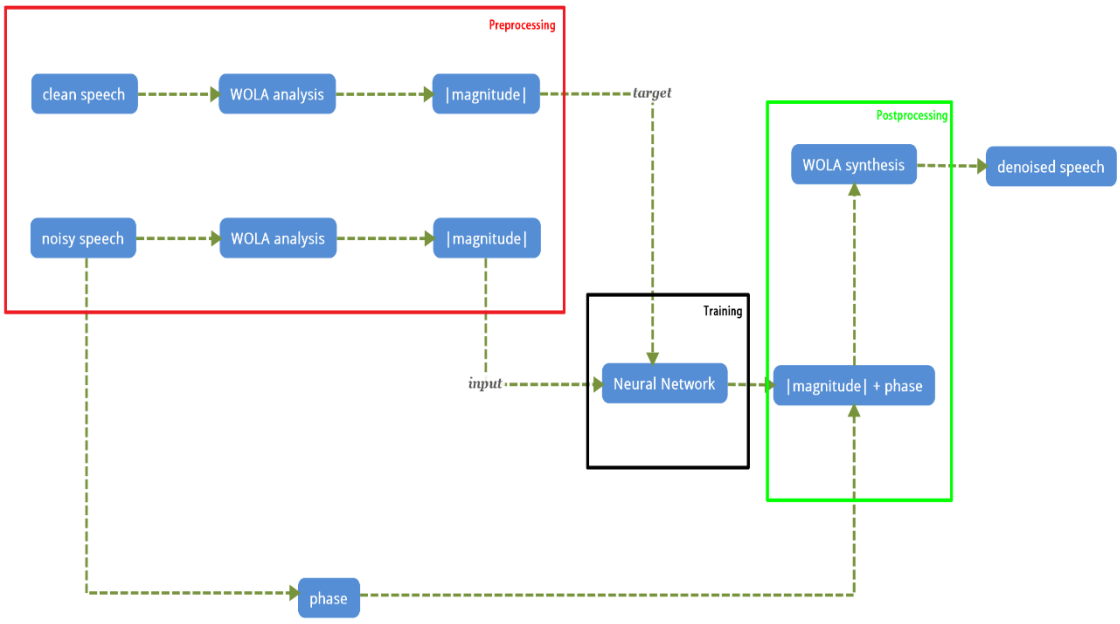

Figure 16. Speech denoising flowchart

3.2 Process of Speech Denoising

3.2.1 Preprocessing

3.2.1.1 Dataset

The experiment was conducted on the Google's Speech Commands dataset [2] and TIMIT database [34] with added 27 different types of noise speeches were collected from freely available online resources. The noise is mostly babble but includes different types of noise like instrumental sounds. Both data in the training set (4620 utterances) and the testing set (200 utterances) were added with one of 27 noise speeches at 0dB SNR. After all feature extraction steps were completed, 20% of the training features were assigned as the validation set.

3.2.1.2 Feature Extraction

Since human speech generally falls below 4-5 kHz, the audio signals ("clean speech" and "noisy speech") were downsampled to 8kHz, and the silent frames were removed from the signal. The operation of WOLA is determined by a number of parameters: analysis window size (La), synthesis window size (Ls), input block step size (R), and FFT size (N). WOLA filterbanks can do either EVEN or ODD stacking. ODD stacking has N/2 full-width bands; EVEN stacking has N/2-1 full-width bands and two half-width bands at DC and the Nyquist frequency. The spectral vectors were computed using La = 256, Ls = 256, N = 256, R = 64, and even stacking for WOLA analysis. The frequency resolution was 31.25 Hz (=4kHz/128) per each frequency bin. For both CNN and fully connected, the input feature consisted of 8 consecutive noisy WOLA magnitude vectors (size: 128 × 8, duration: 100ms). Both input features were standardized to have zero mean and unit variance.

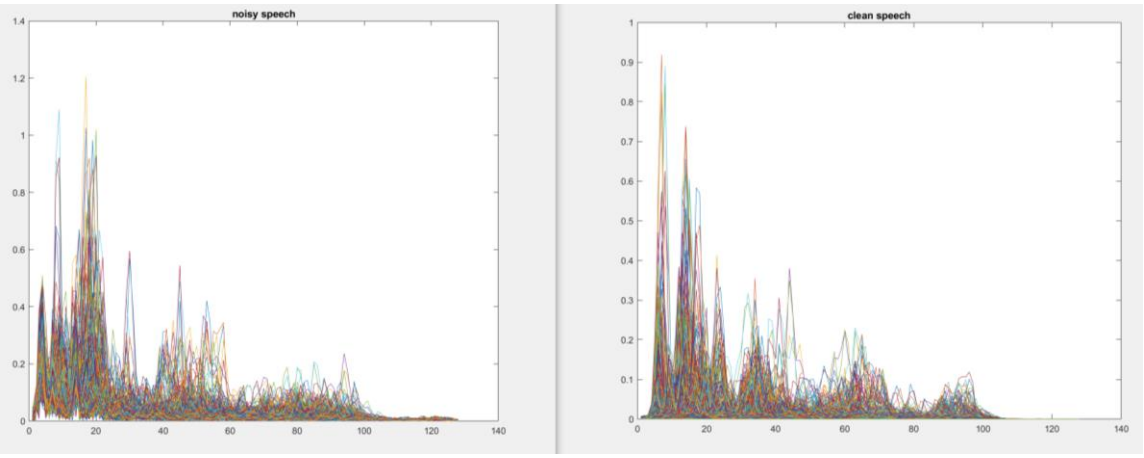

Figure 17. The left plot shows the overlay of 128 feature curves of noisy speech after WOLA analysis. The right plot shows the overlay of 128 feature curves of clean speech after WOLA analysis.

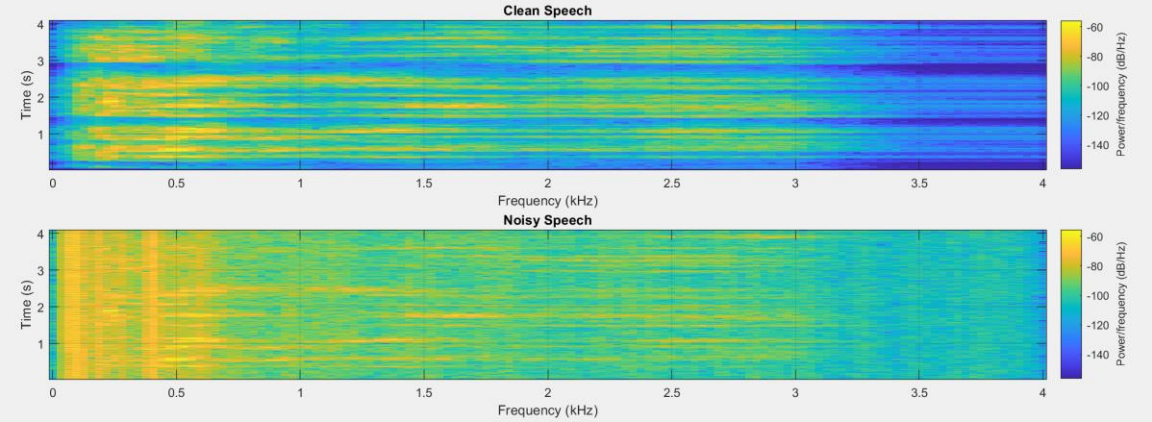

Figure 18. The upper plot shows the spectrogram of clean speech, and the lower plot shows the spectrogram of noisy speech.

As shown in Table 1, we simulate MFCC and WOLA execution time when extracting the same audio input. When all the settings of WOLA and MFCC are approaching 256, as an extreme condition for MFCC and WOLA, the execution time of WOLA is much faster than MFCC. However, if we decrease the number of “OverlapLenth” in MFCC and “R” in WOLA, the execution time of MFCC will decrease and outperform WOLA when “OverlapLength” is less than 128.

	WOLA	MFCC		WindowLength	OverlapLength		La = 256
1s	0.009587	0.128089		256	255		Ls = 256
2s	0.110566	0.777381					N = 256
3s	0.122945	0.837688					R = 256
4s	0.185415	1.138788					
5s	0.23362	1.444826					
6s	0.19371	1.668241					

Table 1. An example of MFCC vs. WOLA execution time in software simulation, 1s – 6s is the audio length.

3.2.2 Training

The fully connected network is simple, which has an input layer, two fully connected layers followed by a batch normalization Layer and a ReLU layer, a single fully connected layer, and a regression layer, as shown in Figure 19.

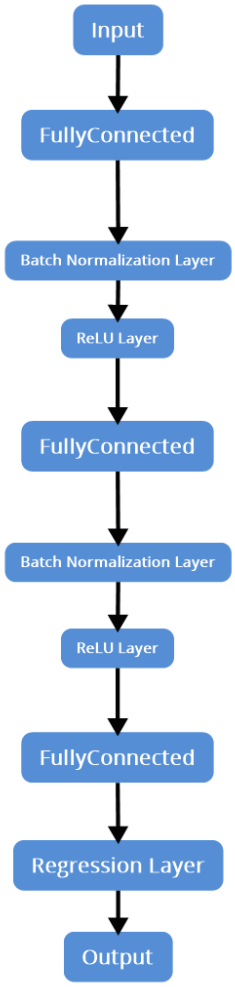

Figure 19. Fully connected network structure used in speech denoising.

The CNN is more complex than the fully connected network, which has a 2-D input layer comprising 16 convolutional layers (five *ConvBlock and one ConvBlock). The first 15 convolutional layers are groups of 3 layers, repeated five times, with filter widths of 9, 5, and 9, and the number of filters of 18, 30, and 8, respectively. The last convolutional layer has a filter width of 129 and 1 filter. In this network, convolutions are performed in only one direction (along the frequency dimension), and the filter width along the time dimension is set to 1 for all layers except the first one. Similar to the fully connected

network, convolutional layers are followed by ReLu and batch normalization layers, as shown in Figure 20.

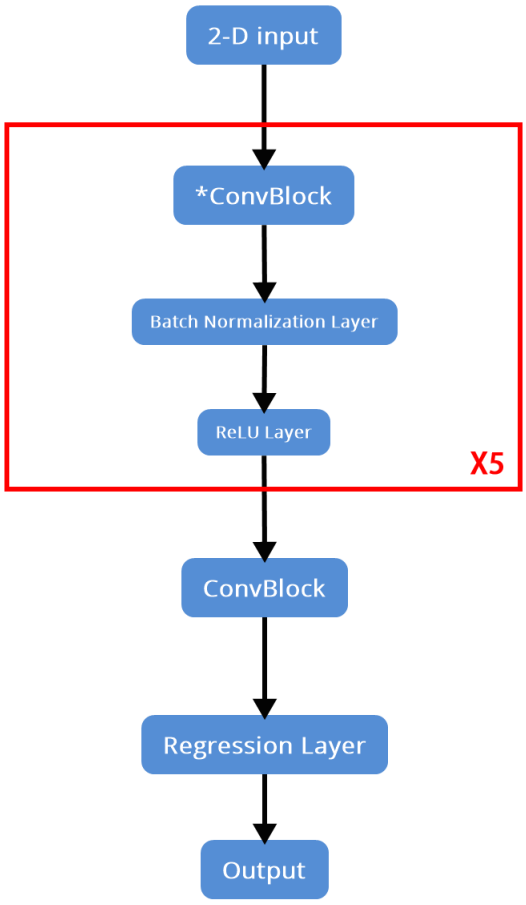

Figure 20. CNN structure used in speech denoising. “X5” means multiply by five times.

Both CNN and fully connected networks were trained using backpropagation with gradient descent optimization using Adam [53] with a mini-batch size of 128. The learning rate started from 1e-5 maximum Epochs = 3 and decreased the learning rate by a specified factor (0.9) every time a certain number of epochs (1) has passed.

3.2.3 Postprocessing

To avoid extreme differences (more than 45 degrees) between the noisy and clean phase, the clean spectral magnitude was encoded described in [35]:

$$S_{phase} = S_{clean} \cos(\theta_{clean} - \theta_{noisy})$$

Besides, the spectral phase was not used in the training phase. At reconstruction, a noisy spectral phase was used instead to perform WOLA synthesis with the same parameter setup in preprocessing and recovering speech. However, because the human ear is not susceptible to phase differences smaller than 45 degrees, the distortion was negligible. Through 'phase aware scaling,' the phase mismatch is smaller than 45 degrees, and For both networks, the output feature consisted of a 'phase aware' magnitude vector (size: 128×1, duration: 32ms) and was standardized to have zero mean and unit variance.

3.3 Results

Comparing Figure 21 and Figure 22, it can be observed that both CNN and fully connected networks have successfully denoised the noisy speech. However, denoised speech (Convolutional) has a slightly better performance compared to the denoised speech (Fully connected), where both the plot and the spectrogram showed that denoised speech (Convolutional) is more similar to Clean Speech. Moreover, by comparing the plots and hearing the output of denoised speech (Convolutional) and denoised speech (Fully connected), it's observed that denoised speech (Convolutional) is similar to denoised speech (Fully connected). The only difference is denoised speech (Convolutional) sounds a little bit clearer than denoised speech (Fully connected).

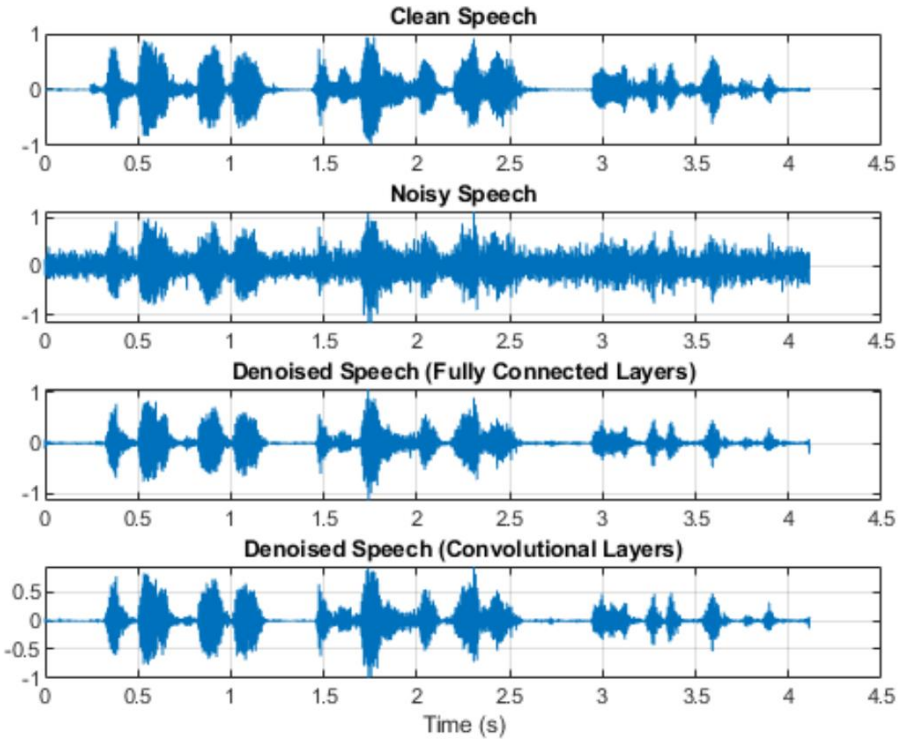

Figure 21. From top to down: plots of amplitude vs. time of Clean Speech, Noisy Speech, Denoised Speech (Fully connected), and Denoised Speech (Convolutional).

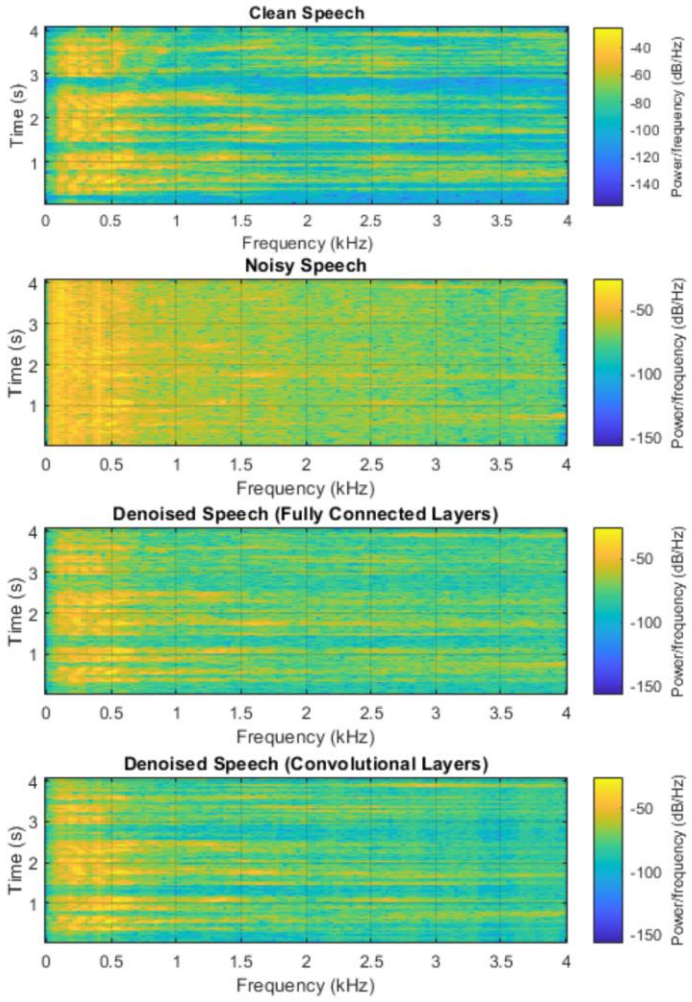

Figure 22. From top to down: spectrograms of Clean Speech, Noisy Speech, Denoised Speech (Fully connected), and Denoised Speech (Convolutional).

Chapter 4. Incorporating Speech Denoising into the KWS System

This chapter covers the process of combining the best-trained KWS system proposed in Chapter 2 with the speech denoising system proposed in Chapter 3.

4.1. Speech Denoising with 32-channel Vocoded Speeches

We used the 32-channel vocoded speech as the new target in the speech denoising system proposed in Chapter 3. Thus, the speech denoising system will convert noisy speech to 32-channel vocoded speech. The results are shown in Figures 23 and 24, which are similar to Figures 21 and 22 – both denoised speech es were successfully performed.

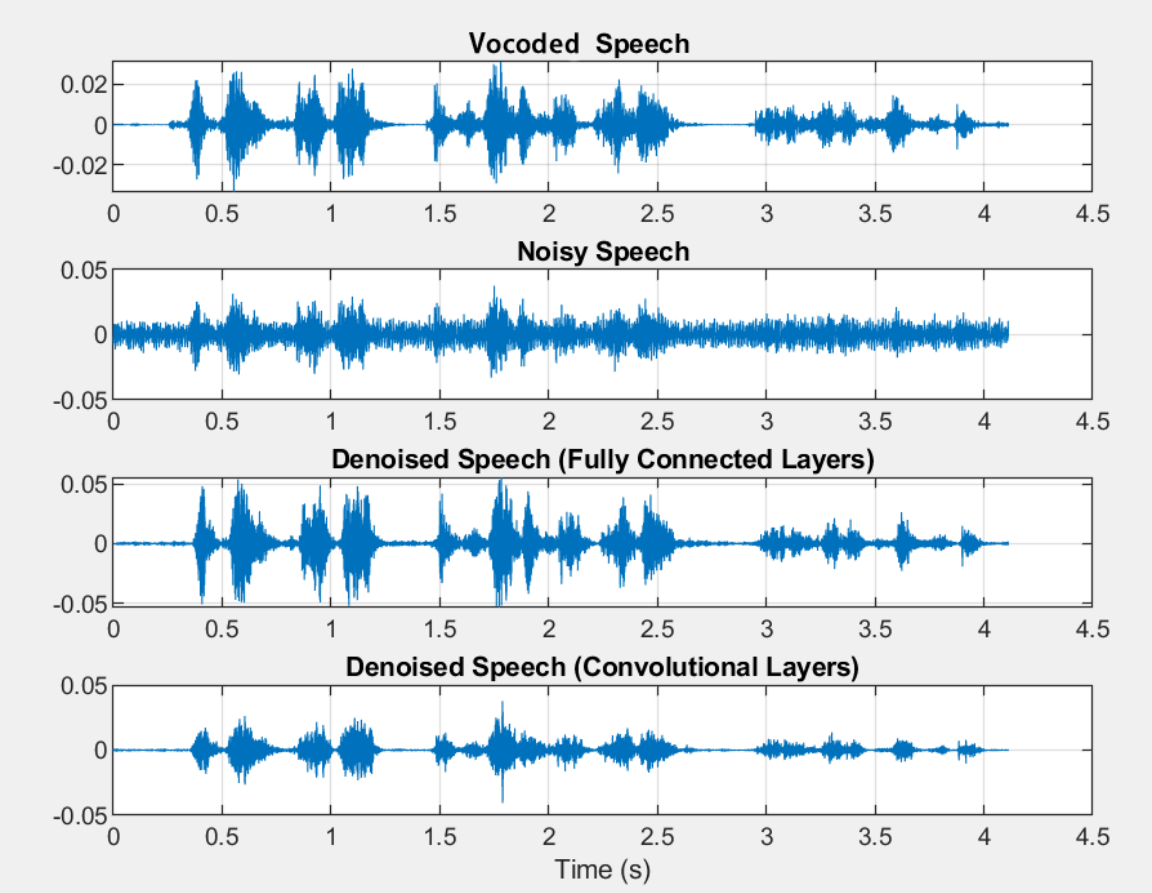

Figure 23. Using vocoded clean speech and vocoded noisy speech. From top to down: plot of time vs. amplitude of Vocoded Speech, Noisy Speech, Denoised Speech (Fully connected), and Denoised Speech (Convolutional).

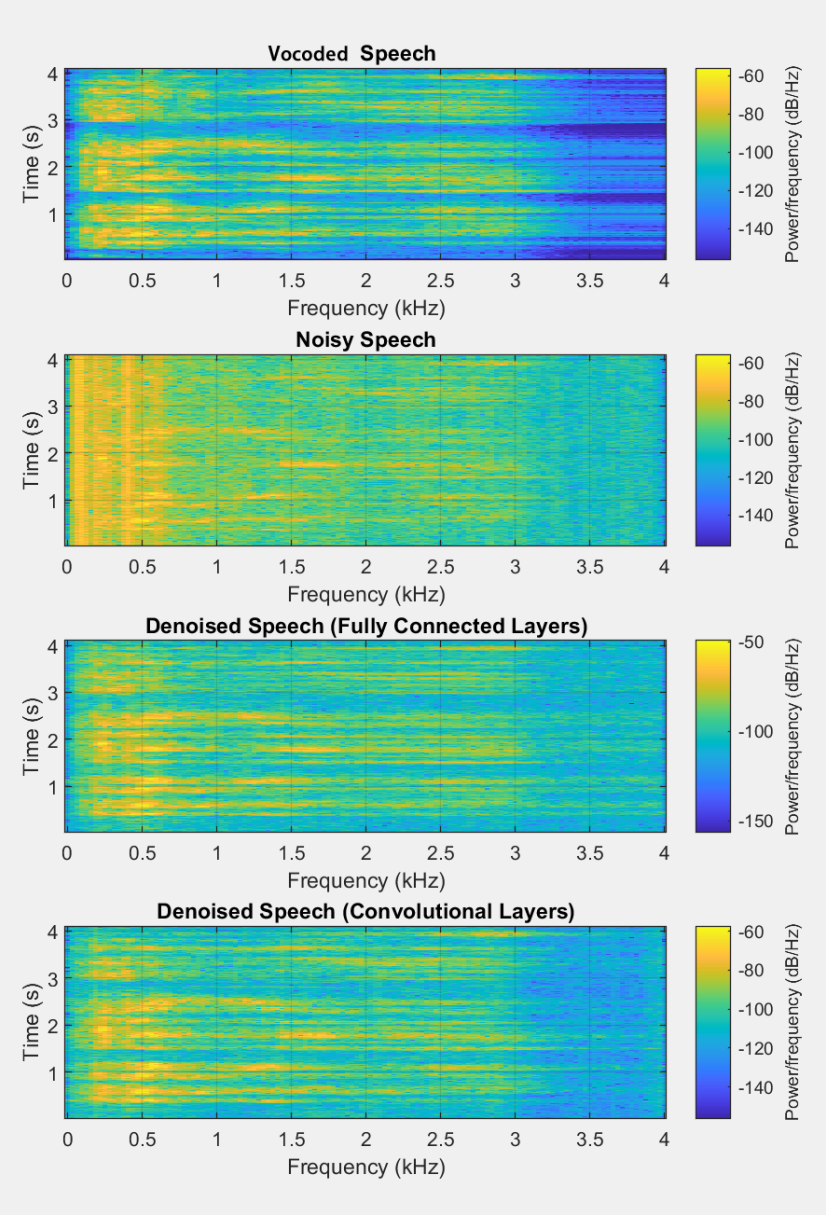

Figure 24. Using vocoded clean speech and vocoded noisy speech. From top to down: spectrograms of Vocoded Speech, Noisy Speech, Denoised Speech (Fully connected), and Denoised Speech (Convolutional).

4.2 Combining Speech Denoising with KWS System

In this section, we examine the performance of the combing speech denoising system proposed in Chapter 3 with the KWS trained with the 32-channel vocoded speech proposed in Chapter 2. The process of this examination was described in Figure 25. We used 5dB SNR noise corrupted noisy voices as input. Those noisy voices contain 12 keywords: "yes," "no," "up," "down," "left," "right," "on," "off," "one," "two," along with unknown and background noise, with each keyword, has an amount of 400. After speech denoising, 400 clean voices were generated for each keyword for further classification. Figure 26 shows the confusion matrix when classifying the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with the unknown and background noise. The validation accuracy = 88.89%, which is better than the best result shown in Figure 13.

Figure 25. Using speech denoising system to convert noisy voice input into clean voice, then feed clean voice into the KWS trained with 32-channel vocoded speech.

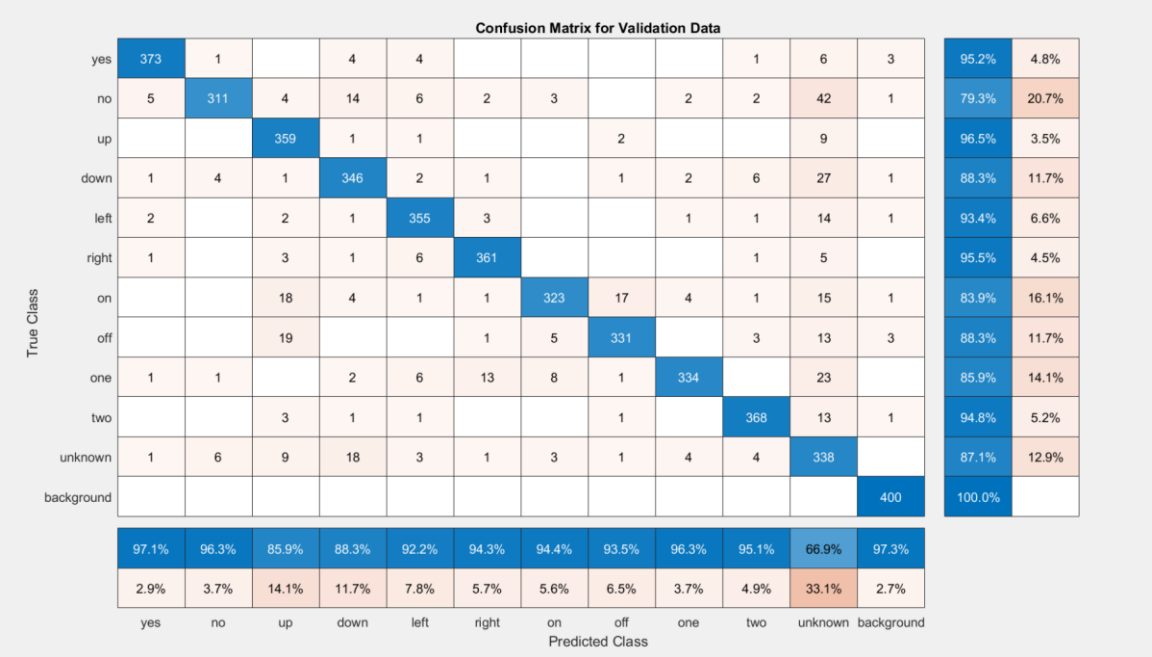

Figure 26. Confusion Matrix of the trained KWS as shown in Figure 20, when classifying the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with the unknown and background noise.

4.3 Combining Vocoded Speech Denoising with KWS System

In this section, we examine the performance of combing the vocoded speech denoising system proposed in Chapter 4.1 with the KWS trained with the 32-channel vocoded speech proposed in Chapter 2. The process of this examination was described in Figure 27. We used 5dB noise corrupted noisy voices as input. Those noisy voices contain 12 keywords: "yes," "no," "up," "down," "left," "right," "on," "off," "one," "two," along with unknown and background noise, with each keyword, has an amount of 400. After speech denoising, 400 clean voices were generated for each keyword for further classification. Figure 28 showed the confusion matrix when classifying the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with unknown and background noise. The validation accuracy = 91.07%, which is better than the best result shown in Figure 13.

Figure 27. Using Vocoded Speech Denoising system to convert noisy voice input into 32-channel vocoded clean voice, then feed clean voice into the KWS trained with 32-channel vocoded speech.

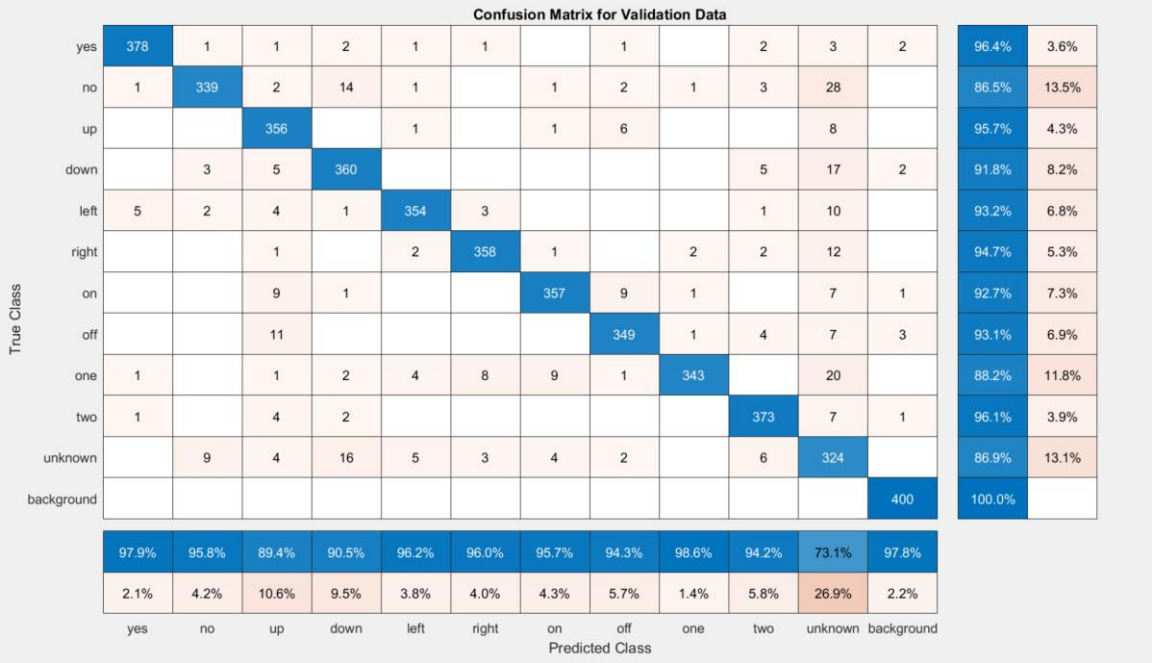

Figure 28. Confusion Matrix of trained KWS showed in Figure 22, when classifying the following 12 keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "one", "two", along with unknown and background noise.

As summarized in Figure 29, the KWS system setup described in Chapter 4.3 achieved the highest accuracy in our study – 91.07%. It's proven that using a speech denoising system in front of the KWS system will greatly improve the accuracy.

Figure 29. The performance curve of two speech denoising systems + KWS trained with 32-channel vocoded speech, comparing with the summary data from Figure 27.

We further tested the performance of vocoded speech in noise by adding +10dB SNR noise corrupted data in training and using k=6 cross-validation with the whole dataset: original data + vocoded data + noise corrupted data. As shown in Figure 30, the use of vocoded data + noise corrupted data will significantly improve the KWS performance in noise.

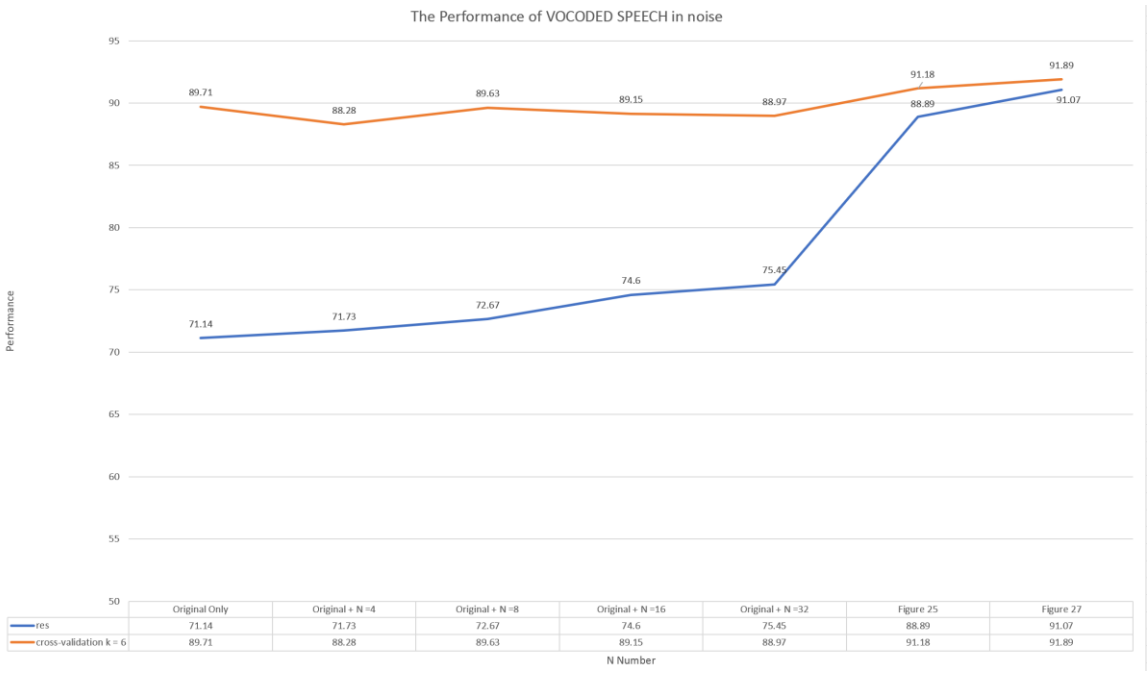

Figure 30. K=6 cross-validation with the whole dataset: original data + vocoded data + noise corrupted data, comparing results in Figure 29.

Chapter 5. Implementation on Microcontroller

5.1 Design

This chapter covers the processes of transferring the KWS system described in Chapter 2 into an Arduino Nano 33 BLE Sense (Clock 64MHz, Flash 1MB, RAM 256KB), including necessary modification in MFCC and the trained CNN proposed in Chapter 3 since the targeted hardware -- Arduino Nano 33 BLE Sense has limited computational power. As shown in Figure 30, the 16-bit built-in microphone in Arduino Nano 33 BLE Sense will read voice input. Then a modified MFCC will extract the incoming voice's feature as the input of the NN classifier. The NN classifier will calculate the final confidence score. The category of the final confidence score will determine if the LED light will light up or not.

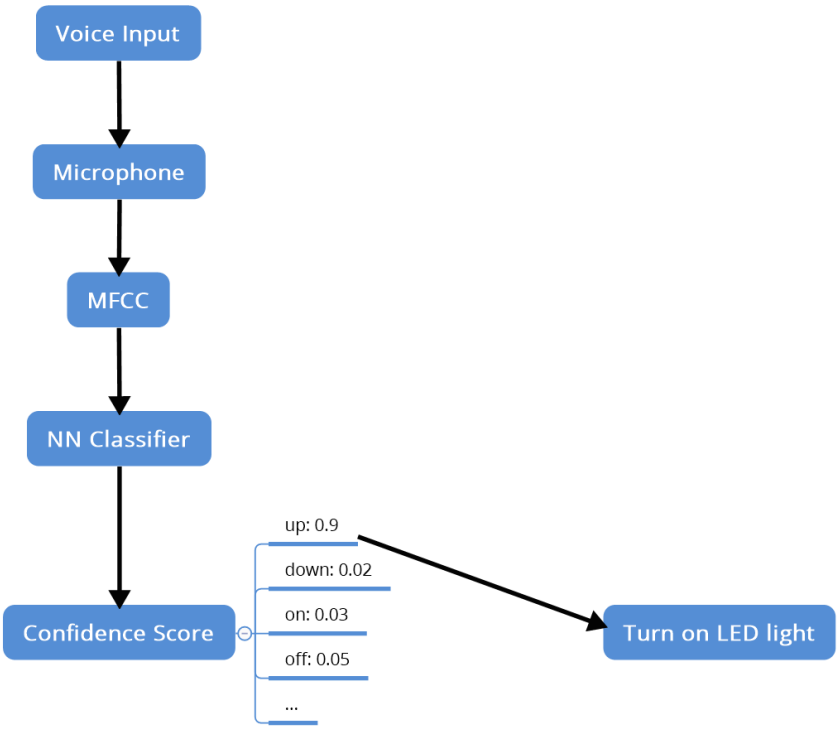

Figure 31. Implementation on microcontroller flowchart.

5.2 Implementation on microcontroller

5.2.1 MFCC

In this chapter, 13 MFCC features (49x13) were computed every 20 ms with a 20 ms frameshift at 256 FFT length. As shown in Figure 31, by reducing the numbers of coefficients from 50 to 13 in MFCC, the resolution of the feature gets worse. But, the processing time of feature extraction will be shorter.

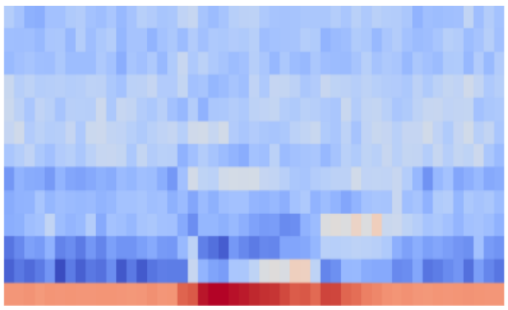

Figure 32. MFCC of 1s voice “down.”

The MFCC feature extraction module was implemented in C++ by using Edge Impulse DSP and Inferencing SDK. By using this SDK, as shown in Figure 32, we simply set up the necessary parameters for MFCC feature extraction, and then we can get corresponding MFCC features from this SDK through Arduino IDE.

```
ei_dsp_config_mfcc_t ei_dsp_config_3 = {
    1,
    13,
    0.02000f,
    0.02000f,
    32,
    256,
    101,
    300,
    0,
    0.98000f,
    1
};
```

Figure 33. Parameters for configuring MFCC by using Edge Impulse DSP and Inferencing SDK

5.2.2 NN Classifier

After reducing the numbers of MFCC coefficients, we also reduced the size and structure of the CNN proposed in Chapter 3. The new reduced CNN, shown in Figure 33, only has six layers in order to reduce the required computational power during NN classification. During training, we reduced the batch size to 32 but kept using Adam optimizer with an initial learning rate of 5e-3.

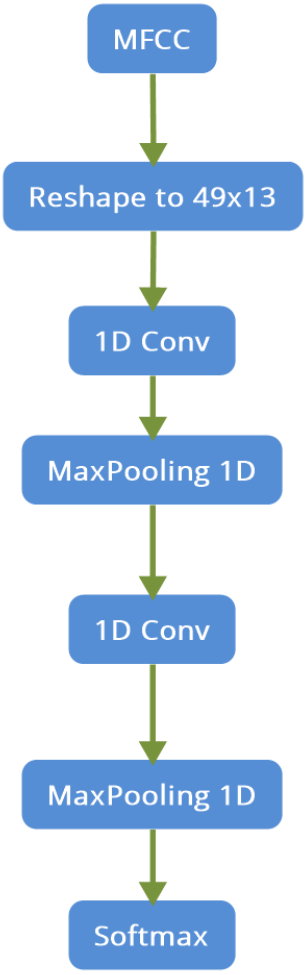

Figure 34. CNN structure used in implementation on microcontroller

In order to efficiently use the trained CNN, we tried int8 quantization to reduce the size, which affects the RAM and ROM usage in hardware. This process was completed by using Edge Optimized Neural (EON) Compiler, which is built upon TensorFlow Lite for microcontrollers that have the built-in int8 quantization function. We used EON Compiler to compile our trained CNN into C++ source code after int8 quantization, and the compiled C++ source code can be used in Arduino IDE shown in Figure 34.

```
const TfArray<1, int> tensor_dimension4 = { 1, { 5 } };
const TfArray<1, float> quant4_scale = { 1, { 0.00033171122777275741, } };
const TfArray<1, int> quant4_zero = { 1, { 0 } };
const TfLiteAffineQuantization quant4 = { (TfLiteFloatArray*)&quant4_scale, (TfLiteIntArray*)&quant4_zero, 0 };
const ALIGN(8) int8_t tensor_data5[5*208] = {
  -107, 20, -13, -9, 13, 18, 35, 11, 26, -28, -26, 3, 4, 3, -66, -7, -55, 18, -12, 7, 35, -6, 7, 26, 10, -22, -40,
  29, -3, 1, 26, 17, 47, 49, -24, -3, -24, -3, 13, -4, 0, 47, -2, 19, 22, -28, 24, 32, 37, 44, -2, -49, 17, -4, 13
  -5, -7, 23, -8, -12, -11, -30, 3, -15, 12, 8, 15, 11, -11, -7, 17, -3, -2, -4, 5, -13, -23, -8, 3, 19, -19, 5, -
  5, 16, 2, -52, -6, -40, -32, -1, 5, -28, 7, 6, 4, -4, 6, 5, 11, -1, 33, -25, -26, -15, -59, -2, -16, -4, 5, 6, -
  -11, -21, -9, 27, -20, -20, -26, 19, 0, -6, 15, -27, 19, 3, -20, -12, -3, -17, -18, -2, -16, -17, 0, 0, 11, 9, 4
};
```

Figure 35. part of C++ source code compiled from EON Compiler after quantizing(int8) trained CNN in this chapter.

5.2.3 Arduino Scripting

After having MFCC and trained CNN (NN classifier) implemented in C++, we can use a script in Arduino IDE to enable the built-in microphone in Arduino Nano 33 BLE Sense. The implemented MFCC function to continuously extract the input signal once per second, and the NN classifier will begin generating confidence score right after the MFCC calculation complete. Based on the confidence score in the corresponding category, the Arduino Nano 33 BLE Sense will determine if the LED light will blink showed in Figure 35.

```
// Turn on led if "on" or "up", turn off if "off" or "down"
if (result.classification[3].value > 0.7 || result.classification[4].value > 0.7){
    digitalWrite(led_pin, HIGH);
  }else {
    digitalWrite(led_pin, LOW);
    }
```

Figure 36. Arduino script that turns on the LED light if "on" or "up" recognized and turns off the LED light if "off" or "down" recognized.

5.3 Results

After we trained the reduced CNN by using 32-channel vocoded speech and validated the trained CNN in +10dB SNR noise corrupted clean speech, the accuracy showed in Figure 36, which is 76.1% is similar to the accuracy shown in Figure 8 as 75.45%.

	_NOISE	DOWN	OFF	ON	UP
_NOISE	99.6%	0%	0.4%	0%	0%
DOWN	4.1%	80.2%	7.8%	5.3%	2.5%
OFF	2.3%	0%	93.5%	3.7%	0.5%
ON	3.0%	1.3%	19.0%	75.9%	0.9%
UP	12.5%	3.9%	44.0%	4.7%	35.0%
F1 SCORE	0.90	0.86	0.68	0.80	0.51

Figure 37. Confusion matrix of trained the reduced CNN by using 32-channel vocoded speech.

After uploading the final script to Arduino Nano 33 BLE Sense, the output shown in Figure 32 as the whole implementation on the microcontroller part needs 229464 bytes to store, which is about 23% of the entire flash of the Arduino Nano 33 BLE Sense. When running, the whole implementation on the microcontroller will need 47264 bytes of RAM, which is about 18% of the entire dynamic memory of the Arduino Nano 33 BLE Sense. Figure 37 proved that the C++ implemented MFCC and NN classifier, as well as the Arduino script, can run in the Arduino Nano 33 BLE Sense without consuming all its computational power.

Figure 38. Arduino IDE output when successfully uploads scripts.

When testing the Arduino Nano 33 BLE Sense, we need to stay close enough to it since its microphone is too small to hear the voice from a long distance. Staying our mouth closer to the board will help to increase the confidence score shown in Figure 38. Once the confidence score on “on” or “up” is greater than 0.7 (in most cases, if we stay close enough to speak “down,” “off,” “up,” “on,” the corresponding confidence score will always be greater than 0.7. If we put the board near a running fan, the confidence score of “noise” will be greater than 0.7. The LED light will blink, as shown in Figure 40. If the LED light is not triggered, the display of the board will be shown in Figure 39.

```
Predictions (DSP: 123 ms., Classification: 11 ms., Anomaly: 0 ms.):
    _noise: 0.00000
    down: 0.02344
    off: 0.04688
    on: 0.92578
    up: 0.00391

Predictions (DSP: 124 ms., Classification: 10 ms., Anomaly: 0 ms.):
    _noise: 0.13672
    down: 0.03516
    off: 0.00391
    on: 0.01953
    up: 0.80469

Predictions (DSP: 123 ms., Classification: 11 ms., Anomaly: 0 ms.):
    _noise: 0.91016
    down: 0.01953
    off: 0.00781
    on: 0.02734
    up: 0.03906

Predictions (DSP: 123 ms., Classification: 11 ms., Anomaly: 0 ms.):
    _noise: 0.00000
    down: 0.00391
    off: 0.86719
    on: 0.12500
    up: 0.00391

Predictions (DSP: 123 ms., Classification: 11 ms., Anomaly: 0 ms.):
    _noise: 0.01953
    down: 0.85156
    off: 0.00391
    on: 0.01953
    up: 0.10547
```

Figure 39. MFCC and classification processing time and confidence score printed out on Arduino Serial Monitor in real-time.

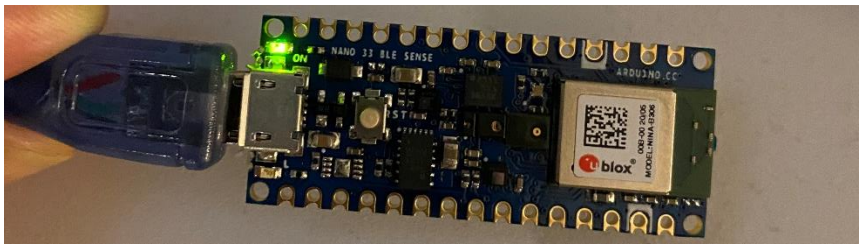

Figure 40. LED light (orange) does not blink when the confidence score in the corresponding category does not meet the requirement.

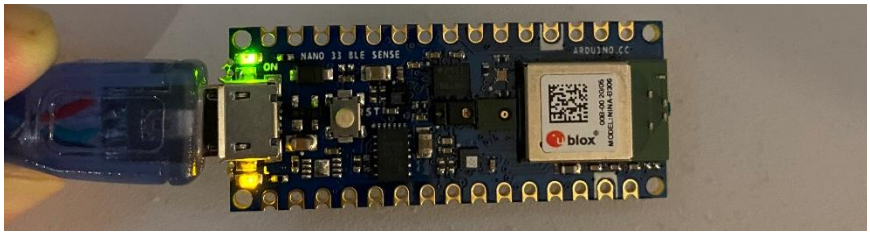

Figure 41. LED light (orange) blinks when the confidence score in the corresponding category meets the requirement.

Chapter 6. Conclusions and Future Studies

6.1 Conclusions

In this thesis, we first demonstrated the possibility of using 16-channel and 32-channel vocoded speech to increase the size of training data in the KWS system. While the vocoded speech in the trained CNN performed similarly to human subjects, 32-channel vocoded speech with 92.96% accuracy has nearly the same performance as human subjects recognizing sentences in 16-channel (93%). Such finding suggests that vocoded speech has the potential for being used in training the KWS system. On the other hand, the augmented data by mixing vocoded speech with the original dataset improves the KWS system in moderate noise (SNR = 10dB), resulting in an improvement of 4.3% by mixing the 32-channel vocoded speech with the original dataset. Training and testing with a larger set of keywords need to be further studied to show how vocoded speech can add benefits to a real KWS system.

Subsequently, a neural-network-based speech denoising system was introduced to solve the problem of using noisy speech in the proposed KWS system in Figure 7. Our results showed that both the fully connected speech denoising system and convolutional speech denoising system successfully converted noisy speech into clean speech using computed regression after training. By hearing the output from both denoising systems, we confirmed that convolutional speech denoising has better performance as it has less noise in the background.

We further demonstrated this by using a speech denoising system as the input to the best KWS system created in Chapter 3. The overall system shown in Figure 27 has a promising improvement: ~15% in validation accuracy, which strongly proves that using a speech denoising system in front of a KWS system is necessary and beneficial.

Figure 42. The overall system of this thesis.

By doing implementation on a microcontroller with 32-channel vocoded speech, we demonstrated that the KWS system trained with a 32-channel vocoded dataset could be implemented on a microcontroller with necessary NN optimization and required modifications in preprocessing.

6.2 Future Studies

In this thesis, we only tested a 1:1 ratio in data augmentation using N-channel vocoded speech. For more particular use, other ratios can be tested to best fit different kinds of requirements. For example, by increasing the number of vocoded speeches in the augmented dataset, the trained KWS may perform better in noisy environments since more noise features will be introduced. Moreover, we concluded using 32-channel vocoded speech in the augmented dataset can improve the KWS system performance in +10 dB SNR noise. Using 4-8 channel vocoded speech in the augmented dataset may improve the KWS system performance in +0 dB or +5dB SNR noise. Since 4-8 channel vocoded speech will introduce more noise features. Furthermore, using the combination of different channels of vocoded speech may further increase the KWS system performance in various noisy environments. For example, using 1-32 channel vocoded speech in the augmented dataset may resulting in a trained KWS system with high performance in 0 – +10 dB SNR noise.

In order to accurately measure the executing time of MFCC and WOLA, implementing those two feature extractors in the same hardware platform is recommended. Since executing time in software simulation heavily depending on

algorithm optimization, for example, using the same DSP for implementation and measuring the executing time between MFCC and WOLA.

It's exciting to see vocoded data + noise corrupted data will further improve KWS performance in noise. And, more future work can be tested to see how vocoded data can benefit model trained with noise corrupted data. For example, if performance drop by removing vocoded data, or how can vocoded data benefit KWS in different noise levels.

On the other hand, the use of Recurrent Neural Network (RNN) may outperform CNN due to its ability to use the internal state (memory) to process sequences of inputs. Unlike CNN, in which all inputs are independent of each other. All the inputs in RNN are related to each other, which makes RNN incredibly successful on some sequence modeling tasks. However, RNN cannot connect the relevant information if the gap between the relevant inputs is large. In this case, long-short-term memory (LSTM) network, a particular type of RNN, is introduced to handle long-term dependencies using gating mechanisms.

The overall system shown in Figure 41 can be further implemented on the Arduino Nano 33 BLE Sense to check if there is any improvement in real-time since the reduced MFCC and NN classifier only consumed around 20% of its ROM and RAM, as shown in Figure 37. The rest of 80% ROM and RAM may be enough for implementing NN-based speech denoising and could significantly boost the performance of the trained KWS system since all inputs to the trained KWS system will be clean speeches.

REFERENCES

[1] G. Chen, C. Parada and G. Heigold, "Small-footprint keyword spotting using deep neural networks," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 2014, pp. 4087-4091, doi: 10.1109/ICASSP.2014.6854370.

[2] P. Warden, "Speech commands: A dataset for limited vocabulary speech recognition," arXiv:1804.03209, 2018.

[3] M. Dorman and P. Loizou, "Speech Intelligibility as a Function of the Number of Channels of Stimulation for Normal-Hearing Listeners and Patients with Cochlear Implants," The American Journal of Otology, Vol 18, No.6(Suppl), 1997.

[4] L. Friesen, R. Shannon, D. Baskent, and X. Wang, "Speech recognition in noise as a function of the number of spectral channels: Comparison of acoustic hearing and cochlear implants," The Journal of the Acoustical Society of America 110, 1150 (2001); doi: 10.1121/1.1381538.

[5] M. Dorman, P. Loizou, and D. Rainey, "Speech intelligibility as a function of the number of channels of stimulation for signal processors using sine-wave and noise-band outputs," The Journal of the Acoustical Society of America 102, 2403 (1997); doi: 10.1121/1.419603.

[6] P. Loizou, M. Dorman, and Z. Tu, "On the number of channels needed to understand speech," The Journal of the Acoustical Society of America 106, 2097 (1999); doi: 10.1121/1.427954.

[7] M. Mara, "Media and Prosthesis: The Vocoder, the Artificial Larynx, and the History of Signal Processing." Qui Parle: Critical Humanities and Social Sciences, vol. 21 no. 1, 2012, pp. 107-149.

[8] B. Harald, "History of Electronic Sound Modification", Journal of the Audio Engineering Society. 32 (10): 730–739.

[9] M. Airaksinen, L. Juvela, P. Alku and O. Räsänen, "Data Augmentation Strategies for Neural Network F0 Estimation," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 2019, pp. 6485-6489, doi: 10.1109/ICASSP.2019.8683041.

[10] E. Chang and R. Lippmann, “Using voice transformations to create additional training talkers for word spotting,” Advances in Neural Information Processing Systems, pp. 875–882.

[11] T. Quatieri and R. McAulay, “Shape invariant time-scale and pitch modification of speech,” IEEE Transactions on Signal Processing, 40(3), 497–510. DOI: https://doi.org/10.1109/78.120793.

[12] N. Jaitly, E. Hinton, “Vocal tract length perturbation (VTLP) improves speech recognition,” ICML Workshop on Deep Learning for Audio, Speech and Language, volume 117.

[13] N. Kanda, R. Takeda, and Y. Obuchi, “Elastic spectral distortion for low resource speech recognition with deep neural networks,” IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 309–314. DOI: https://doi.org/10.1109/ASRU. 2013.6707748.

[14] A. Ragni, K. Knill, S. Rath and M. Gales “Data augmentation for low resource languages,” INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, 14-18 Sep 2014, Singapore. International Speech Communication Association (ISCA), pp. 810-814.

[15] X. Cui, V. Goel, and B. Kingsbury, “Data augmentation for deep neural network acoustic modeling,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, 23(9), pp. 1469–1477.

[16] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, “Audio augmentation for speech recognition,” INTERSPEECH, 2015.

[17] R. Hsiao, J. Ma, W. Hartmann, M. Karafiat, F. Grezl, L. Burget, I. Szoke, J. H. Cernocky, S. Watanabe, Z. Chen, S. H. Mallidi, H. Hermansky, S. Tsakalidis, and R. Schwartz, “Robust speech recognition in unknown reverberant and noisy conditions,” 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), 2015, pp. 533–538.

[18] A. Raju, S. Panchapagesan, X. Liu, A. Mandal, and N. Strom, “Data Augmentation for Robust Keyword Spotting Under Playback Interference,” arXiv preprint arXiv:1808.00563, 2018.

[19] K. Davis, R. Biddulph, and S. Balashek, “Automatic recognition of spoken digits,” The Journal of the Acoustical Society of America, vol. 24, no. 6, pp. 637–642, 1952.

[20] S. Levinson, L. Rabiner, and M. Sondhi, “An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition,” The Bell System Technical Journal, vol. 62, no. 4, pp. 1035– 1074, Apr. 1983.

[21] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting", INTERSPEECH, 2015, 1478-1482.

[22] Y. Huang, T. Hughes, T. Z. Shabestary, and T. Applebaum, "Supervised Noise Reduction for Multichannel Keyword Spotting," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 5474-5478, doi: 10.1109/ICASSP.2018.8462346.

[23] N. Zeghidour and D. Grangier, “Wavesplit: End-to-end speech separation by speaker clustering,” arXiv preprint arXiv:2002.08933, 2020.

[24] D. Wang and J. Chen, "Supervised Speech Separation Based on Deep Learning: An Overview," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 26, no. 10, pp. 1702-1726, Oct. 2018, doi: 10.1109/TASLP.2018.2842159.

[25] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, “Speech enhancement based on deep denoising autoencoder,” INTERSPEECH, 2013, no. August, pp. 436–440.

[26] Y. Lai, F. Chen, S. Wang, X. Lu, Y. Tsao and C. Lee, "A Deep Denoising Autoencoder Approach to Improving the Intelligibility of Vocoded Speech in Cochlear Implant Simulation," in IEEE Transactions on Biomedical Engineering, vol. 64, no. 7, pp. 1568-1578, July 2017, doi: 10.1109/TBME.2016.2613960.

[27] S. R. Park and J. Lee, “A fully convolutional neural network for speech enhancement,” arXiv preprint arXiv:1609.07132, 2016.

[28] D. Liu, P. Smaragdis, and M. Kim, “Experiments on deep learning for speech denoising,” INTERSPEECH, 2014, pp. 2685–2689.

[29] S. Ahadi, H. Sheykhzadeh, R. Brennan, and G. Freeman, “A Weighted Overlap Add-based Front-end for Speech Recognition,” Journal of Iranian Association of Electrical and Electronic Engineers, 1(2), 15-20.

[30] J. Junkawitsch, L. Neubauer, H. Höge, and G. Ruske, "A New Keyword Spotting Algorithm with Pre-Calculated Optimal Thresholds," Technical University of Munich, Germany, 1996.

[31] I. Kamarulafizam, S. Hussain, and J. Najeb, K. Arif, and A. Chowdhury, “Heart Sound Analysis Using MFCC and Time Frequency Distribution,” 3rd Kuala Lumpur International Conference on Biomedical Engineering, 2006, pp. 402-405.

[32] Y. Zhang, N. Suda, L. Lai, and V. Chandra, “Hello edge: Keyword spotting on microcontrollers,” arXiv preprint arXiv:1711.07128, 2017.

[33] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” International Conference on Learning Representations (ICLR), 2015.

[34] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, and D. Pallett, “DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM. NIST speech disc 1-1.1,” NASA STI/Recon technical report, vol. 93, 1993.

[35] P. Mowlaee and R. Saeidi, “Iterative closed-loop phase-aware single-channel speech enhancement,” IEEE Signal Processing Letters, vol. 20, no. 12, pp. 1235–1239, 2013.

[36] C. Choi and Y. Lee, “A Review of Stimulating Strategies for Cochlear Implants,” Cochlear Implant Research Updates, Dr. Cila Umat (Ed.), ISBN: 978-953-51- 0582-4.

[37] R. Li and K. Nie, "Improving Keywords Spotting Performance in Noise with Augmented Dataset from Vocoded Speech," 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Auckland, New Zealand, 2020, pp. 1653-1656.