Fine-tuning an Automatic Speech Recognition Model for a Canadian Indigenous Counselling Program

by

Emmanuel Olaniyanu

A Thesis submitted to the Faculty of Graduate Studies of

In partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Graduate Program of Biomedical Engineering

University of Manitoba

Winnipeg, Canada

Abstract

Automatic Speech Recognition (ASR) systems have seen a marked improvement in performance after adopting the End-To-End (E2E) approach. However, the performance of ASR models is still largely dependent on the quantity and quality of data they are trained on. Popular ASR systems today are trained on thousands of hours of data but consistently fail to maintain their performance when exposed to outlier accents, vocal pitches, and demographic speech. While ASR systems have greatly improved in recent years, the biggest hurdle remains the lack of niche speech data. Most available speech data fall into a few voice types and are not representative of the average ASR system user. The majority of machine learning projects require large amounts of data.

However, the adaptability of E2E ASR models allows them to be fine-tuned to outlier speech using small amounts of representative data. The project presented in this thesis aimed to fine-tune an ASR model for use in an Indigenous Counselling Program. An open-source ASR system called Mozilla DeepSpeech was used to train the models used for the project. Representative speech data was gathered from audiobooks and organized into speech corpora to fine-tune DeepSpeech's pre-trained models. DeepSpeech's live transcription software was also implemented on the Unity Development Engine to ensure compatibility with the Indigenous Counselling Program. Three models were trained using the new speech data. Two models were trained using pitch frequency-specific data. One general model was trained using all the new speech data.

The results showed a minimum average relative WER or WER improvement of 8.90% for all the models, on the dataset they were trained on. Furthermore, the general model showed little to no improvement in performance over the pitch specific models when tested on their

trained datasets. This demonstrated the importance of using representative speech data in ASR model training. Overall, the models showed a marked improvement in performance when trained on the intended user accent and voice type.

Acknowledgments

Firstly, I give all thanks and acknowledgment to God, the owner of all knowledge and truth, for teaching me everything I know and for planning lessons for the things I don't know.

Secondly, I would like to thank Dr Zahra Moussavi for her guidance and support in tackling one of the most challenging and exciting projects I have ever done. You have a passion for teaching that is scarce today. Your tireless efforts on behalf of your students are a blessing to all of us and will never be forgotten.

Thank you to Dr. Pradeepa Yahampath and Dr. Dean McNeil for your encouragement and constructive criticism. Your wise advice has been integral throughout my degree and the process of this project.

I would like to thank all my friends and research team for sitting with me through every meeting and class, for teaching me and listening to me, and for providing the support that can only be given by your peers.

A big thank you to my parents, George and Modupe Olaniyanu, and the rest of my family for putting up with me. You have given me a gift no amount of money can buy. Love.

Lastly, I would like to thank God because I can never thank Him enough.

Dedication

To the truth, which is always worth pursuing. Whether beautiful or ugly.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Full Name
ASR	Automatic Speech Recognition
E2E	End-To-End
CTC	Connectionist Temporal Classification
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long-Short-Term Memory
WER	Word Error Rate
CER	Character Error Rate
F0	Fundamental Frequency
MFCCs	Mel-Frequency Cepstral Coefficients
ReLUs	Rectified Linear Activation Units
GPU	Graphic Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory
VR	Virtual Reality
AI	Artificial Intelligence

1.0 Introduction

Automatic Speech Recognition (ASR) systems are programs designed to transcribe or identify spoken language. They have many applications and can be found used in voice recognition, speech transcription, language translation, etc. In the early years of development, traditional ASR systems were designed to parse speech in modules, transcribing speech sounds from phonemes to words and then to sentences. Most modern ASRs are created largely using machine learning and probability theory [1]. This makes an ASR system's performance, like most machine learning algorithms, dependent on the data used to train the system.

To train an ASR system two data components are necessary namely, recorded speech and more importantly a corresponding speech transcript. The accuracy of the speech transcript is integral to training the ASR system. Both components together are called speech corpus. The difficulty in creating these speech corpora is one of the major issues in training an ASR system. While there are many open-source speech corpora, high-performing ASR systems often require significantly large amounts of data to reduce errors to acceptable levels. Unfortunately, the need for human supervision in transcribing training data and the variability in voices make them expensive to create.

When selecting data for any machine learning algorithm, one of the most important aspects to consider is the representativeness of the data. Studies have shown that ASR systems significantly increase in performance when trained with speech data that is representative of their target audience in gender, anticipated noise levels, and accent [2] [3] [4]. This poses unique challenges to ASR system design and modeling as the performance of the system is dependent on its intended use.

2.0 Background

Traditional and Modern ASR systems

Automatic Speech Recognition Systems have experienced large changes in design and performance in recent years. Continuous improvements have been made to the system's design, driven largely by advancements in machine learning. This section of the project background will provide an overview of traditional and modern ASR systems. It will highlight some of the novel ideas that caused the shift from traditional ASR systems to modern ASR systems and the benefits of those ideas.

Speech recognition has always been a complex problem to solve. In particular, the high variability of outputs and inputs made creating a comprehensive solution to the problem challenging. Traditional or hybrid ASR models aimed to solve this problem by breaking it up into more simplistic parts. Each process in transcription, was done separately allowing the system to improved and trained separately. Unfortunately, this made the whole model's performance dependent on its weakest link. It also made the model inflexible to any variability in user speech.

Most traditional ASR models have three main parts, namely, an acoustic model, a pronunciation or lexicon model, and a language model [5]. The acoustic model converted the extracted speech features into phonemes. Phonemes are the smallest units of speech perception [6]. The phonemes are then sent to the pronunciation model, where they are converted into words. Finally, the language models would combine the words into speech transcripts. The largest issue with hybrid ASR models was the requirement for the use of phonemes. Since speech transcripts do not contain phonetic information, phonemes had to be assigned to words using lexicon dictionaries. This often required linguists to manually create new lexicon entries for

every new word. This made training hybrid ASR models difficult to train. It also required that new models had to be trained for different accents, speech types and languages.

Modern or End to End (E2E) ASR models use a more holistic approach to speech recognition. Most E2E models, only use one or two models. In all cases, the lexicon or pronunciation model is no longer present. Instead, E2E models either split the work of the pronunciation model between the language and acoustic model or have one model do the work of all three. E2E models forgo the need for phoneme to word alignments and lexicon dictionaries [5]. This makes the models easier to train and more adaptable to outlier speech types. In exchange E2E models have a larger computational and data cost. E2E models are also more easily adapted to other languages which may have lower amounts of speech data [7].

The first E2E approach that was able to perform speech transcription without the phoneme to word alignments required hybrid models was the Connectionist Temporal Classification (CTC) [5]. CTC does this by assuming an alignment between speech and a transcript. It also allows the repetition of speech labels and uses a blank space to represent sounds with low alignments with available labels. CTC is one of the most popular speech recognition methods due to its ability to handle temporal classification tasks by assuming alignment between the input and output sequences.

Another popular speech recognition approach is the use of Recurrent Neural Networks (RNN) cells like the Long-Short-Term Memory (LSTM) cell [5]. LSTM cells allow the model to use some information from previous speech frames to help predict the output of new speech frame. It is currently the most popular method for designing ASR models. Advancements in

machine learning such as the development of CTC and RNN models have played a significant role in the development of E2E models today.

Performance Metrics of ASR

The performance of ASR systems is evaluated based on two main metrics called the Word Error Rate (WER) and the Character Error Rate (CER). The WER is the proportion of incorrectly transcribed words to the total amount of transcribed words. It is usually represented as a percentage. A lower percentage indicates greater performance and a higher percentage indicates lower performance. The CER is the proportion of incorrectly transcribed characters to the total amount of transcribed characters. In general, WER describes the performance of the ASR systems acoustic model, and CER describes the performance of the language model [8]. A relatively high WER and low CER are often synonymous with a poorly trained language model. A relatively low WER and a high CER are often synonymous with a poorly trained acoustic model. However, improvements in both models will have a positive effect on the WER of the model. As such, WER is the most popular performance metric for an ASR model. The absolute WER is the exact WER a model performed at after training. It is a positive value ranging from 0 to 100%. The relative WER is the difference between the WER before and after training. In other words, it represents an improvement or loss in WER. It can be either positive or negative and ranges from -100% to +100%.

Anatomy and Function of the Larynx and Human Speech Production

The English language alone consists of thousands of words, characters, and, at its base level, phonemes or sounds. The combination of sounds at certain frequencies forms the

phenomes that are associated with vowels, characters, words, etc. The patterns in generating phonemes can be exploited by machine learning models to generate a probability or confidence score and identify which phoneme is being used. A complete understanding of ASR systems requires a working knowledge of how the human voice generates speech.

There are two main ways in which the human voice is controlled, and a corresponding organ controls each. The first way is through harmonics, using the vocal folds or cords. The second way is through formants, using the vocal tract.

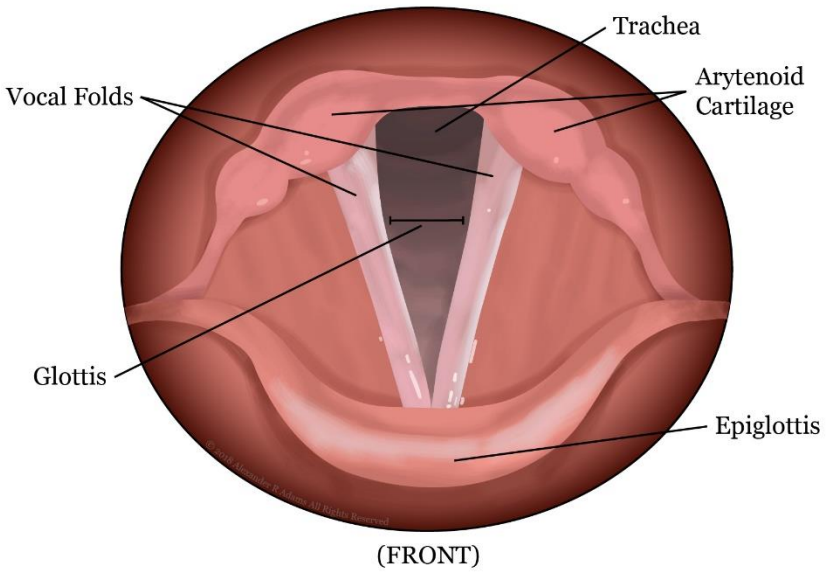

Figure 2.1 Diagram of the vocal folds [9].

Harmonics are generated by the vibration of the vocal cords. A diagram of the vocal cords or vocal folds can be seen in Figure 2.1. Periodic vibrations of the vocal cords will generate harmonics at higher frequencies. The main frequency of the vocal cord's vibration is called the fundamental frequency (F0). The fundamental frequency is controlled by the shape, tension, and length of the vocal folds, which can be manipulated by tightening, opening, or loosening the vocal cords. The tighter and shorter the vocal cords, the higher the frequency of the harmonic generated. F0, which is generated by the vocal cords, is the highest in amplitude and lowest in frequency [10]. F0 is also commonly referred to as pitch. Pitch is the human perception of fundamental frequency [11]. Subsequent higher frequency harmonics will be lower in amplitude. As such, higher harmonics are hardly heard, and the vocal tract is used to generate higher frequency sounds at louder amplitudes.

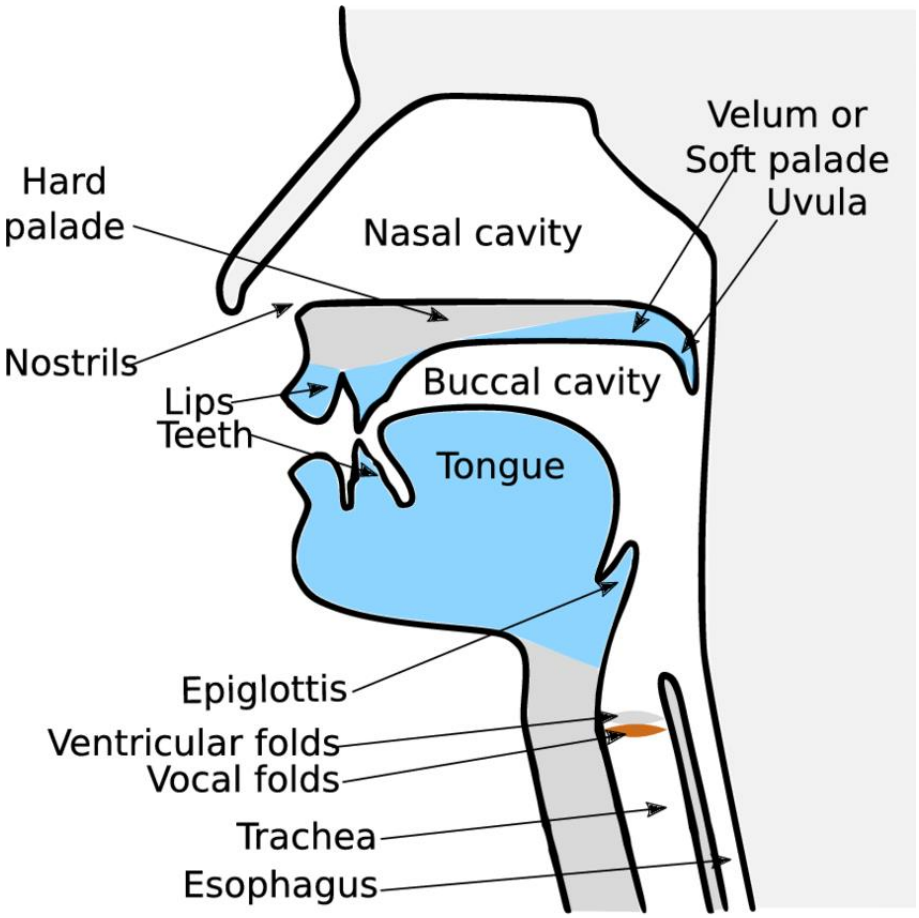

Figure 2.2 Diagram of the vocal tract and surrounding structures [12].

The second structure that controls human speech is the vocal tract. The vocal tract is made up of the oral and nasal cavities, the pharynx, and the larynx. A diagram of the vocal tract can be seen in Figure 2.2. The vocal tract functions as a resonator [13]. Resonators are containers of air that vibrate when they encounter a sound wave that matches their resonant frequency. The pitch at which a resonator vibrates is dependent on its shape, size, density and the size of its opening [10]. Larger resonators vibrate at lower pitches and smaller resonators vibrate at higher pitches. In human speech the vocal tract acts as resonator and is used for coarse control of vocal pitch. Speech is largely driven by the vocal tract due to its comparative size. This is

because the small size of the vocal cords prevents the structure from generating loud sounds at higher frequencies.

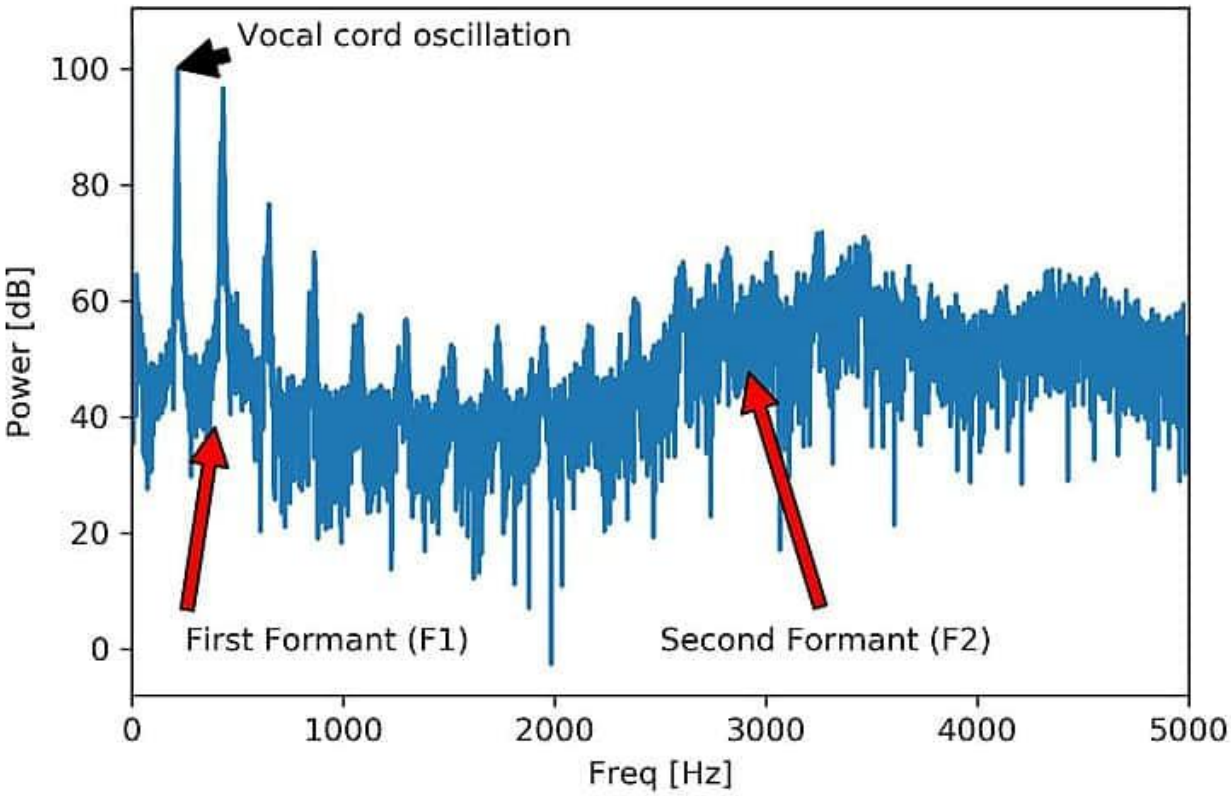

Figure 2.3 Frequency spectrum showing first (F1), second (F2) formants and vocal cord oscillation [14].

Formants are resonance frequencies that are generated due to the vocal tract's unique shape. They are separated by around 1000Hz and the lowest frequency formant (F1) is the loudest. An illustration of the difference between formants and harmonics can be seen in Figure 2.3. Formant frequencies are generated by the specific shape and size of one's vocal tract and can be controlled by modifying the shape of the vocal tract using the tongue, body, and lips [15]. The frequency of one's formants is what allows voices to remain recognizable, even at different pitches. Formants are key to the production of speech. One study shows that vowel sounds in the English language are produced by a combination of three formants [16].

Formants perform a large role in generating accented speech as well [17]. In Figure 2.4 and Figure 2.5, one can see the difference in formant combination for the same letters in the American English accent and the British English accent. This shows the importance of Formants in an ASR models transcription of accented speech. ASR models take advantage of patterns in formants to predict the phoneme being spoken by the user and organize it into readable text.

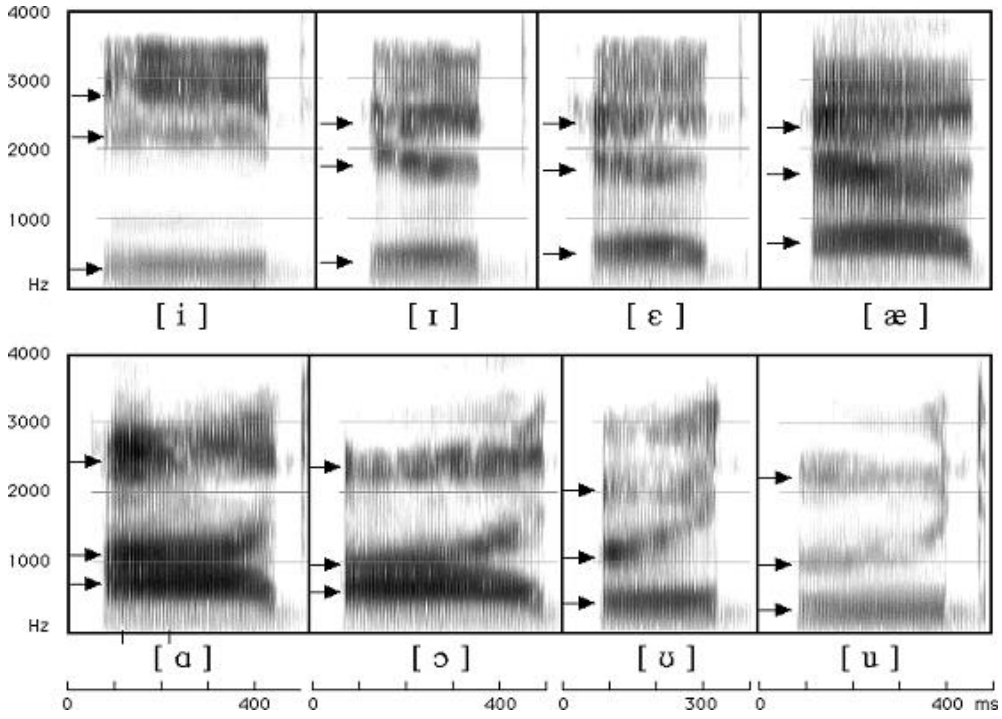

Figure 2.4 Spectrograms of American English Vowels [16].

(The arrows indicate formant frequencies)

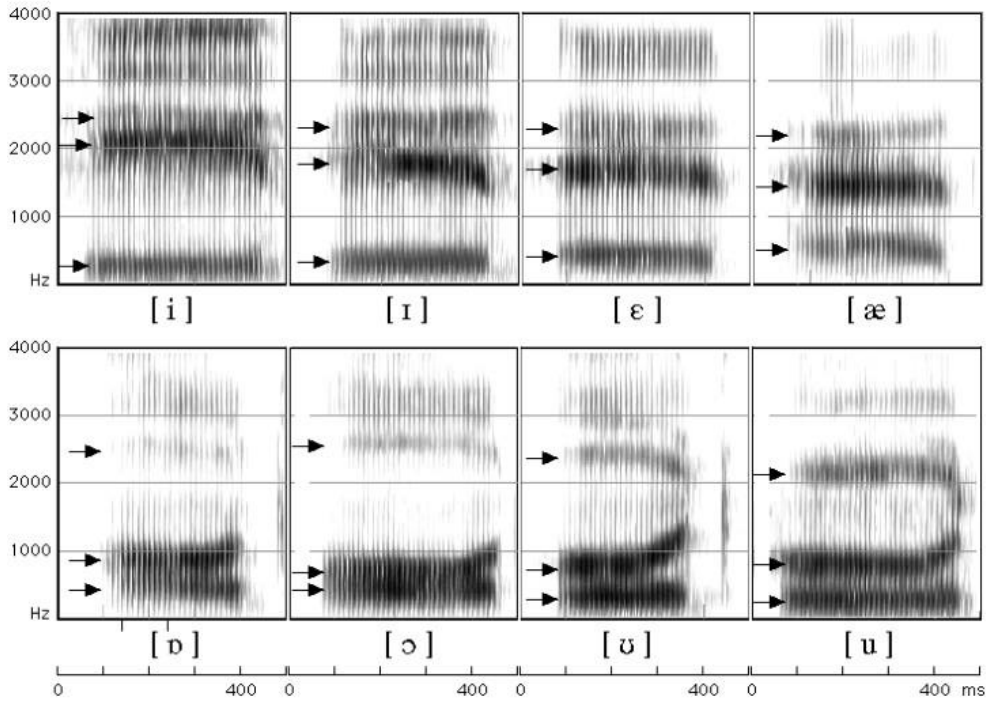

Figure 2.5 Spectrograms of British English Vowels [16].

(The arrows indicate formant frequencies)

While formants and the harmonics generated by the vocal cords vary in function, they are often related. One study investigating the relationship between F0 and formant frequencies found that both were moderately correlated [18]. Another study comparing the formant frequency differences between men and women found that their formant frequencies differed on average by 18-20% [19]. Similarly, the average man's vocal pitch (F0) was found to range from 78 to 182Hz, while the average woman's vocal pitch was found to range from 126 to 307Hz [20]. This points to a vocal pitch-specific correlate in speech that can be exploited in the data selection process of an ASR model's training.

3.0 Methodology

The goal of this project was to train an ASR system for use in a counselling and mental health support program. This project functions as part of a larger one called the Elder Project. The objective of the Elder Project is to design a virtual reality counselling program primarily for First Nation youth. The project has three parts namely, a virtual reality environment for the counselling to take place, an ASR system to transcribe the speech of the user, and a Virtual Elder to interact with the user. The Virtual Environment chosen for the project is the Unity Development engine. The goal of the project was to create an ASR system model for the counselling program that would provide good performance for the speech type of the user. The chosen ASR system has a low training cost, and audiobooks were used to source the speech data for this project. The chosen speech data is representative of the target user's speech type, including but not limited to the target user's accent, vocabulary, and expected environmental noise levels. The chosen ASR system was also adapted to the Unity Development Engine. The following steps were taken in the design process to achieve the project objective.

1. Selecting an Automatic Speech Recognition System:
    - An ASR system was selected based on the following criteria.
        i. Performance of its Pre-trained English Model
        ii. Cost
        iii. Adaptability to Future Training
2. Collecting Speech Data
    - Speech recordings from audiobooks were found or created to further train the ASR system.

- The speech recordings contain speech that is representative of the target user's accent, vocabulary, and other necessary characteristics.

3. Organizing Speech Data:
    - The speech data was organized into speech recordings with a corresponding speech transcript.
    - The recordings and transcript were converted into speech corpora that are compatible with the chosen ASR system (DeepSpeech).
4. Training the Models
    - The pre-trained ASR model was trained with the new speech data.
5. Testing System Performance
    - The fully trained model was tested using speech data that was previously collected and organized.
6. System Implementation
    - The ASR system and fully trained model functions on the Unity Development Engine.
    - The ASR system can work in parallel with the other parts of the main Elder Project.

Selecting an Automatic Speech Recognition System

Currently, there are more than twenty automatic speech recognition systems in the public domain. Each system has its strengths and weaknesses. For this project an ASR system was selected on the following criteria in order of importance: cost, performance, ease of use and the existence of a pre-trained model. DeepSpeech was the chosen ASR system for this project and

excelled in the above criteria. DeepSpeech is an open-source speech recognition engine. Unlike similar ASR systems, which charge per hour for the transcription of speech, DeepSpeech's code and models are released under the Mozilla Public License and are free to use. The fact that DeepSpeech is open-source also aided in meeting the ease-of-use criteria. This is due to the common use of the system by other individuals, leaving a trail of solutions for any issues encountered during the project. When comparing performance, DeepSpeech has a notable WER of 7.06% [21]. Similar ASR systems like Google Cloud Speech, Microsoft Azure Speech Service, and Amazon Transcribe have WER ranging from 12% to 22% [22]. Lastly, DeepSpeech possesses a pre-trained English model, which is also freely available to the public. For these reasons, Mozilla DeepSpeech was chosen as the project's ASR system.

DeepSpeech Architecture

DeepSpeech's ASR system works on two models, namely, the acoustic model and the language model or scorer.

DeepSpeech's acoustic model is created using a recurrent neural network (RNN) as it's core. Recurrent neural networks are neural networks that maintain their memory of past inputs. This allows them to work well for tasks like speech recognition in which future inputs are partially dependent on past inputs. See Figure 3.1 for an illustration of DeepSpeech's acoustic model architecture.

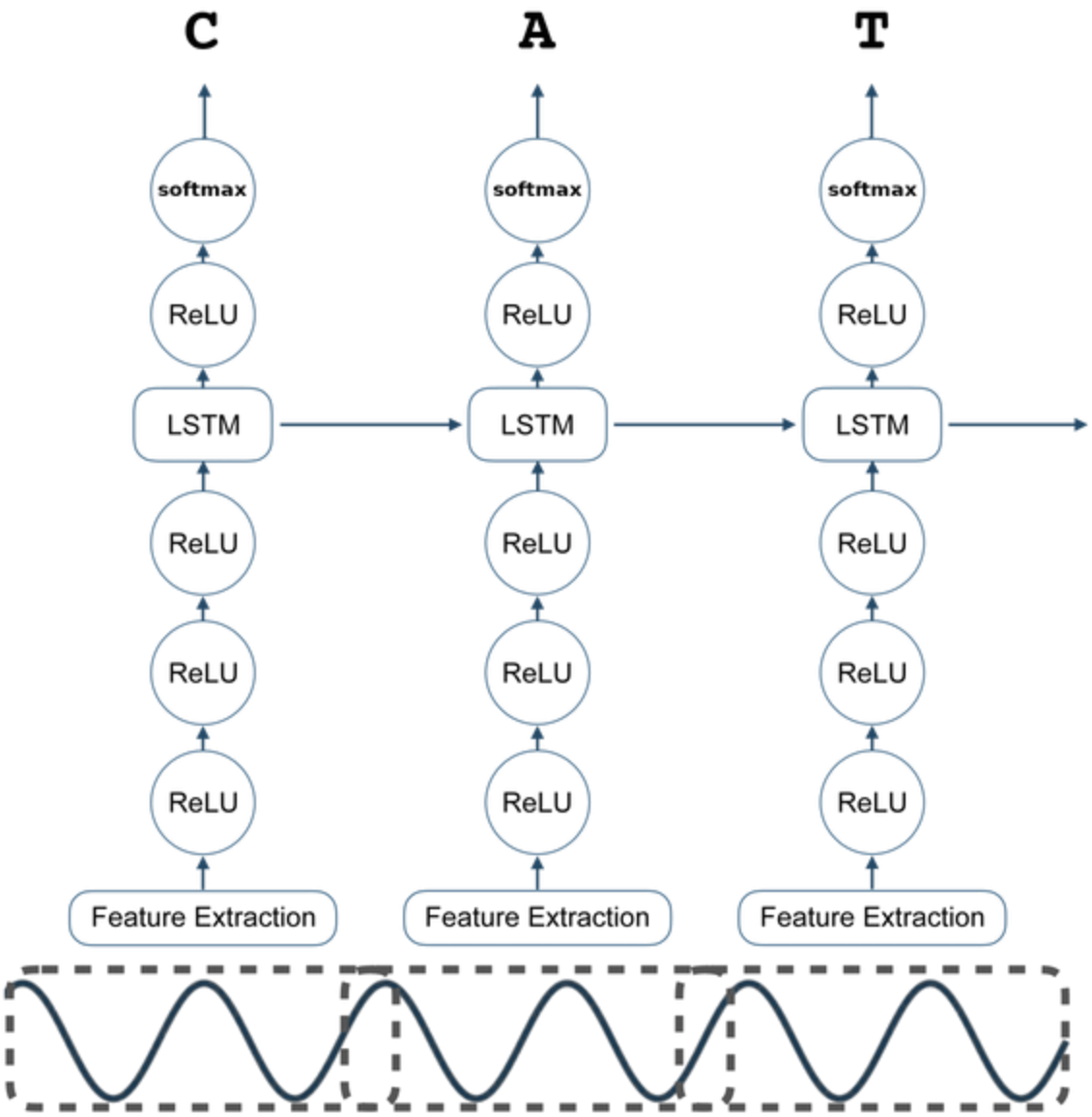

Figure 3.1 Flowchart of DeepSpeech's acoustic model architecture and transcription process [23].

After receiving a segment of audio, it gets sent to the input layer of the acoustic model. The input layer extracts Mel-Frequency Cepstral Coefficients (MFCCs) from the spectrogram of the audio signal. MFCCs are coefficients that represent the short-term power spectrum of an audio signal. The MFCCs are then sent to the first five layers of hidden units of the model. The

first three layers and the fifth layer are non-recurrent units that use the rectified linear activation function. They are called Rectified Linear Activation Units (ReLU), and they only use the weights on which they are trained. The fourth layer uses a Long-Short-Term Memory (LSTM) cell. This is the recurrent part of the network. It alters its output based on the data from previous iterations of the model, giving the model more information to work with when prediction occurs. The last layer, illustrated as softmax in Figure 3.1, is the output layer of the model. This layer is made up of logit functions that assign probabilities to characters based on the inputs to the layer. The most probable character is then sent to the language model or scorer.

DeepSpeech's acoustic model training process uses a stochastic optimization method called Adam [24]. Stochastic optimization involves the use of randomness in finding the minimum or maximum of an objective function. This type of optimization method works especially well on systems that have inherent noise [25]. The method's use of randomness can also help the system find the global optimum more easily and prevent the system from getting stuck at a local optimum [26]. In the case of ASR training, the goal of the optimization algorithm is to minimize the loss function of the ASR model. Loss refers to the error in the system's prediction. DeepSpeech uses the Connectionist Temporal Classification (CTC) loss function to calculate the model's prediction error [23]. CTC works well in machine learning problems like speech recognition, where temporal differences in inputs are common. For example, if the user sounded out the letter "a" for several seconds, CTC would help the model recognize that the letter "a" is not an inaccurate transcription [27].

DeepSpeech's language model is an n-gram model that predicts the next word or word sequences based on the previous sequence of words [28] [29]. The language model does not play any role in the training and validation process. It is used to help structure the output of the

acoustic model and to provide context clarification [28]. For example, the language model would help the system decide between the words "red" and “read” when the user says, "I read a book."

This project focuses on training DeepSpeech’s acoustic model due to the model’s overall importance in speech recognition as well as when dealing with different accents and vocal pitches.

Collecting Speech Data

There are two main approaches to training an End-to-End ASR system. The first and more costly approach is to train the system for robustness. The goal of creating a robust ASR system is to achieve equal WER across a wide range of speech types with differing genders, accents, noise levels, etc. Training a robust ASR system thus requires a large and diverse speech data set [30]. This necessitates a balanced speech data set where every necessary variable is equally represented in the data. The speech data must also be balanced, equally representing each speech type to prevent inequalities in performance. The larger the range of voice types accounted for in training, the lower the overall performance of the system. Robustness is preferred when designing a system for general use where variability in the system user is expected. The drawbacks of training a robust ASR system are the expense of data acquisition and the loss of voice-specific performance by the dilution of speech data.

The second approach is to design the system for specificity. The purpose of an ASR system designed for specificity is to achieve the best performance within a narrow band of voice types. Less speech data is required in comparison to a robust system, and the system is much cheaper to

design. This results in poor performance when transcribing accents and noise levels outside the training speech data. The effectiveness of this approach relies on how well the target voice type can be identified and represented in the speech data set [2]. This approach to designing ASR systems has become increasingly popular due to the high cost of training robust ASR systems. A popular method that follows this approach is fine-tuning. Fine-tuning involves training a pre-trained robust ASR model with a speech data set containing the target speech type [31]. This results in a model that has the best WER for the target speech type while maintaining passable performance for speech types outside the target range.

For this project, a pre-trained model was fine-tuned to best suit our target speech type and environment. The speech data that will be used to train the ASR system will be acquired from audiobooks. Audiobooks provide a significant amount of speech data in a form that is more easily converted into a usable speech corpus. Selecting speech data that is representative of the intended user is integral to the model's performance. The audiobooks were selected based on the following speech and signal parameters:

Noise levels

The noise levels of any signal are integral to its observability. High environmental noise levels in speech lead to speech distortion and, in severe cases, speech misinterpretation. To properly account for noise in the system the environmental noise level must be evaluated. The user is assumed to be using the Elder Project in a private environment with low noise. Studies have shown that simulating expected noise in training data can improve the performance of an ASR system in transcription [32] [33]. For this project no noise was added to the speech data. This is

due to the relatively high signal-to-noise ratio expected in the user environment and the resistance that DeepSpeech possesses to noise [1].

Vocabulary

While the vocabulary of Indigenous and non-Indigenous speakers does not vary significantly, it is important to note that there are several cultural terms used by Indigenous speakers that are not commonly used by non-Indigenous speakers. Such terms may be missing or inaccurately pronounced in general English-trained speech models. Some examples of these terms include the names of Indigenous tribes, common cultural events, and colloquial words. To minimize errors due to poor vocabulary, the speech data that contained recordings of such terms was selected. Audiobooks that contain information about Indigenous culture and traditions were selected to familiarize the ASR system's acoustic model with such terms.

Accent

The difference in accent used to train the ASR poses the most significant challenge to successful transcription. While other causes such as vocabulary and noise can lead to isolated errors in transcription, differences in accent will lead to errors that persist throughout speech transcription. The issue is further exacerbated by the lack of speech data for niche accents. There are upwards of 160 accents and dialects in the world while the best publicly available ASR systems, with the largest speech data sets, have only 6 accents accounted for in their speech data [34] [4]. Studies show that differences in training and user accents can lead to significant increases in WER and poorer overall system performance [35]. To combat this the new speech data was selected from audiobooks with Indigenous narrators with accents. Due to a lack of measuring

methods for accent thickness and identification the narrators were chosen based on their Indigenous Identity.

Gender and Pitch Frequency

Studies show that differences in the gender used to train the system and the user's gender result in increased transcription error [35]. While there may be other differences in speech that correlate with gender, one of the main differences is in pitch frequency [2] [36]. Using speech data with representative pitch frequency during ASR training has also been shown to significantly reduce WER [37] [38]. One approach to resolving errors due to gender differences would be to include equal training for all pitch frequencies in the human range of speech. This would account for differences in pitch frequency and any other speech differences that are correlated with gender. For this project, the speech data was organized by pitch frequency into groups. Each pitch frequency group contained speech data that is within a range of frequencies. There are two pitch frequency groups namely the low pitch frequency group and the high pitch frequency group. Each pitch data set was used to train a pitch frequency-specific model. The low pitch model was trained on the low pitch data set and the high pitch model was trained the high pitch data set. See Appendix A for the pitch frequency groups and their corresponding pitch ranges. The number of groups and the range of frequencies was based on the variation of speech data that is acquired. A general model was also trained with all the speech data regardless of pitch frequency.

Audiobooks were selected on the above criteria. See Appendix A for the list of audiobooks, their corresponding narrators and their separation into pitch frequency specific data sets.

Organizing Speech Data

Data Transformation Process

Speech data is a necessary component to any ASR model's training. The data has two parts namely, recorded speech and corresponding text. These individual pieces of data must be organized and combined into a speech corpus. The speech corpus itself is made up of two data types. A csv file and all the segmented audio wav files. The audio files contain speech audio that corresponds to one sample or row in the csv file. The csv file contains information about each sample of speech data including the transcript of the sample, the name of its corresponding speech audio file and the size of the audio file. Figure 3.2 shows an example of a speech corpus' csv file.

	A	B	C	D	E	F
1	wav_filename	wav_filesize	transcript			
2	CFABD_N1-0.wav	56320	lena stopp			
3	CFABD_N1-1.wav	122880	nineteen seventy six			
4	CFABD_N1-2.wav	314880	i always told turtle when i was raising her if a man acts like a			
5	CFABD_N1-3.wav	360960	she hardly ever listened to me mostly she would make a sour			
6	CFABD_N1-4.wav	183038	turtle wasnt much for talking but her emotions barked like a b			
7	CFABD_N1-5.wav	290562	and sometimes id say things like just because you look more			
8	CFABD_N1-6.wav	67840	id throw back my head and laugh a good one			
9	CFABD_N1-7.wav	171520	tla mostly i liked to tease her about everardo			

Figure 3.2 Image of the required speech corpus csv formatting

For this project 6 audiobooks and eBooks were converted to into speech corpora. The eBooks were organized into text files containing one chapter's worth of text. The text was arranged into paragraphs and synced with the corresponding audio using a python module called

audiosegment [39]. The module uses the offsets and onsets of speech to map each audio file to corresponding text in syncmap file [40]. See Figure 3.3 for a picture of matched audio and text in sync map file.

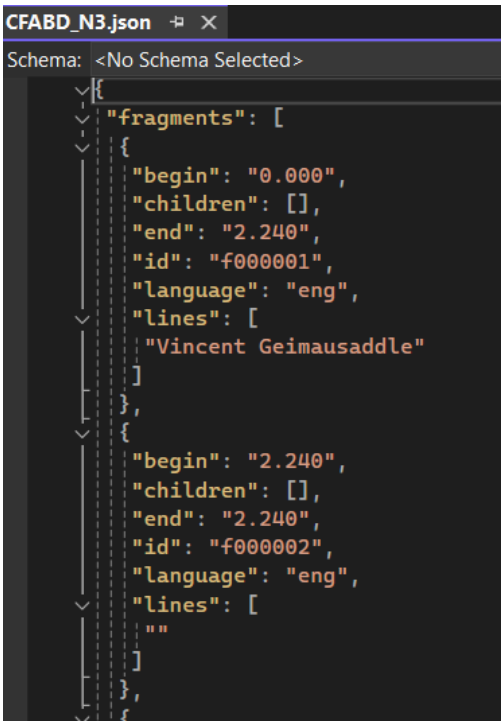

Figure 3.3 Image of 2 segments of matched audio and text in a sync map file

The syncmap file is then used to generate the speech corpus' csv file and to split the original audio file into corresponding segments of speech audio. Due to the use of onset and offset when splitting segmenting the data the speech corpus may still contain samples of data with empty transcripts and characters that are not recognised by DeepSpeech. These errors were removed to make the speech corpus compatible with DeepSpeech. The audio files were required to be in wav file format have a constant sampling rate of 16kHz and use a mono-channel. The

transcripts could not contain any punctuation and had to have all numbers written in cardinal form [41]. Both the audio and transcripts were modified to meet all the above criteria and to ensure that the speech corpus was compatible with DeepSpeech's training software.

Pitch Frequency Classification

The vocal pitch of one's voice is dictated by the anatomy of their vocal cords and tract. The semi-periodic vibration generated by the movement of air through the vocal cords has a fundamental frequency, F0. Vocal pitch is defined by the human perception of the fundamental frequency of an individual's voice. In this project the value of F0 was used to judge the vocal pitch of the audiobook narrators. The value of F0 was calculated using a weighted average of fundamental frequencies generated from a 15 to 30 minutes of audio. See Figure 3.4 for a formula for the weighted average.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i \cdot w_i}{\sum_{i=1}^{n} w_i}$$

$$\bar{x} = \frac{(x_1 \cdot w_1) + (x_2 \cdot w_2) + \ldots + (x_n \cdot w_n)}{w_1 + w_2 + \ldots + w_n}$$

Figure 3.4 Vector formula for weighted average [42].

The fundamental frequencies were calculated using machine learning for pitch estimation called CREPE [43]. CREPE is a pitch tracking algorithm that is based on a convolutional neural network (CNN). The algorithm was trained on male and female voices and can track pitch consistently with a over 90% accuracy [43]. The algorithm provided a fundamental frequency and its corresponding confidence score for each second of audio. The confidence score was used as weight for the average.

The audiobook narrators were split into two groups based on their derived vocal pitch namely, low pitch and high pitch group. The low pitch group corresponded to the narrators with lower vocal pitch frequencies. The derived vocal pitch of the low pitch group ranged from 133Hz to 173Hz. The high pitch group corresponded to the narrators with higher vocal pitch frequencies. The derived vocal pitch of the high pitch group ranged from 186Hz to 235Hz. See Appendix A for a list of the selected narrators, their groupings and the corresponding vocal pitch estimates.

Training the Models

Hardware and Software Requirements

Hardware Requirements

One of the limitations to training machine learning models is computational cost. Depending on the speed of training and the complexity of the model, training machine learning models can often require high computational power. While it is possible to use CPUs to train machine learning models, GPUs have become a staple in machine learning model training

because of their high computational power. GPUs specialization in performing large amounts of mathematical computation are well suited to execute the large-scale matrix calculations that machine learning training requires [44]. GPUs also perform parallel processing which allows for multiple calculations to be done simultaneously. The computational power available during model training often varies even among GPUs and is a major limitation in terms of overall model performance. Due to the poor performance of CPUs, DeepSpeech's training software requires the use of GPUs to function. For this project, all the models were trained using a NVIDA GeForce RTX 3080 Laptop GPU and an Intel® UHD Graphics GPU. Both GPUs were used simultaneously through the of a popular machine learning software called TensorFlow. TensorFlow and other software requirements will be discussed in detail in the following section of this report. See Table xx for an overview of the hardware used for this project.

Table 3.1 Laptop Hardware Specifications

Hardware Component	**Component Used**
Processor	11TH GEN Intel® Core™ i7
GPUs	Nvidia geforce rtx 3080
	Intel ® UHD Graphics
Original OS	Windows 10 Home
Total Installed RAM	32 GB
System	MSI
System Model	GP66 Leopard 11UH
System Type	X64 Based PC

Software Requirements

DeepSpeech's training software the following major software requirements:

Table 3.2 Major software dependencies for DeepSpeech model training

Software Requirements	**Version Required**	**Version Used**
Python	3.6	3.8
Operating System	Mac or Linux	Ubuntu (Virtual Linux Env)
CUDA	10.0	12.5
cuDNN	v7.6	V8.93
NVIDA Driver	≥ 411.31	555.9
DeepSpeech	v0.9.3	0.9.3
virtualenv	N/A	N/A
Docker	N/A	26.01
TensorFlow	1.15.4 (GPU version)	NVIDIA TensorFlow 22.02 (1.15.5)

Training Split and Hyperparameters

Training, Validation and test split

When training a machine learning model, it is important to divide the available training data into different data sets. This allows for unbiased training. DeepSpeech uses three types of data sets to train its ASR models'; namely the training set, the validation set and the test. The training

data set contains the data in which the machine learning model is trained on. It is the part of the data set that directly changes the model and is often the largest part of the data split. The validation set functions as a part of a mechanism to corroborate the accuracy of the model after each round of training. After each round of training the validation set is used to test the progress of the model. This evaluation can then be used to alter hyperparameters for the next round of training or even halt training to prevent model overfitting. After each validation the current model would be compared to the previous best model. The new best model will then be saved for future comparison. This helps the system to make consistent progress in training the model. Lastly, the test set is the data set that is used to evaluate the performance of the model after the training is completed. The information provided by the final test is integral when comparing the performance of different hyperparameter combinations. A flowchart of the model training process can be seen in Figure 3.5.

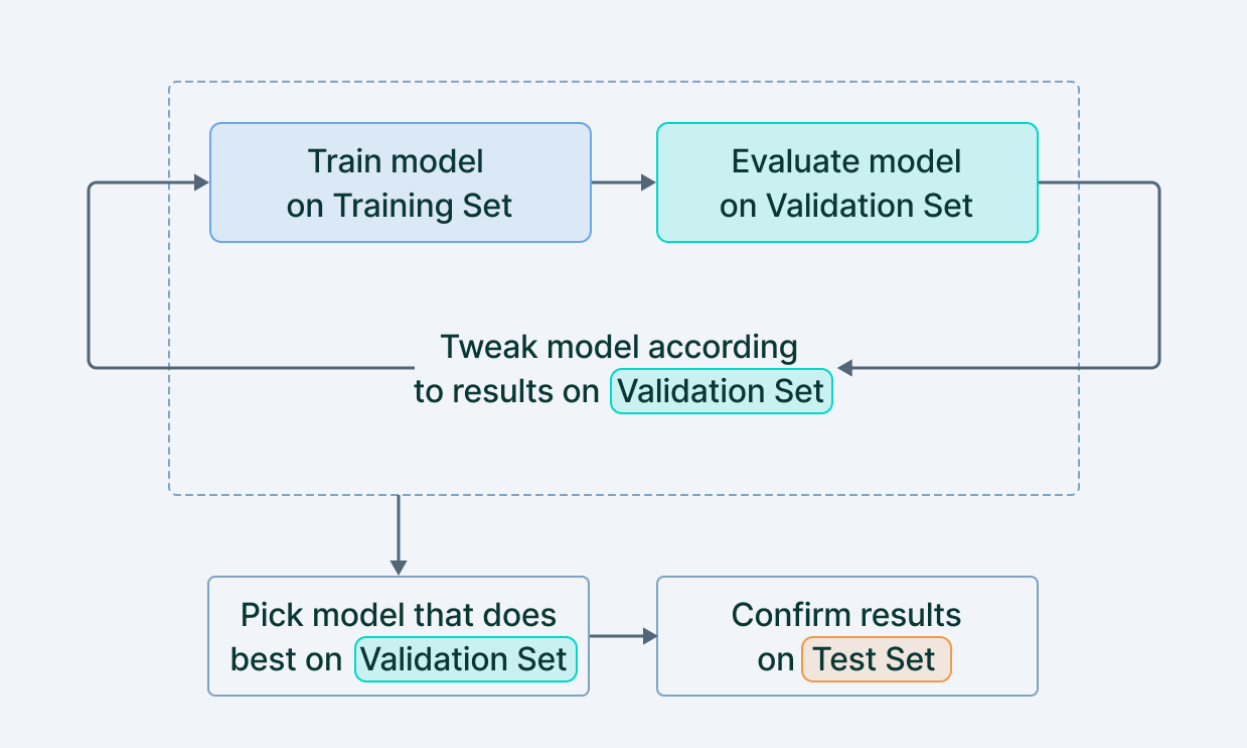

Figure 3.5 Flowchart of training process and the role of the Training, Validation and Test set [45].

The distribution of the data into each set is integral to the performance of the model. The key trade off is between high statistical variance in training or in testing. Less training data results in higher statistical variance in training while less test and validation data results in higher variance when evaluating the model [45].

One study that compared the performance of different data split ratios found that the size of the dataset determines the optimal dataset distribution. Specifically, the study showed that smaller datasets performed best with splits like 70/30 (train/test), while larger datasets benefited from using increasingly smaller test and validation sets [46]. Another study that investigated the effect of data splitting on machine learning model performance found that the 70/30 split was the ideal split for the model's performance [47]. For this project, 70% of the data was used for training,

20% of the data was used for validation and 10% of the data was used for testing. An illustration of the chosen train-dev-test split can be seen in Figure 3.6.

It is important to maintain the training, validation and test split ratio if subsequent models to remove any bias from each model's performance measurement. While the nature of speech data makes it difficult to maintain the train-dev-test split ratio, efforts should be made to minimize systemized bias during training. For this project the data for each iteration of training was distributed while minimizing the difference between each data split.

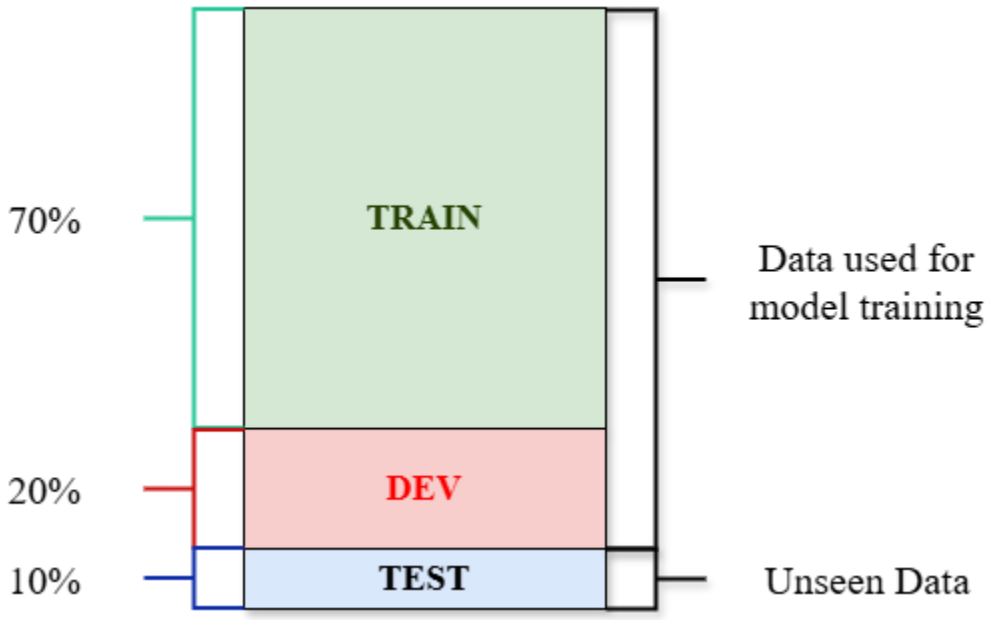

Figure 3.6 Illustration of the selected Train-Dev-Test Split for all models

Another important component of dataset splitting is the chosen data splitting technique. There are three main data splitting techniques namely, random sampling, stratified sampling and cross-validation.

Random sampling involves shuffling data and randomly selecting samples for the training, validation or test set. After a sample is assigned to one set it cannot be used in another set. The distribution of samples is done in line with the chosen split ratio. Random sampling is the most widely used data splitting technique and has been shown to work well with class-balanced datasets [48] [49]. All publicly available models created by DeepSpeech use random sampling in their training process.

In imbalanced datasets there are unequal proportions of data classes in the training, validation and test sets. This can result in unintended bias being added to the model [50]. This problem is resolved by using stratified sampling [49]. Stratified sampling involves distributing samples to each set in equal proportions to ensure that each data class is accounted for in each set [51]. This is samples are split while maintaining the chosen split ratio. An illustration of stratified sampling can be seen in Figure 3.7. The goal is to maintain the original ratio of classes throughout the training, validation and testing process.

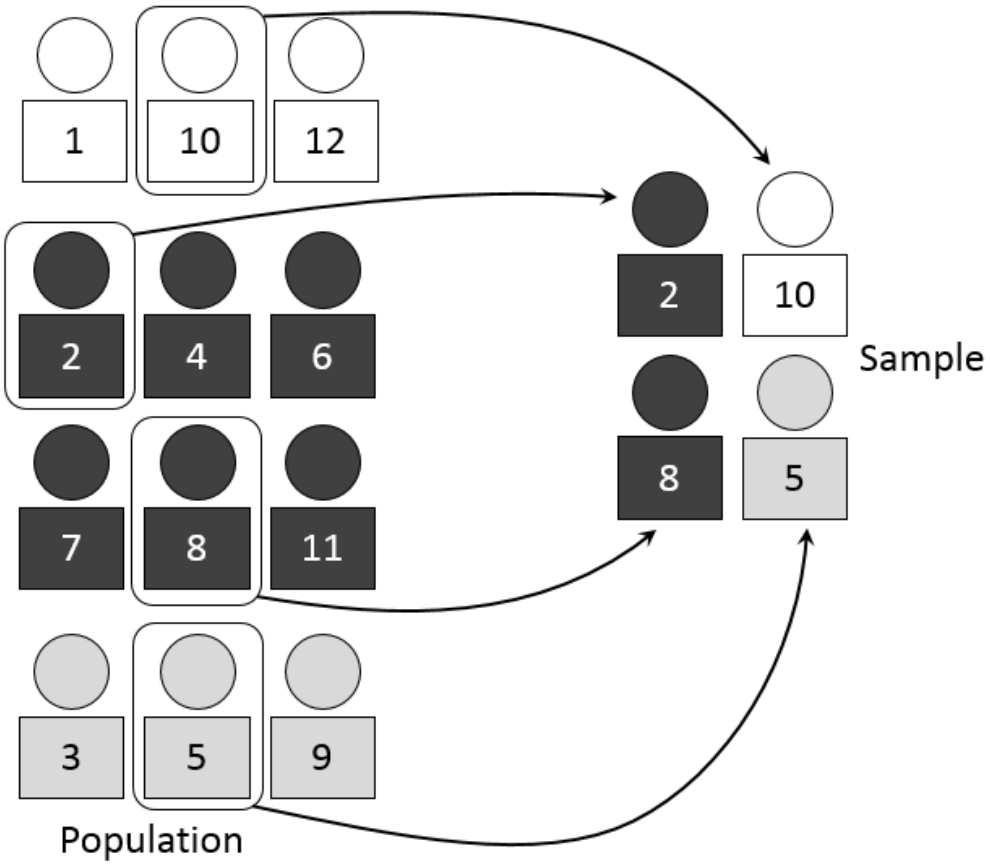

Figure 3.7 Illustration of selecting one sample using the stratified sampling technique [52].

Another data splitting technique is cross validation or specifically k-fold cross validation. Cross-validation involves running the training process through multiple iterations. Each iteration has a different combination of samples in the training, validation or test set. This prevents overestimation or underestimation of the model's performance due to statistical bias. K-fold cross-validation involves dividing the data into k number of equally sized folds. The folds are then shuffled and randomly distributed into training, validation and test sets according to the chosen split ration [53]. An illustration of k-fold cross-validation can be seen in Figure 3.8. This allows for a more accurate analysis of the model's performance. Cross validation is more commonly used for machine learning models that focus on classifying samples into small number of groups.

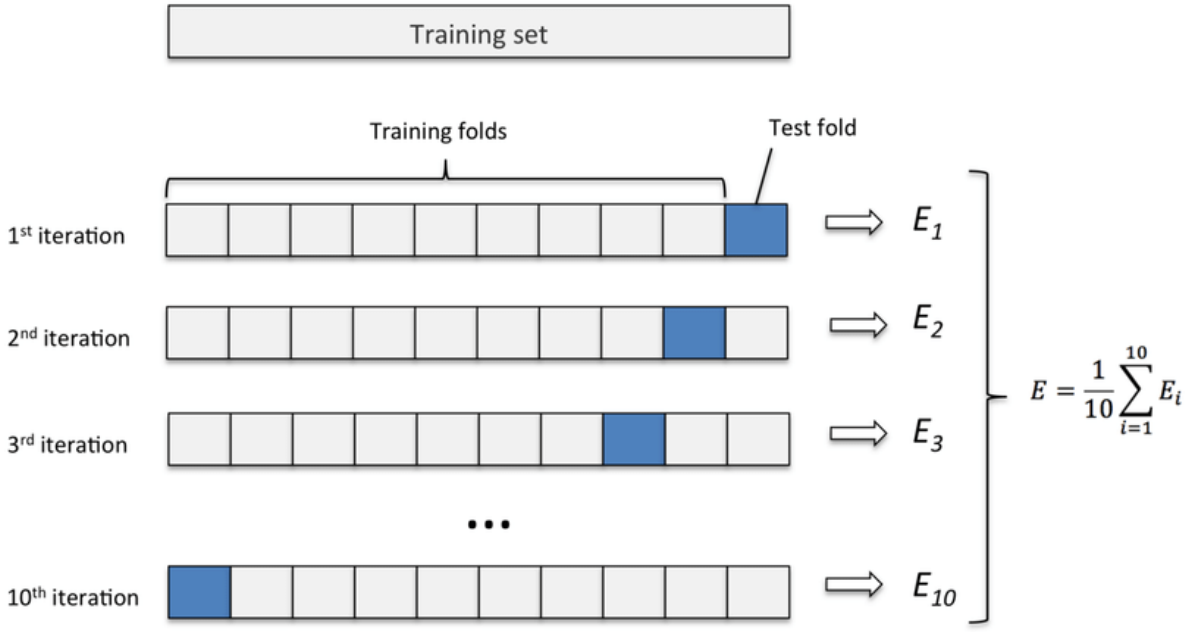

Figure 3.8 Illustration of k-fold cross validation technique [54].

Most publicly available ASR systems are trained and tested using the random sampling technique [1] [55] [56]. This is due to the complexity of stratifying speech data which contain many potential classes. Despite the difficulties of using other sampling techniques, there are benefits to incorporating other sampling techniques into the training and testing process of the model. One study comparing stratified cross validation with cross-validation found that stratified cross validation resulted in less variance and bias [57]. This project's model used a combination of both stratified and cross-validation techniques. The datasets were based on the vocal pitch in the general model's training, validation and test set. The proportions of low and high-pitched speech were maintained for the training, validation and test set of the general model. They were also stratified based on the audiobooks. This means that the proportions of speech data for the

audiobooks were maintained in the training, validation and test set. The proportions were also maintained across each cross-validation iteration. Three iterations were conducted in which the test data was randomly selected while maintaining the vocal pitch, audiobook and data split proportions.

Hidden layers

Another important hyperparameter in the training process is the number of hidden layers used in the neural network. Most neural networks are divided into three parts or layers namely, the input layer, the hidden layer(s) and the output layer. The input layer receives the data as it is and sends it to the hidden layer(s). The hidden layer then converts the data into a group of features that the output layer can then use to predict the results. The number of hidden layers used in a model indicates the level of complexity the model can predict accurately. The higher the number of hidden layers the better the model. Higher numbers of hidden layers are also associated with more computational cost during training. All the fine-tuned models used in this project had 2048 hidden layers, the same number of hidden layers as DeepSpeech's pre-trained English model. See Appendix B for a list of hyperparameters. This allowed the model to carry on what it had learned from past training experiences.

Steps, Batch sizes and Epochs

An epoch represents one round of model training with the chosen training set. The main trade-off regarding the number of epochs is between the model's performance and potential overfitting. For each epoch that the model is trained, it becomes more attuned to the data set. When balanced with a good learning rate this will maximize the benefits of the training data, but if over trained this can lead to overfitting. Overfitting occurs when a model can no longer be accurately

used to classify data that is outside the training data [58]. In the case of the ASR model being used in this project, overfitting would occur if the model would lose the ability to transcribe that is not contained in the speech corpus. However, if not enough epochs are run there is a potential for underfitting. This occurs when the model is not sufficiently trained enough to maximize the performance of the model. A balance must be struck to maximize the benefits of training without losing versatility in the model. A higher number of epochs also increases the computational cost of training. For this project, a regularization technique called early stopping was used to select the number of epochs for training. See the Early Stopping below for more information on the process. The exact epoch size for each training session is outlined in Appendix B.

The next variable to consider when training a model is the batch size. Batch size represents the number of samples that are used in one unit or step of training, validation or testing. A sample represents a single data entry in a speech corpus containing one audio file and its corresponding text. Together with the number of epochs, batch size determines the total number of steps in one training, validation or testing session. For example, a training session with a batch size of 4 and an epoch number of 10 would have a total of 40 steps. After each step of training, the error between the model's predicted results and the real result is used to modify the model's internal parameters. This process is called Gradient Descent and is commonly used to optimize the cost/loss function in deep learning models [59].

The main trade-off to consider when choosing batch size is between training speed and the model's generalization error. When a model is predicting an outcome during training a smaller batch size will result in noisy error estimates, which helps regularize the model. However, this slows down the learning speed of the model. Larger batch sizes result in more accurate error estimates and increased generalization error in the completed model. However, the accurate error

estimates lead to faster model convergence and learning speed. In general, studies show that smaller batch sizes are better for model performance [60] [61]. A batch size of 32 has been shown to make the most of the benefits of smaller batch sizes while minimizing training time [61]. However, batch sizes are also limited by the available memory of the system being used to train the model. The larger the batch size the more memory is required to temporarily store the batch during one step of training. For this project a batch size of 12 was used in all iterations of each training, validation and testing session. This mainly due to the limitations of the memory on the system being used to train the models.

Early Stopping

One way to ensure an appropriate balance between over training and under training is a popular regularization method called early stopping. Early stopping involves the interruption of model training before overfitting can occur. After every training session in an epoch, the model is tested on the chosen validation dataset. This validation session is used to calculate different performance metrics of the trained model. One of those metrics is the loss or cost of the model. The model's loss or cost is an estimate of the model's inaccuracy or error. During the training process the prediction error of the model on the validation dataset decreases. Overtraining is often exemplified by an increase in the error the model [62]. Often overtraining begins before the model's error on the validation has even begun to increase. The purpose of early stopping is to halt training before this can occur. Model training systems that use early stopping have different ways for detecting overfitting and triggering the halting of the training process. DeepSpeech's training software compares the calculated loss or cost and stops the training process when the change in loss between a chosen number of epochs does not exceed a minimum delta [63].

In DeepSpeech, there are three variables that dictate the use of early stopping namely, epoch, es_epoch and es_min_delta. The variable epoch represents the maximum number of epochs that will be run even if the model fails to meet the criteria for an early stop. For this project, all iterations used 30 for the maximum threshold for training. None of the models reached this threshold before an early stop occurred. The second variable called es_epoch sets the number of epochs that should occur before the criteria for an early stop is checked. An es_epoch value of 4 would mean that an early stop could only occur every 4 epochs, starting from the first epoch. For this project an es_epoch value of 2 was selected. This was based on the learning speed of the models. The loss or cost significantly reduced between every 2 epochs. Finally, the es_min_delta represents the minimum required improvement in the model's loss to prevent the early stop from triggering. For this project, a minimum reduction in loss of 0.06 was required for training process to continue unhindered.

Dropout and Dropout Rate

Another way of preventing overfitting is by using a regularization method called dropout. Dropout involves randomly removing nodes from the model during training. Understanding why the dropout method works requires a deeper understanding of how neural networks work. Neural networks are made up of nodes, each made up of weights and equations. These nodes and weights largely define the neural network and its function. One of the signs of a model or neural network that is overfit are high magnitude weights. The dropout method combats this by temporarily blocking access to random nodes within the layer during training [64]. An illustration of the dropout method can be seen in Figure 3.9. The hyperparameter used to control the probability of a node getting removed is called the dropout rate. The parameter represents a probability and as such ranges from 0 to 1. Where a dropout rate of 0 guarantees that no nodes will be dropped, a

dropout rate of 1 guarantees that all nodes will be dropped and 0.5 gives each node a 50% probability of being dropped.

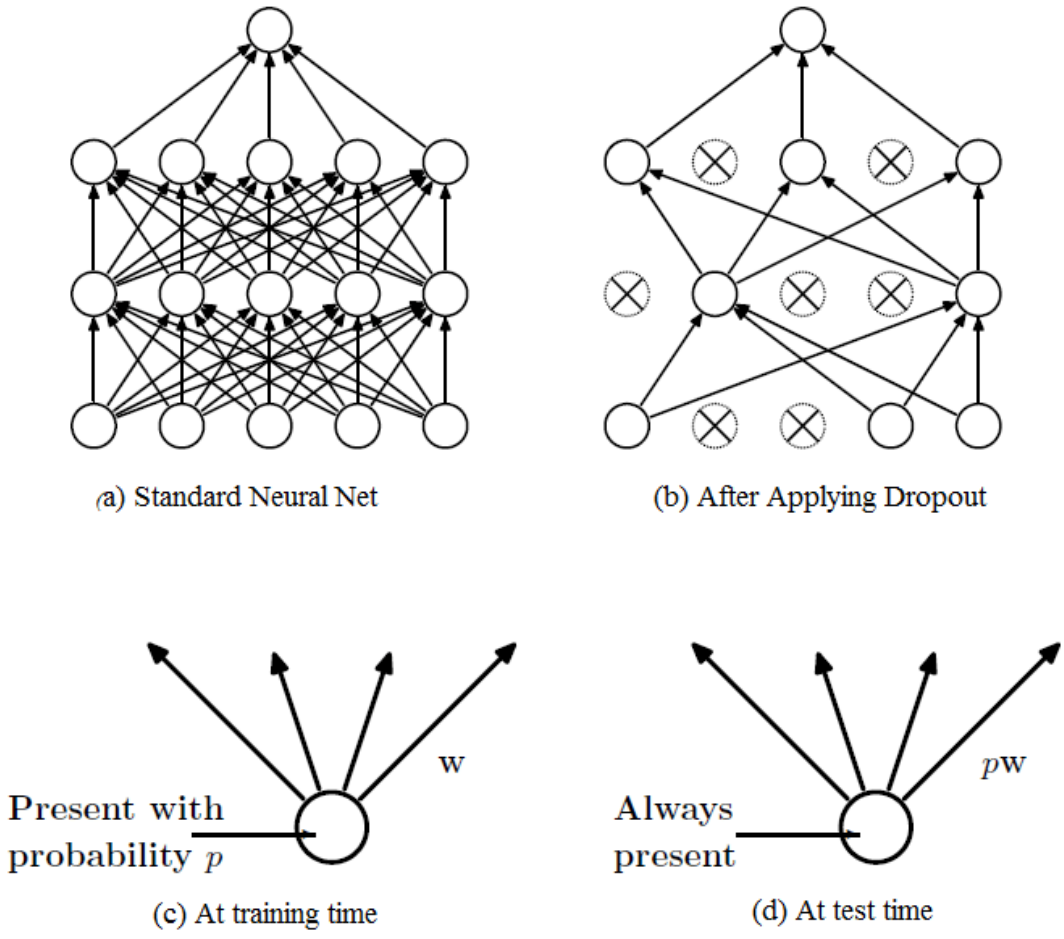

Figure 3.9 Illustration of the Dropout regularization method [64].

(a) Neural network without dropout. (b) Neural network during train. (c) Node during training with dropout. (d) Node during testing after dropout scaling.

Due to the absence of some nodes during the training the weights of the model will become larger. This is offset by scaling the weights by the dropout rate at the end of the training process.

The size of the available training data is the predominant factor in deciding the dropout rate when training a model. The higher the dropout rate the higher the model's tolerance to overfitting. This results in lower dropout rates being used for training sessions with large amounts training data [63]. DeepSpeech's training software uses a default dropout rate of 0.05 [63]. For reference most updates to their English model use speech data in the order of thousands. For this project a dropout rate of 0.4 was used to minimize overfitting. This is due to the comparative size of the training data used in this project to the size of the data already used to train the pre-trained English model.

Learning Rate

Learning rate is a hyperparameter that controls the speed of the model's adaptability during training. Learning rate is one of the most important hyperparameter driving model training. The learning rate ranges from 0 to 1. Where 1 is the highest learning rate and 0 results in no learning. The main trade-off to consider when deciding on a learning rate is between the speed of training and the model's stability [65]. A learning rate that is too small will converge too slowly, requiring a high number of epochs to increase the model's performance. Conversely, a learning rate that is too high will result in model that converges too quickly and has unstable results. A balanced learning rate that allows for the steady improvement of the model is integral to the training process. Unfortunately, it is not usually possible to calculate the best learning rate for training [66]. As such trial and error is often necessary to determine the best learning rate for training. For this project, a learning rate of 0.0001 was used for all iterations of training. This value was suggested by the developers of DeepSpeech [63]. It is based on their own experiments fine-tuning models using the DeepSpeech training software.

Testing Methodology

Training Hyperparameters

Hyperparameters are variables that are used to control the training process of a machine learning model. They have a large impact on the performance of the models the system trains and need to be carefully chosen to get the best results. Hyperparameter tuning is the process of selecting the ideal hyperparameters for training a particular model. It is often used in machine learning training to optimize the performance of the trained models. For this project, many of the hyperparameters were selected based on the size of the speech data, hardware availability and suggestions made by the developers of DeepSpeech.

However, the value of the batch size for the training, validation and testing sessions was determined through trial and error. Batch sizes of 1, 4, 12, 14, 16 were tested and were found to have little effect on the training process. The final batch size of 12 was chosen due to the limitations of the hardware used for training and the reduction of generalization error that is common when training with small batch sizes. For each iteration of training all hyperparameters where kept constant to prevent the introduction of bias into the system. The general model and the pitch frequency specific models were tested using speech data sets in each frequency group.

System Implementation

The Elder Project consists of a VR Environment, an Automatic Speech Recognition system and an AI Response system. The project is being developed on the Unity Development Engine and as such every part of the project must be able to function on it. Specifically, the ASR system must be able to provide live speech transcription for the user.

While DeepSpeech does have live speech transcription software, its speech transcription is incompatible with the Elder Project. Specifically, DeepSpeech's system has automatic and uninterruptable transcription. While this allows for constant speech transcription, it allows for room for noise and unintended transcription. DeepSpeech cannot process speech data while listening for new speech. DeepSpeech's live transcription software was also developed using Python programming language which is incompatible with Unity which uses C++.

This project used an open-source Unity implementation of DeepSpeech called deep-speech-unity [67]. The open-source software was modified to use a push to talk option and display the transcribed text on the screen. The push to talk button gives the user more control over the ASR system and prevents unintended transcription. A picture of the DeepSpeech on Unity implementation's start screen before and after speech transcription can be seen in Figure 3.10 and Figure 3.11 respectively.

Figure 3.10 DeepSpeech and Unity Implementation (Start Screen)

Figure 3.11 DeepSpeech and Unity Implementation (Test)

(The sentence “this a test” was said while holding the P button) using a laptop microphone)

Summary

In Summary, all the objectives of the project were completed. An open-source ASR system with a pre-trained English model, called Mozilla DeepSpeech was selected for this project. Representative speech data from audiobooks were selected and organized into speech corpora for model training. Three models were trained for the project. The low pitch model was trained on the low pitch speech data, the high pitch model was trained on the high pitch speech data and the general model was trained on all the data regardless of pitch. Roughly, 70% of the compiled speech data was used for training, 20% was used for validation and 10% was used for testing. The split was maintained approximately across each model and iteration. Stratified crossvalidation, with three iterations, was used to sample the data into the training, validation and testing splits. All hyperparameters, except the number of epochs was kept constant in each model and iteration. Finally, a Unity implementation of DeepSpeech was used and modified to allow the user to control their speech through a push-to-talk method.

4.0 Results

This section outlines the results for all 3 iterations of training and for all 3 models. Figure 4.1 contains a graph of the average of the WER for each model before and after training. Figure 4.2 contains averages of the improvements in WER or relative WER for each model before and after training. The other graphs outline all the information, as well as the absolute and relative WER for each iteration separately.

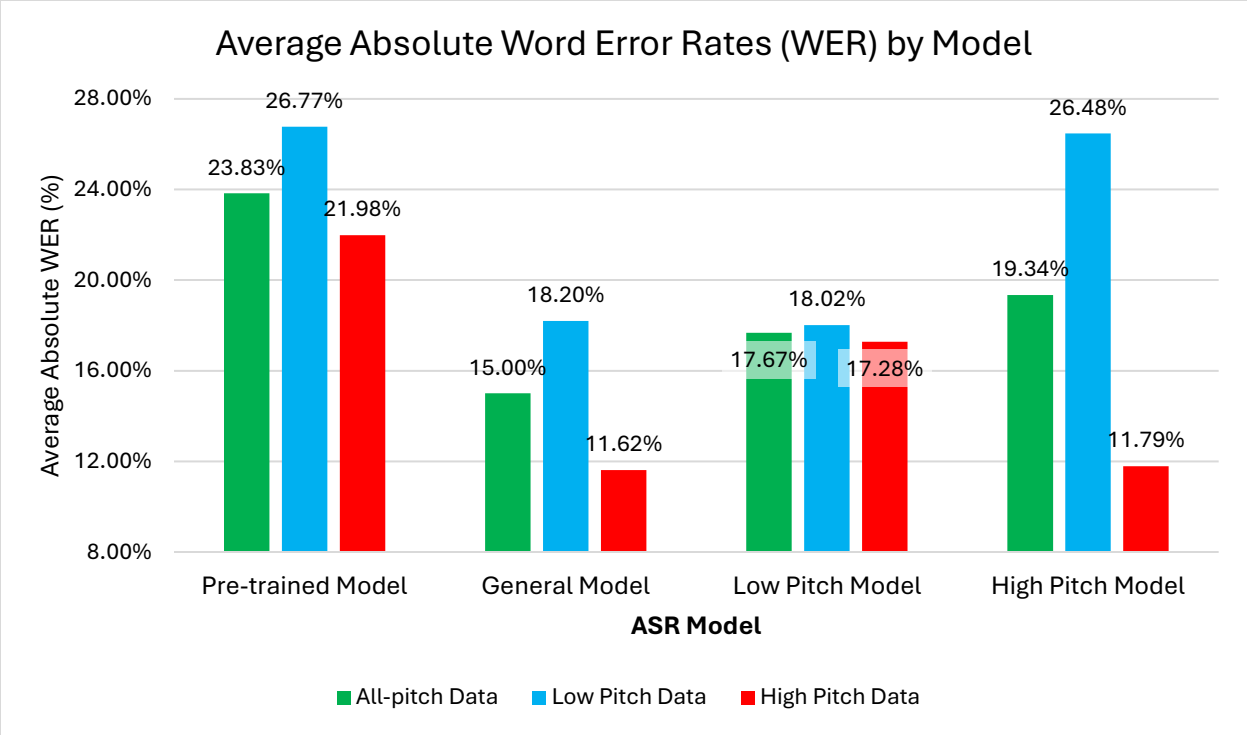

Figure 4.1 Average Absolute WER by Model for all 3 iterations.

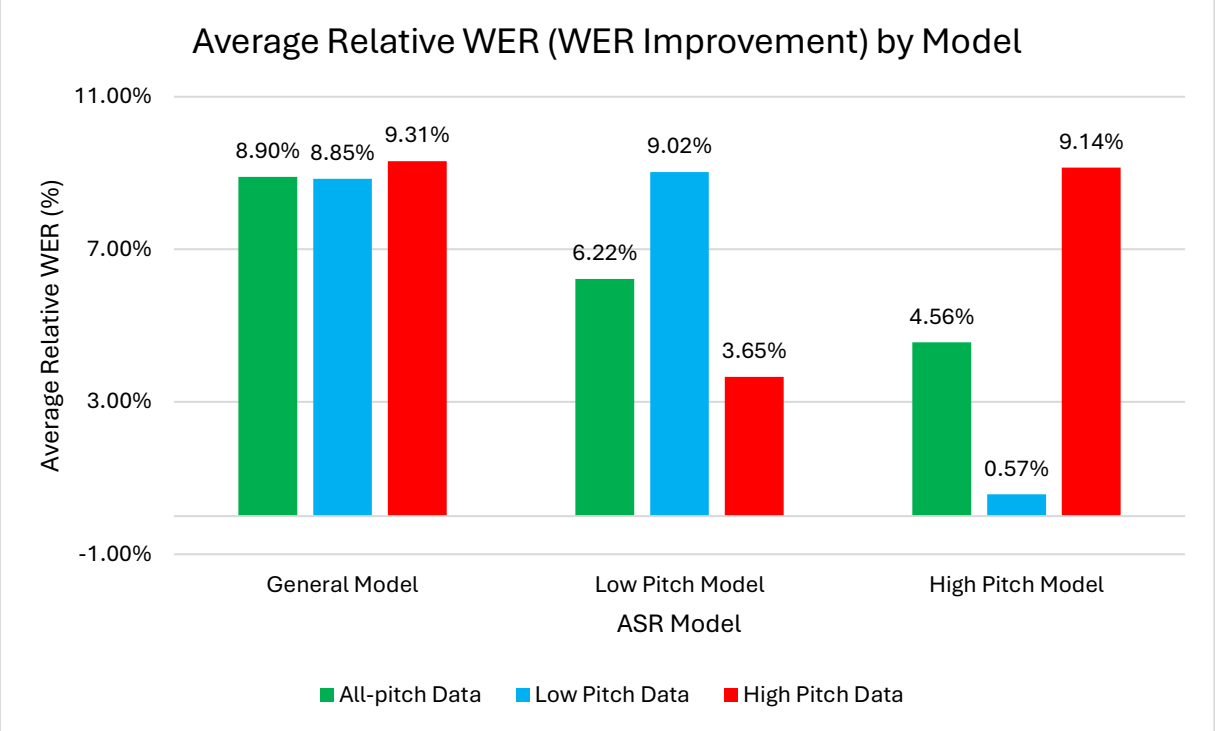

Figure 4.2 Average Relative WER or Average WER Improvement by Model for all 3 iterations.

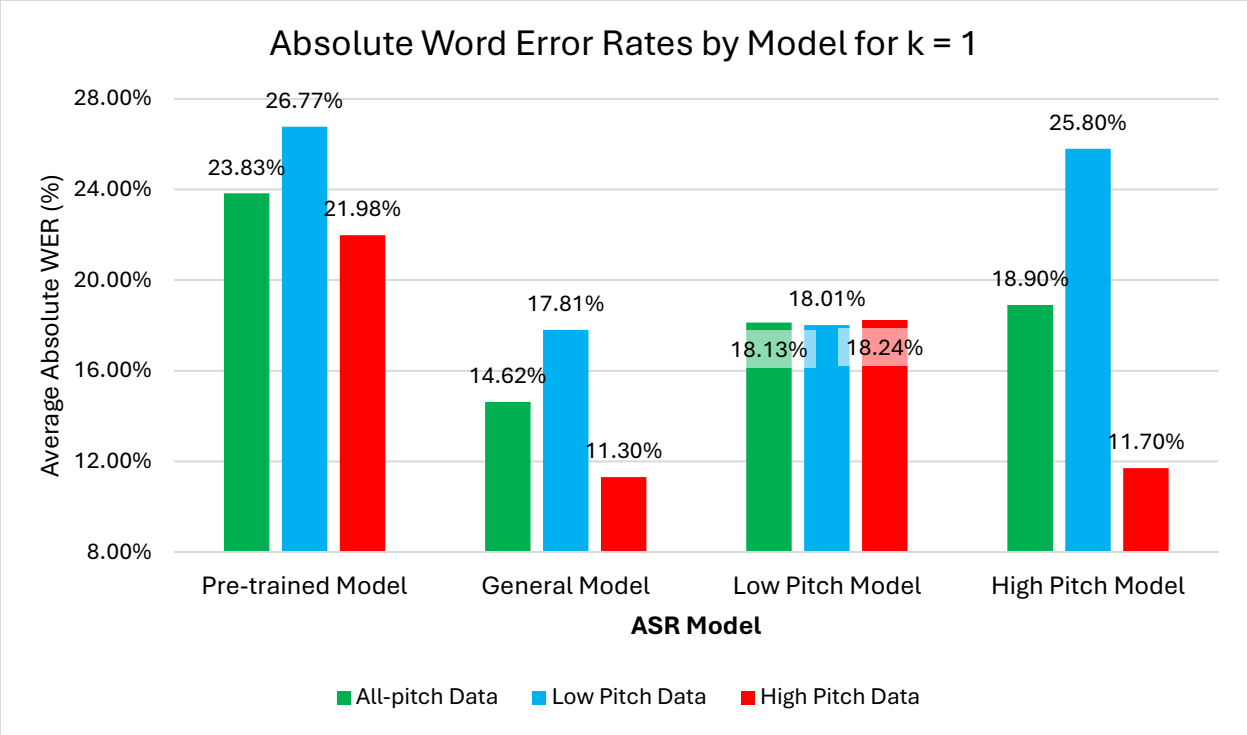

Figure 4.3 Absolute WER by Model for the 1st iteration of stratified crossvalidation.

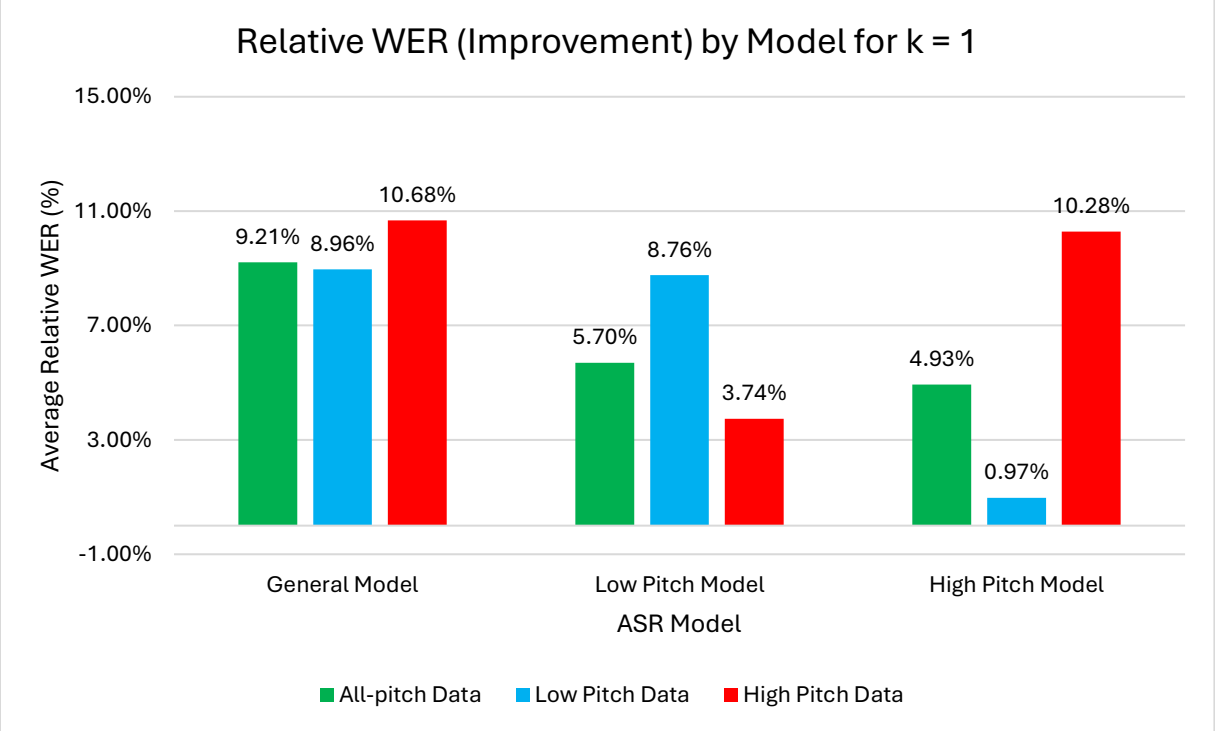

Figure 4.4 Relative WER or WER Improvement by Model for the 1st iteration of stratified crossvalidation.

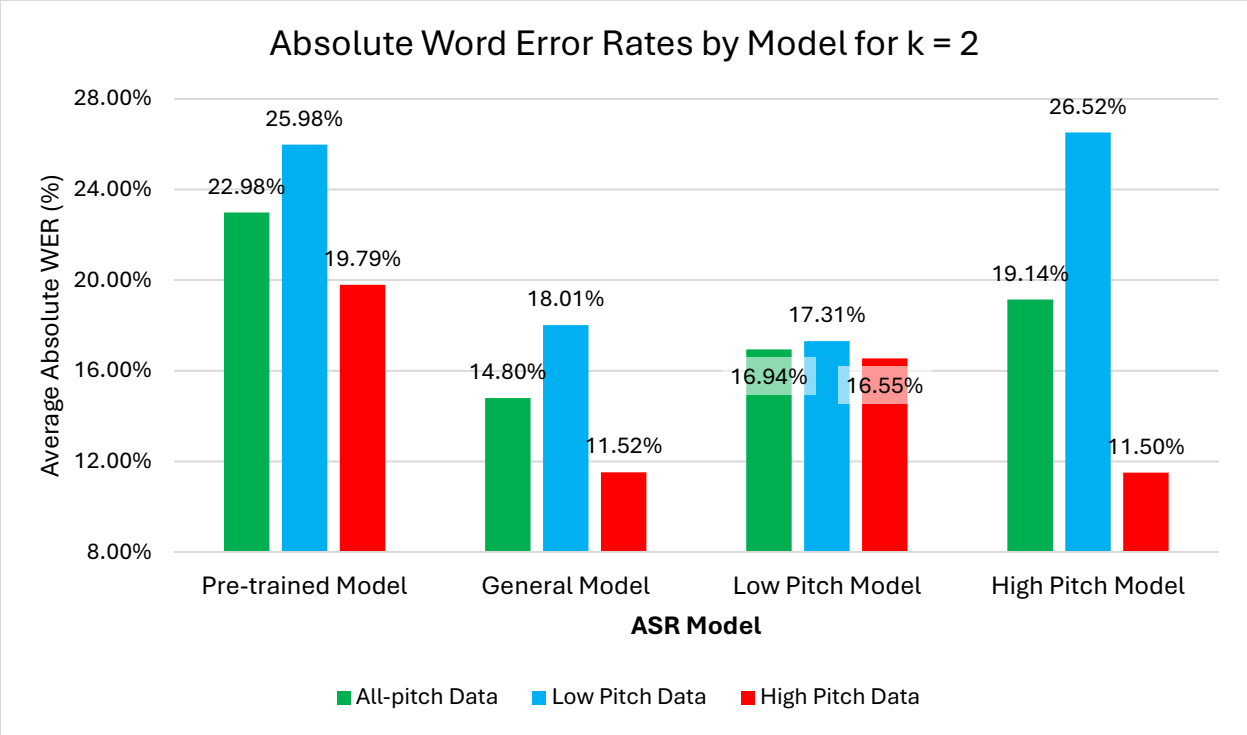

Figure 4.5 Absolute WER by Model for the 2nd iteration of stratified crossvalidation.

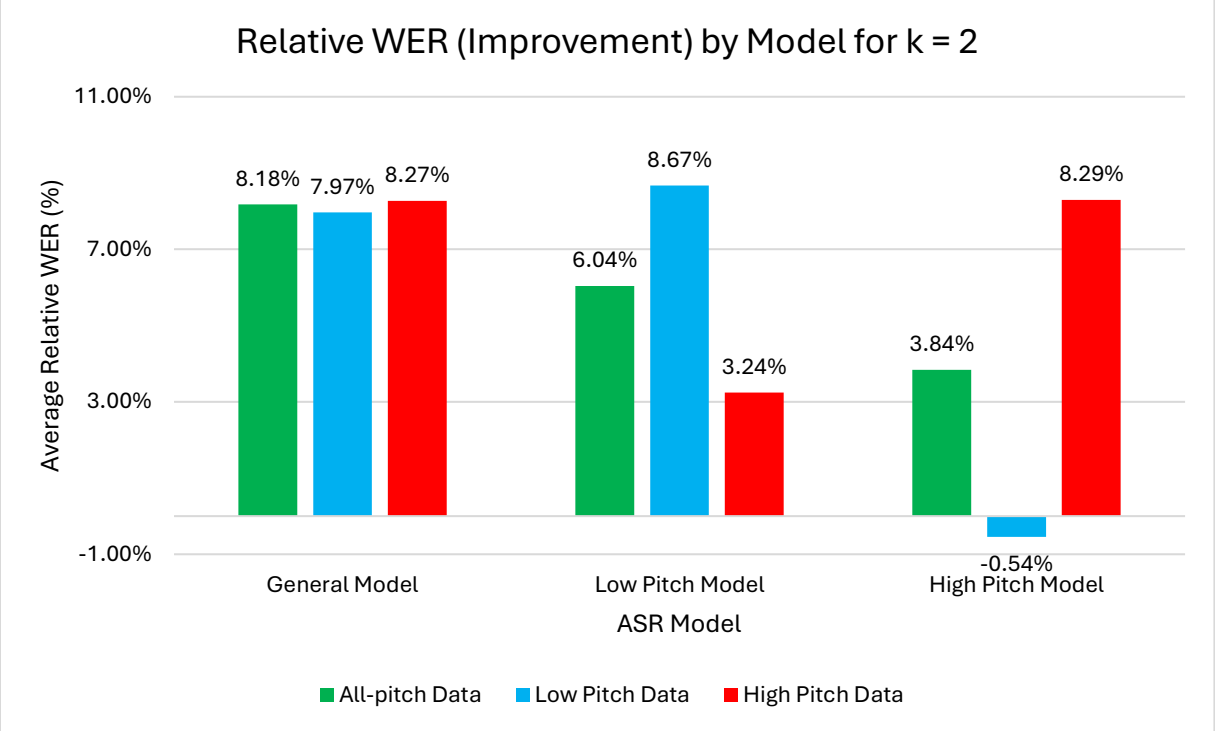

Figure 4.6 Relative WER or WER Improvement by Model for the 2nd iteration of stratified crossvalidation.

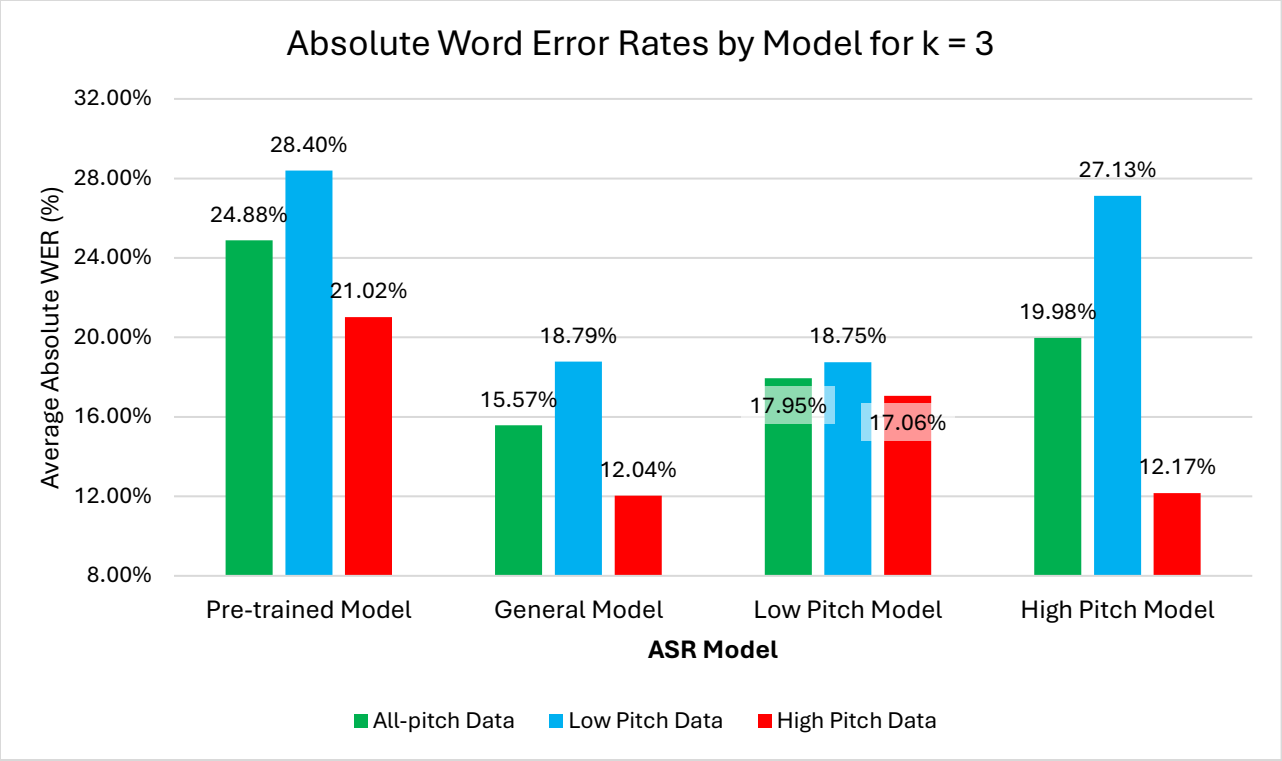

Figure 4.7 Absolute WER by Model for the 3rd iteration of stratified crossvalidation.

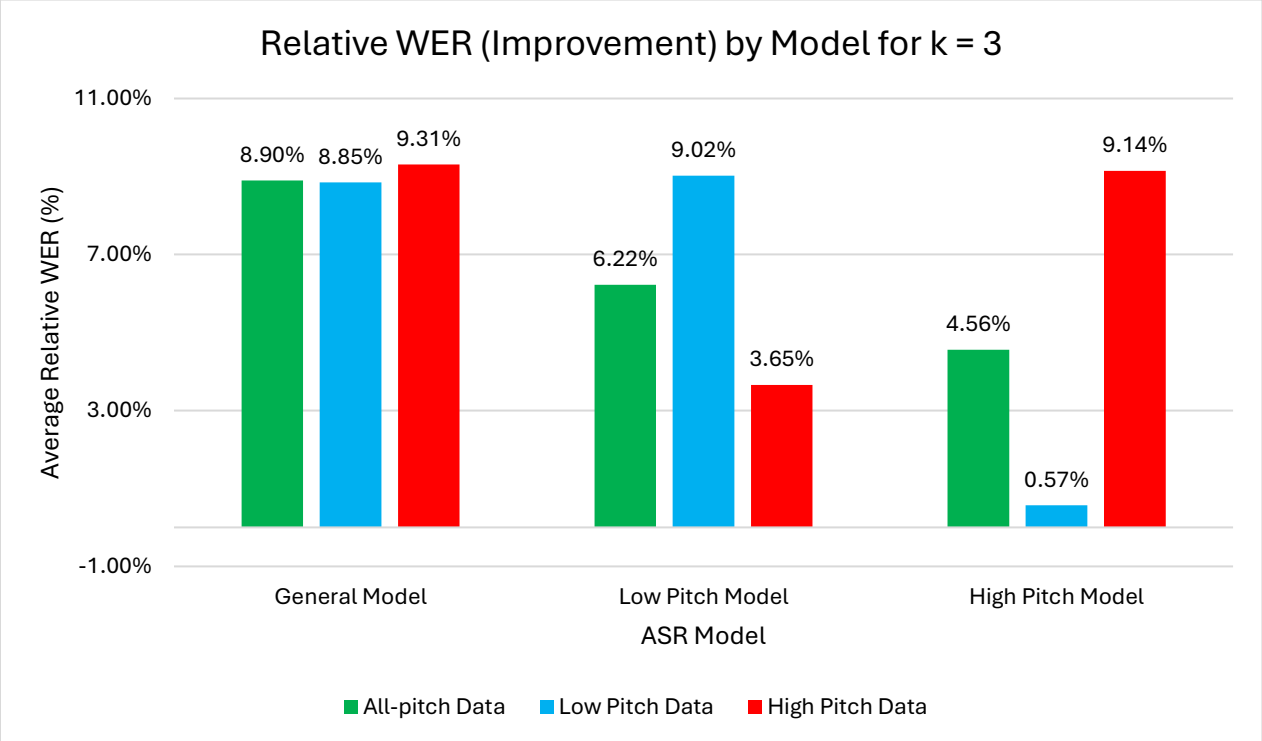

Figure 4.8 Relative WER or WER Improvement by Model for the 3rd iteration of stratified crossvalidation.

5.0 Discussion

Key Points and Discussion

All models had an average WER improvement greater than 8.85% on their trained data, with the general model averaging relative WERs of 8.85% and 9.31%% for the low and high pitch data respectively. This performance is inline with similar experiments done to fine-tune models to various accents which had relative WERs ranging from 7 – 10% [68] [69]. The general model also only improved on the high pitch model by 0.17% when transcribing the high pitch data. While the low pitch model performed better than the general model on the low pitch data by 0.17%. The extra 25 to 26 hours of unrepresentative speech data had little to no effect on the performance of the general model on the high and low pitch data sets. The same amount of representative data was able to improve the model's prediction by 8 to 9%. This shows the importance of using representative data when training an ASR model. Finally, the largest difference in absolute WER between iterations of stratified crossvalidation was 2%. This suggests that random sampling is sufficient when training and testing ASR models.

Word Error Analysis

Training an ASR model with representative data can result in consistent improvements in the model's performance. However, it is important to analyze the causes of the WER to make any further improvements in WER. The word errors that the models encountered fall into three major categories namely, accent errors, language errors and character errors.

Accent errors are transcription errors that are made due to differences in the accent of the training speech data and the user's speech data. Majority of the errors that the pre-trained model

encountered fall in this category. One can identify accent errors by incorrect vowel transcription. Studies have shown that the main differences in accents are in vowel sounds, due to the change in formant positioning [16]. An example of accent error encountered by the pre-trained English model can be seen below in Figure 5.1. The removal of accent error was the source of majority of the improvement for this project's models. This is because accent errors are due to issues with the acoustic model which was selected for training in this project.

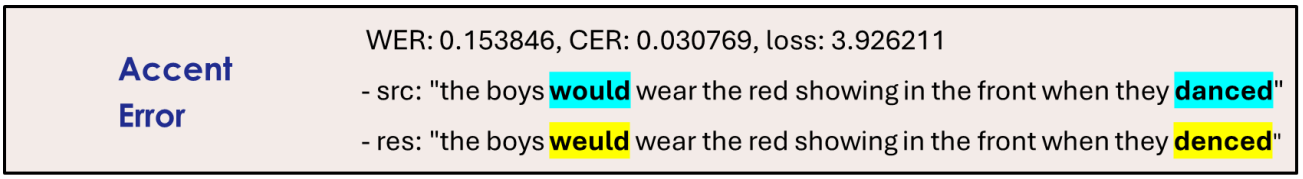

Figure 5.1 Example of an accent error encountered by the pre-trained model

Language Errors are transcription errors that are due to the attempt to transcribe words or phrases that the acoustic and language model is not familiar with. These errors usually have the largest individual WER and the largest loss. They add to the WER of the model while not necessarily indicating the acoustic model's inability to transcribe the speech. Training the acoustic model with this language would make it better at transcribing it but its improvements will not be notice able if the scorer is not trained on that language. There were very few language errors in the speech data, but their existence suggests that the absolute WER of the models are better than they seem. An example of a language error made by the pre-trained and trained models can be seen below in Figure 5.2.

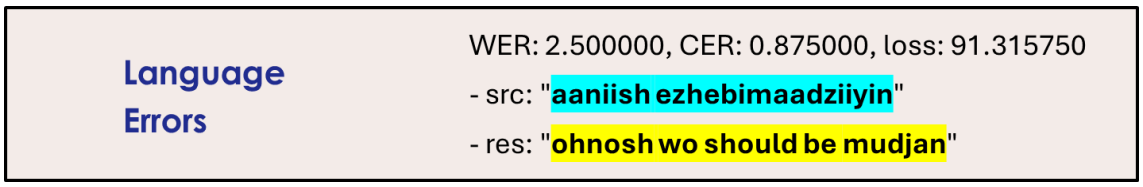

Figure 5.2 Example of a language error encountered by the pre-trained model

Finally, character errors are transcription errors that occur because of issues with the language model. They are characterized by high WER but low loss. They occur when the acoustic and language model correctly transcribe speech, but the scorer or language model does not recognize the transcription as being accurate. An example of this type of transcription error can be seen in Figure 5.3. Character errors contribute to the total WER but are caused by errors in model evaluation. These errors can be alleviated by training or changing the language model. Few character errors were observed before and after training, but they suggest that the absolute WER of the models are better than observed.

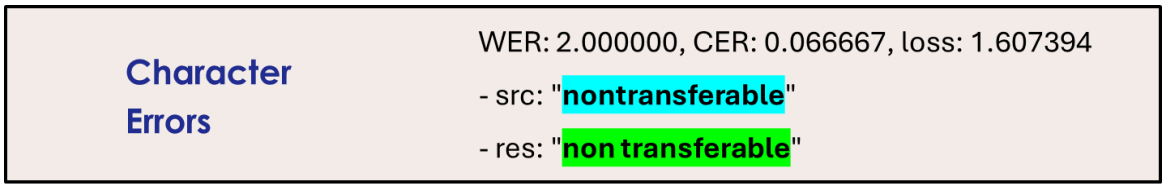

Figure 5.3 Example of a character error encountered by the models

These errors point to the importance of analyzing the word errors of a model before training. This helps identify the model's issues. Thereby allowing for a strategic approach to improving the model's performance.

6.0 Conclusion, Limitations and Future Work

Conclusion

In Conclusion, the goal of this project was design a ASR system for an Indigenous counselling program called the Elder Project. An open-source ASR system model called DeepSpeech was selected for this project. Representative speech data was acquired from audiobooks and were combined into speech corpora that are compatible with DeepSpeech. DeepSpeech's pre-trained English model was fine-tuned to the intended user's voice type using the created speech corpora. Two pitch-specific models and one general model were trained on the collected data. All the models showed a WER improvement of at least 8.85% keeping up with similar experiments on accented speech. Finally, DeepSpeech's transcription software was implemented on the Unity Development Engine allowing the models to work with the other parts of the Elder Project.

Limitations

DeepSpeech Development

As of 2021, DeepSpeech was no longer being developed. While DeepSpeech still remains competitive in terms of ASR performance, the lack of updates has resulted in software bugs. Majority of the software bugs are due to ASR systems incompatibility with newer GPUs and the updated version of tensor flow. As a result, it was necessary to revert the version of the GPUs drivers and corresponding software to allow DeepSpeech to function. While this was possible with the GPUs used for this project, NVIDA and tensor flow eventually plan to cease support for

previous GPU driver versions on their new GPUs. This would result in DeepSpeech being incompatible with new hardware.

Hardware RAM

Another limitation that was encountered during this project was the amount of Random Access Memory (RAM) that was available on the computer that was used for this project. The RAM size limited the maximum batch size available for training and prevented the ideal batch size from being used for this project.

Future Work

While significant improvement has been made in performance for the project's models, more work can still be done to minimize WER of the models. This section of the conclusion will go over some suggestions for future work on this project and in the field of speech recognition.

Investigate Accent Trade-Off

One interesting project would be to investigate the effect accented speech data has on the pre-trained model's performance. The project would test the model's performance on the previously trained accent and voice type (US male) [23]. This would show the effect of training the model on new accents. The project could also test the model's performance on outlier accents before and after training to reveal if training on a different accent has a positive, neutral or negative effect on the pre-trained model's ability to transcribe other outlier accents.

Retrain a language model

Another project idea would be to retrain a new language model to work better with counselling program. While DeepSpeech is incapable of fine-tuning its language model, it possible to retrain a new language model for a counselling. The language model could include the text from the audiobooks selected in this project and train a language model that is better suited for a counselling program.

Identifying and rating accented speech

Another interesting project would be to train machine learning algorithm to identify accents and rate the accent thickness. One of the issues with sourcing speech for this project was finding accented speech. An algorithm that could identify and rate accent thickness would help identify accented speech data.

References

[1] Hannun, A. *et al*, “Deep Speech: Scaling up end-to-end speech recognition”, arXiv e-prints, 2014. doi:10.48550/arXiv.1412.5567.

[2] Nagórski, Arkadiusz et al. “Optimal selection of speech data for automatic speech recognition systems.” *Interspeech* (2002).

[3] Chen, C., Hou, N., Hu, Y., Shirol, S., and Chng, E. S., “Noise-robust Speech Recognition with 10 Minutes Unparalleled In-domain Data”, arXiv e-prints, 2022. doi:10.48550/arXiv.2203.15321.

[4] Hinsvark, A., “Accented Speech Recognition: A Survey”, arXiv e-prints, 2021. doi:10.48550/arXiv.2104.10747.

[5] M. Labied, A. Belangour, and M. Banane, ‘Delve deep into End-To-End Automatic Speech Recognition Models’, in *2023 International Seminar on Application for Technology of Information and Communication (iSemantic)*, 2023, pp. 164–169.

[6] N. Kazanina, J. S. Bowers, and W. Idsardi, ‘Phonemes: Lexical access and beyond’, *Psychonomic Bulletin & Review*, vol. 25, no. 2, pp. 560–585, Apr. 2018.

[7] M. Labied and A. Belangour, "Moroccan Dialect ‘Darija’ Automatic Speech Recognition: A Survey", 2021 IEEE 2nd International Conference on Pattern Recognition and Machine Learning (PRML), pp. 208-213, Jul. 2021.

[8] “deepspeech-playbook,” *deepspeech-playbook*. https://mozilla.github.io/deepspeech-playbook/TESTING.html#word-error-rate--character-error-rate--loss-and-model-performance (accessed Aug. 01, 2024).

[9] “Learn to Love Your Folds - Anatomy of Your Voice, Pt. 1 — Blog,” *Alexander R Adams*. https://adams-voice.com/blog/vocal-anatomy

[10] “Harmonics Vs. Formants,” *VoiceScienceWorks*. https://www.voicescienceworks.org/harmonics-vs-formants.html

[11] “3.10. Fundamental frequency (F0) — Introduction to Speech Processing,” *speechprocessingbook.aalto.fi*. https://speechprocessingbook.aalto.fi/Representations/Fundamental_frequency_F0.html

[12] G. Andrade-Miranda, ‘Analyzing of the vocal fold dynamics using laryngeal videos’, 06 2017.

[13] R. Mugitani and S. Hiroya, ‘Development of vocal tract and acoustic features in children’, *Acoustical Science and Technology*, vol. 33, no. 4, pp. 215–220, 2012.

[14] Keiran, “Formant-Shifting: How It Works with Pitch-Shifting - Home Studio Magic,” Apr. 10, 2023. https://homestudiomagic.com/formant-shifting-how-it-works-with-pitch-shifting/ (accessed Aug. 01, 2024).

[15] “Acoustic Phonetics: Formants,” *Umanitoba.ca*, 2019. https://home.cc.umanitoba.ca/~krussll/phonetics/acoustic/formants.html

[16] P. Ladefoged, *A course in phonetics*, 3rd Edition. Fort Worth : Harcourt Brace Jovanovich College Publishers, 1993, pp. 185–187. Available: https://archive.org/details/courseinphonetic00lade/mode/1up

[17] “2.2. Formants of Vowels – Phonetics and Phonology,” *Eduhk*. https://corpus.eduhk.hk/english_pronunciation/index.php/2-2-formants-of-vowels/

[18] P. F. Assmann and T. M. Nearey, ‘Relationship between fundamental and formant frequencies in voice preference’, *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. EL35–EL43, Jul. 2007.

[19] G. E. Peterson and H. L. Barney, ‘Control Methods Used in a Study of the Vowels’, *The Journal of the Acoustical Society of America*, vol. 24, no. 2, pp. 175–184, Mar. 1952.

[20] K. Pisanski, A. Groyecka-Bernard, and P. Sorokowski, ‘Human voice pitch measures are robust across a variety of speech recordings: methodological and theoretical implications’, *Biol Lett*, vol. 17, no. 9, p. 20210356, Sep. 2021.

[21] “Release DeepSpeech 0.9.3 · mozilla/DeepSpeech,” *GitHub*. https://github.com/mozilla/DeepSpeech/releases/tag/v0.9.3

[22] Sciforce, “Automatic Speech Recognition (ASR) Systems Compared,” *Sciforce*, Jul. 07, 2021. https://medium.com/sciforce/automatic-speech-recognition-asr-systems-compared-6ad5e54fd65f

[23] “DeepSpeech Model — Mozilla DeepSpeech 0.10.0-alpha.3 documentation,” *deepspeech.readthedocs.io*. https://deepspeech.readthedocs.io/en/master/DeepSpeech.html (accessed Aug. 01, 2024).

[24] D. P. Kingma and J. Ba, ‘Adam: A Method for Stochastic Optimization’, *arXiv [cs.LG]*. 2017.

[25] J. C. Spall, ‘Stochastic Optimization’, in *Handbook of Computational Statistics: Concepts and Methods*, J. E. Gentle, W. K. Härdle, and Y. Mori, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 173–201.

[26] M. J. Kochnderfer and T. A. Wheeler, *Algorithms for optimization*. Cambridge, Massachusetts Etc.: The MIT Press, 2019, p. 8.

[27] “Connectionist Temporal Classification,” *GeeksforGeeks*, Dec. 27, 2023. https://www.geeksforgeeks.org/connectionist-temporal-classification/

[28] “deepspeech-playbook,” *deepspeech-playbook*. https://mozilla.github.io/deepspeech-playbook/SCORER.html#building-your-own-scorer (accessed Aug. 01, 2024).

[29] S. L. Aouragh, A. Yousfi, S. Laaroussi, H. Gueddah, and M. Nejja, ‘A new estimate of the n-gram language model’, *Procedia Computer Science*, vol. 189, pp. 211–215, 2021.

[30] Maison, L. and Estève, Y., “Improving Accented Speech Recognition with Multi-Domain Training”, arXiv e-prints, 2023. doi:10.48550/arXiv.2303.07924.

[31] Deng, K., Cao, S., and Ma, L., “Improving Accent Identification and Accented Speech Recognition Under a Framework of Self-supervised Learning”, arXiv e-prints, 2021. doi:10.48550/arXiv.2109.07349.

[32] Lu, Haoyu et al. “Speech and Noise Dual-stream Spectrogram Refine Network with Speech Distortion Loss for Robust Speech Recognition.” *ArXiv* abs/2305.17860 (2023): n. pag.

[33] Kumalija, Elhard & Nakamoto, Yukikazu. (2022). “Performance evaluation of automatic speech recognition systems on integrated noise-network distorted speech”. Frontiers in Signal Processing. 2. 999457. 10.3389/frsip.2022.999457.

[34] “List of Dialects of English.” *Wikipedia*, Wikimedia Foundation, 13 Aug. 2023, en.wikipedia.org/wiki/List_of_dialects_of_English#Extinct. Accessed 16 Aug. 2023.

[35] Feng, S., Kudina, O., Halpern, B. M., and Scharenborg, O., “Quantifying Bias in Automatic Speech Recognition”, arXiv e-prints, 2021. doi:10.48550/arXiv.2103.15122.

[36] Latinus, M., Taylor, M.J. “Discriminating Male and Female Voices: Differentiating Pitch and Gender.” *Brain Topogr* **25**, 194–204 (2012). https://doi.org/10.1007/s10548-011-0207-9

[37] Magimai.-Doss, Mathew et al. “Using pitch frequency information in speech recognition.” *Interspeech* (2003).

[38] G. Yeung, R. Fan and A. Alwan, "Fundamental Frequency Feature Normalization and Data Augmentation for Child Speech Recognition," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, ON, Canada, 2021, pp. 6993-6997, doi: 10.1109/ICASSP39728.2021.9413801

[39] “audiosegment module — AudioSegment documentation,” *audiosegment.readthedocs.io*. https://audiosegment.readthedocs.io/en/latest/audiosegment.html

[40] G. Hu and D. Wang, ‘Auditory Segmentation Based on Onset and Offset Analysis’, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 2, pp. 396–405, 2007.

[41] “deepspeech-playbook,” *deepspeech-playbook*. https://mozilla.github.io/deepspeech-playbook/DATA_FORMATTING.html (accessed Aug. 01, 2024).

[42] "Weighted Average Calculator," *Inch Calculator*. https://www.inchcalculator.com/weighted-average-calculator/ (accessed Aug. 01, 2024).

[43] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, 'CREPE: A Convolutional Representation for Pitch Estimation', *arXiv [eess.AS]*. 2018.

[44] S. Taiwo, "A Comprehensive Guide to Selecting and Estimating GPUs for Serving ML Models.," *Medium*, Jul. 05, 2023. https://medium.com/@samuel-taiwo/a-comprehensive-guide-to-selecting-and-estimating-gpus-for-serving-ml-models-23d2874dcbd8 (accessed Aug. 01, 2024).

[45] P. Baheti, "Train, Validation, and Test Set: How to Split Your Machine Learning Data," *V7labs.com*, Sep. 13, 2021. https://www.v7labs.com/blog/train-validation-test-set

[46] I. Muraina, 'IDEAL DATASET SPLITTING RATIOS IN MACHINE LEARNING ALGORITHMS: GENERAL CONCERNS FOR DATA SCIENTISTS AND DATA ANALYSTS', 02 2022.

[47] Q. H. Nguyen *et al.*, 'Influence of Data Splitting on Performance of Machine Learning Models in Prediction of Shear Strength of Soil', *Mathematical Problems in Engineering*, vol. 2021, no. 1, p. 4832864, 2021.

[48] V. R. Joseph, 'Optimal ratio for data splitting', *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 15, no. 4, pp. 531–538, 2022.

[49] A. Acharya, "Training, Validation, Test Split for Machine Learning Datasets," *encord.com*, Jun. 13, 2023. https://encord.com/blog/train-val-test-split/

[50] F. Hvilshøj, "Balanced and Imbalanced Datasets in Machine Learning [Introduction]," *encord.com*, Nov. 11, 2022. https://encord.com/blog/an-introduction-to-balanced-and-imbalanced-datasets-in-machine-learning/

[51] M. Elfil and A. Negida, 'Sampling methods in Clinical Research; an Educational Review', *Emerg (Tehran)*, vol. 5, no. 1, p. e52, Jan. 2017.

[52] D. Kernler, "English: A visual representation of selecting a random sample using the stratified sampling technique," *Wikimedia Commons*, Oct. 30, 2014. https://commons.wikimedia.org/wiki/File:Stratified_sampling.PNG (accessed Aug. 01, 2024).

[53] J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation," *Machine Learning Mastery*, Oct. 04, 2023. https://machinelearningmastery.com/k-fold-cross-validation/

[54] "[Tutorial] - Cross Validation & Nested CV," *kaggle.com*. https://www.kaggle.com/code/jacoporepossi/tutorial-cross-validation-nested-cv

[55] "Models & Languages Overview," *Deepgram Docs*. https://developers.deepgram.com/docs/models-languages-overview (accessed Aug. 01, 2024).

[56] "Measure and improve speech accuracy | Cloud Speech-to-Text Documentation," *Google Cloud*. https://cloud.google.com/speech-to-text/docs/speech-accuracy

[57] R. Kohavi and Others, 'A study of cross-validation and bootstrap for accuracy estimation and model selection', in *Ijcai*, 1995, vol. 14, pp. 1137–1145.

[58] J. Brownlee, "A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks," *Machine Learning Mastery*, Dec. 06, 2018. https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/

[59] J. Brownlee, "Gradient Descent For Machine Learning," *Machine Learning Mastery*, Mar. 22, 2016. https://machinelearningmastery.com/gradient-descent-for-machine-learning/

[60] D. Masters and C. Luschi, 'Revisiting Small Batch Training for Deep Neural Networks', *arXiv [cs.LG]*. 2018.

[61] Y. Bengio, 'Practical recommendations for gradient-based training of deep architectures', *arXiv [cs.LG]*. 2012.

[62] L. Prechelt, 'Early Stopping - But When?', in *Neural Networks: Tricks of the Trade*, G. B. Orr and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 55–69.

[63] "deepspeech-playbook," *deepspeech-playbook*. https://mozilla.github.io/deepspeech-playbook/TRAINING.html (accessed Aug. 01, 2024).

[64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 'Dropout: a simple way to prevent neural networks from overfitting', *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[65] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016, p. 429.

[66] R. Reed and R. Marks II, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Bradford Books, 1999, p. 72.

[67] K. Babilinski, "Babilinski/deep-speech-unity," *GitHub*, Jul. 21, 2024. https://github.com/Babilinski/deep-speech-unity (accessed Aug. 01, 2024).

[68] P. Chitkara, M. Riviere, J. Copet, F. Zhang, and Y. Saraf, 'Pushing the performances of ASR models on English and Spanish accents', *arXiv [cs.CL]*. 2022.

[69] P. Sullivan, T. Shibano, and M. Abdul-Mageed, 'Improving Automatic Speech Recognition for Non-native English with Transfer Learning and Language Model Decoding', in *Analysis and Application of Natural Language and Speech Processing*, M. Abbas, Ed. Cham: Springer International Publishing, 2023, pp. 21–44.

Appendix A: Selected Audiobooks and Vocal Pitch Groups

Audiobook Names	Author	Narrator(s)	Narrator Sex	Time (minutes)	Vocal Pitch (hertz)
Calling For A Blanket Dance	Oscar Hokeah	Oscar Hokeah	Male	182 min	161.85 Hz
		Rainy Feilds	Female	220 min	216.44 Hz
The Roundhouse	Louise Erdrich	Gary Farmer	Male	759 min	146.65 Hz
Night Of The Living Rez	Morgan Talty	Darrell Dennis	Male	426 min	173.74 Hz
Firekeeper's Daughter	Angeline Boulley	Isabella Star Lablanc	Female	853 min	186.92 Hz
Moon Of The Crusted Snow	Waubgeshig Rice	Billy Merasty	Male	406 min	133.21 Hz
Five Little Indians	Michelle Good	Kyla Garcia	Female	634 min	235.96 Hz
TOTAL TIME FOR HIGH PITCH DATA				1707 min (28h 27 min)	
TOTAL TIME FOR LOW PITCH DATA				1773 min (29h 33min)	

HIGH PITCH DATA

LOW PITCH DATA

Appendix B: Hyperparameters

NO. OF EPOCHS TRAINED BY EACH MODEL

(USING EARLY STOPPING)

MODEL	1ST ITERATION (EPOCHS)	2ND ITERATION (EPOCHS)	3RD ITERATION (EPOCHS)
GENERAL MODEL	25	19	21
LOW PITCH MODEL	24	29	25
HIGH PITCH MODEL	18	21	21

HYPERPARAMETER VALUES FOR ALL MODELS

HYPERPARAMETER	VALUE USED
HIDDEN LAYERS	2048
BATCH SIZE	12
MAXIMUM NO. OF EPOCHS (epochs)	30 EPOCHS
FREQUENCY OF EARLY STOP CHECKS (es_epochs)	2 EPOCHS
MINIMUM REQUIRED CHANGE IN LOSS (es_min_delta)	0.06
DROPOUT RATE	0.4
LEARNING RATE	0.0001