Speech Recognition For Medical Team

Xianming Jin

MInf Project (Part 2) Report

Master of Informatics
School of Informatics
University of Edinburgh

2021

Abstract

Speech Recognition has become popular and widely used in many fields nowadays. Recently, some researches show that the machine learning-based speak recognition system has outperformed the human dispatch for recognising cardiac arrest in emergency calls regarding speed and sensitivity. Those works are provided with a large dataset, making it easier to train the recognition system—however, only a limited emergency call data (3hours) is available in this project. By adding 1.5 hours of training data to the baseline of our previous work [Jin, 2020], we had shown an improvement from 78.63% to 49.00% on WER. We also investigated the effects of various data augmentation techniques, including Speed Perturbation, Volume Perturbation, Noise Injection and SpecAgument, and the feasibility of combining them. At the end of this work, we achieved almost an absolute 1.54% improvement from the best model in our previous work and an absolute 5.32% improvement from a plain model on Word Error Rate (WER) using only the data augmentations approach. The performance could be further improved by more than 1% on WER by training the model longer.

Acknowledgements

I would like to firstly express my sincere appreciation to my supervisor Professor Steve Renals, who consistently helped me and had guided me on the right track at the early stage of this project. It was unfortunate that he had to leave because of the illness, and I sincerely wish him all the best.

I would also express appreciation to my new supervisor Ondrej KLEJCH and Andrea Carmantini, who tirelessly dedicated their time to this project's supervision. This work would not have been possible without their guidance, patients, and encouragements.

Lastly, I am grateful to all my relatives and friends who shared their support, either physically or mentally, in the days when I was most needed. I am particularly thankful to Jia Li. I would not have been able to make it through several important periods without her.

Table of Contents

Chapter 1

Introduction

Automatic speech recognition (ASR) is a process of **transforming audio representation into the textual representation, without necessarily understanding the meaning of the text**. ASR has been an active research field over the past decades, and it has been developed rapidly with the exponential growth of computation power and large data available. It now has many real-life applications such as voice search, digital assistants like Siri and Alexa [Li et al., 2016]. ASR has also advanced to more challenging tasks such as speech recognition for medical teams.

Out-of-hospital cardiac arrest (OHCA) is when cardiac arrest happens outside of the hospital, which is responsible for more than 350,000 deaths on average per year in Europe [EuReCaTWO, 2017, Lancet, 2018]. A study has shown that the survival rate of OHCA patients is positively associated with the provision of Cardiopulmonary resuscitation (CPR) by the bystanders [Viereck et al., 2017]. The provision of CPR is often instructed by the medical dispatcher, who needs to identify OHCA by interacting with the bystander. However, identifying OHCA is not easy because only around 0.8% emergency calls are OHCA cases. Another study has shown that with using ASR combined with a machine-learning framework, the trained system had a significantly higher sensitivity comparing to the human dispatcher (84.1% vs 72.5%), with slightly lower specificity (97.3%vs 98.8%) [Blomberg et al., 2019]. Therefore, **applying speech recognition in medical fields could increase the survival rate of OHCA patients and saves lives**. Moreover, ASR could be used to identify other patients' conditions or prioritise the handlers' calls. Those use-cases motivated us to build a speech recognition for the medical team in the previous project.

Although ASR has been used and succeeded in many fields, **applying ASR to medical domain remains challenging** due to the following facts:

- Limited data available – the size of data that can be used to train a Speech Recognition System is much smaller than other fields. The limitation could be due to reasons such as ethic and privacy.
- Acoustic environment – noisy and maybe reverberant.
- Multi-accent,multi-lingual speakers.

- Limited language model and lexicon in the domain.
- Speaker diarization in emergency teams.

Due to the limitation of the data available in the domain, **AMI corpus was used in our previous project for training the baseline ASR model** [Jin, 2020]. The AMI corpus is a data set containing approximately 100 hours of meeting recordings. The language of the recording is English, but the speakers were primarily non-native English speakers. **The baseline performance turned out to be relatively low** (around 78.63% WER on the medical dataset).

Various reasons that might lead to poor performance had been investigated and discussed in the previous project. Although other factors such as language model and lexicon would also affect the performance, **the low performance was concluded as mainly due to the unmatched training and test dataset**. At the end of the previous research, the 3 hours of the medical dataset were split into training and testing dataset, each containing 1.5 hours of recording. Then a new model was trained using both the AMI dataset and the medical training dataset. **With the extra use of 1.5 hours of medical data, the WER of the model dropped from 78.63%** to **45.21%**. **This significant improvement shows the importance of the matched training and testing dataset**.

However, **acquiring medical data is difficult** due to many issues, such as privacy. **The difficulty of obtaining more data motivated us to focus on generating more synthetic data using data augmentation techniques in this project**. Also, in the previous project, the noise training approach was intended to minimise the degradation of the performance brought by the noisy acoustic environment. Unfortunately, it was not possible to finish the noise training due to some computational environments issues. In this project, noise training would be completed, and the effects of the noise training on the model would be evaluated.

Aims & Contribution

This project's overall aim will be the same as the previous project – building on a state-of-art speech recognition system adapted to the emergency medical domain. Achieving the project's main aim would be beneficial for medical teams and eventually might save lives in the future.

There are two objectives in this project while achieving the aim of the project. The first objective is to investigate the effects of various data augmentation technique in this work individually. Each data augmentation technique improved the ASR system's performance in the environment where it was proposed. Achieving the first objective is essential for improving the ASR system's performance. For example, investigating the effects of noise training could provide an approach that reduces the adverse impact of the noise-acoustic environment, particularly in medical scenarios.

The second objective is to investigate the effects of various combinations of different data augmentation techniques. For example, in the previous work, volume and speed augmentation techniques were applied together while using the extra medical data.

However, the effect of combining two augmentation technique was not investigated in the previous work. Also, in theory, noise training could be combined with speed and volume augmentations, but the effects of combining those augmentations are unknown. Would they be compatible when they are combined? Hence, the second objective is also important because it investigates the feasibility of various combinations of different data augmentation techniques in ASR that might further improve the ASR system's performance while mitigating the cost of collecting more data and alleviate the issues of limited training data.

Overall, the background knowledge required for the reader to understand the various data augmentation techniques will be introduced in Chapter 2. It mainly includes Speed Perturbation, Volume Perturbation, Noise Injection and SpecAugment [Park et al., 2019].

In Chapter 3, the details of the project's task, the evaluation metrics and the dataset would be discussed. At the end of this chapter, the related works would be discussed so the readers could understand how this work is different from others.

In Chapter 4, there will be an introduction to Kaldi and how the baseline model and other models could be built using Kaldi. The experiments and results evaluation will be shown in Chapter 5. Lastly, the conclusion, limitation and further work will be made in Chapter 6.

Chapter 2

Data Augmentation for ASR

Collecting training data is undesirable as it is usually expensive. Data augmentation is a technique that enables us to increase the size of training data by modifying the original data and without actually collecting them. It significantly increases the diversity of the training data and mitigates the cost of collecting the new data. The increased size and diversity of the "synthetic" training data could mimic the test conditions and prevent possible training problems such as overfitting.

Data augmentation techniques are now commonly used in neural network training and other domains [Seita, 2019]. In this chapter, all the data augmentation techniques used in this work would be introduced, including their pros and cons.

2.1 Vocal Tract Length Perturbation

Vocal tract length (VTL) varies from one to the others. This leads to different speakers' different sound signal even when the same words are spoken, causing an increment of difficulties in speech recognition tasks. Vocal tract length normalization (VTLN) was proposed [Li Lee and Rose, 1996, Eide and Gish, 1996] to overcome the difficulties. The basic idea of VLTN is using a warp factor α that linearly warps the frequency axis of the spectrogram of each speaker, accounting for the relative length of speakers' vocal tract compared to a canonical mean. This factor is fitted during training time for each speaker in the training dataset and applied during the decoding. However, a study [Giurgiu and Kabir, 2011] shows the VLTN does not improve on noisy speech.

Vocal Tract Length Perturbation (VTLP) adopts the idea of VTLN. Instead of finding the fitted factor α, it generates a random warp factor α that warps each utterance and warp the frequency axis such that a frequency f is mapped to a new frequency f' [Jaitly and Hinton, 2013]:

$$f' = \begin{cases} f\alpha, & \text{if } f \leq F_{hi}\frac{min(\alpha,1)}{\alpha} \\ S/2 - \frac{S/2 - F_{hi}min(\alpha,1)}{s/2 - F_{hi}\frac{min(\alpha,1)}{\alpha}}(S/2 - f), & \text{otherwise} \end{cases} \tag{2.1}$$

where S is the sampling frequency, F_{hi} is a boundary frequency chosen such it covers the significant formants (e.g. $F_{hi} = 4800$).

The warp factors in VLTN is traditionally assumed to lie between 0.8 and 1.2, whereas the warp factors in VTLP is assumed to lie between 0.9 and 1.1. This is because the goal in VTLP is to corrupt the data instead of normalising the data. The broad range may create unrealistic distortions.

The warping procedure in VLTP is applied directly on the filter banks, which maps the $f(i)$ of the $1 \leq i \leq N$ filter banks using equation Equation 2.1, where $f(i)$ is [Jaitly and Hinton, 2013]:

$$f(i) = m^{-1}(m(Fmin) + \frac{m(F_{max}) - m(F_{min})}{N-1})(i-1)) \tag{2.2}$$

and:

$m(f) = 1127.01 * log(1 + \frac{f}{700})$, which the Mel Scale function,

$m^{-1}(f) = 700(exp(\frac{f}{1127.01} - 1))$, which is the inverse of the Mel Scale function.

F_{max} and F_{min} are the frequency range. Thus, VTLP can perturb the spectral envelope of the audio segment while keeping the audio duration unchanged. The study [Jaitly and Hinton, 2013] shows consistent and significant improvements using VTLP on the TIMIT corpus.

2.2 Tempo Perturbation

In contrast to VTLP, Tempo Perturbation stretches the duration of the audio signal x(t) while keeping the shape of its spectral envelope unchanged. The principle of tempo perturbation is to decompose the input audio signal into the fixed length of short frames (usually 50 to 100 ms) that capture the local pitch content of the signal. The frames are then relocated on the time axis to achieve the actual tempo perturbation while preserving the signal's pitch.

More precisely, this procedure could be described mathematically: The input is a discrete-time audio signal $x : \mathbb{Z} \to \mathbb{R}$, equidistantly sampled at a sampling rate of F_s. The first step of the tempo perturbation is to split x into short analysis frames x_m, $m \in \mathbb{Z}$, each of them having a length of N samples. The analysis frames are spaced by an analysis hopsize H_a [Driedger and Müller, 2016]:

$$x_m(r) = \begin{cases} x(r + mH_a), & \text{if } r \in [-N/2 : N/2 - 1], \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

Then, these frames are relocated on the time axis by a specified synthesis hopsize H_s, giving as:

$$H_s = H_a \cdot \alpha. \tag{2.4}$$

where α is the perturbation factor. As fixing the overlap of the relocated frames is desirable, the synthesis hopsize H_s is often fixed (common choices are $H_s = N/2$ or $H_s = N/4$) while the analysis hopsize is given by $H_a = H_s/\alpha$. Simply superimposing the overlapping relocated frames would cause artefacts, which is undesirable. Therefore,

the analysis frame is suitably adapted to form the synthesis frame y_m before the reconstruction. Then the synthesis frames are superimposed to reconstruct the actual tempo perturbed output signal $y : \mathbb{Z} \to \mathbb{R}$ [Driedger and Müller, 2016]:

$$y(r) = \sum_{m \in \mathbb{Z}} y_m(r - mH_s). \tag{2.5}$$

The synthesis frames y_m could be defined differently depends on different approaches. A well-known approach in tempo perturbation is waveform similarity overlap-add (WSOLA), which makes $y(t)$ share the maximum similarities with $x(t)$ by finding the optimal position of each analysis frame iteratively[Verhelst and Roelands, 1993].

2.3 Speed Perturbation

Speed perturbation is a commonly used data augmentation techniques in state-of-art speech recognition systems. It produces a warped time signal $y(t)$ given an audio segment $x(t)$ and a perturbation factor α as [Geng et al., 2020]:

$$y(t) = x(\alpha t). \tag{2.6}$$

It is equivalent to the following changes in the frequency domain:

$$X(f) \to \frac{1}{\alpha} X(\frac{1}{\alpha} f) \tag{2.7}$$

where $X(f)$ and $\frac{1}{\alpha} X(\frac{1}{\alpha} f)$ represent the Fourier transform of $x(t)$ and $y(t)$ respectively.

Therefore, the speed perturbation leads to both duration and spectral envelope perturbations, shown in Figure 2.1. Note that in speed perturbation, the pitch (spectral envelope) is positively correlated with speed (duration), meaning higher speed resulting in a higher pitch of the signal. Some researches [Geng et al., 2020, Ko et al., 2015] show that the system applied the speed perturbation performs better than the ones applied solely or combined used VTLP and tempo perturbation. However, some [Zhou et al., 2017] could argue that speed perturbation limits the variation in augmented data due to the positive correlation between pitch and speed and consequently hurt the performance.

2.4 Volume Perturbation

Cepstral mean and variance normalisation and other normalisation techniques (e.g. cepstral mean normalisation (CMN)) are widely used in many neural network speech recognition system due to several advantages [Nguyen et al., 2018]:

1. It builds a more noisy-robust speech recognition system by cancelling out the environment changes.
2. They reduce the environment mismatches between training and testing dataset.

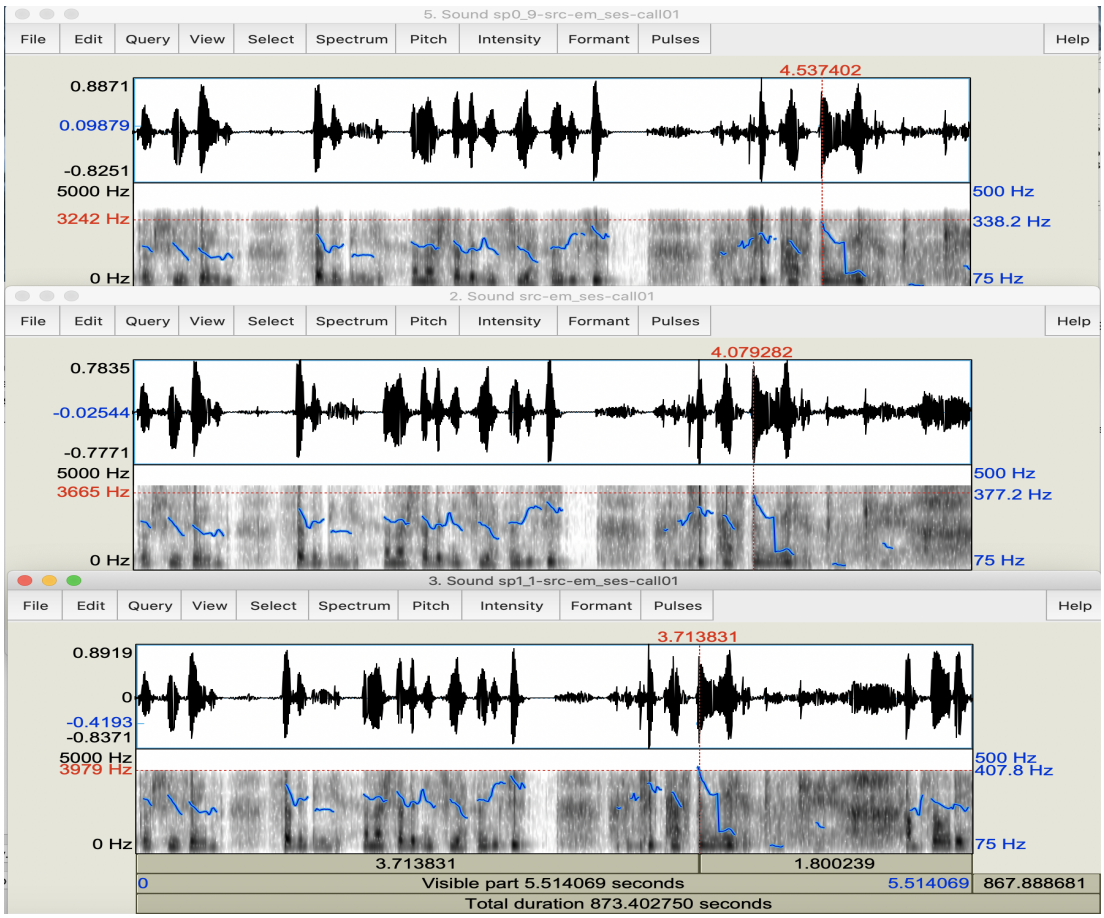

Figure 2.1: Speed perturbation applied to the original medical data (middle) with perturbation factors 0.9 (top) and 1.1 (bottom)

3. The normalised acoustic feature has zero mean, which is essential for neural network training [LeCun et al., 2012].

However, these approaches are not suitable for real-time situations [Nguyen et al., 2018]. Volume perturbation helps obviate the need for those techniques. This is because DNN learns to normalise cepstral mean and variance implicitly as DNN is optimised with data that has a different volume range. The implementation of volume perturbation is simple:

$$y(t) = \alpha \cdot x(t) \tag{2.8}$$

where α is the perturbation factor, $x(t)$ and $y(t)$ represent the amplitude of the input and output signal at time t respectively. Volume perturbation is usually applied across the entire signal using the same perturbation factor, shown in Figure 2.2.

In practice, each recording is usually scaled with a random variable that is drawn from a uniform distribution over $[\frac{1}{8}, 2]$. The idea of volume perturbation is simple, but it was observed that the volume perturbation has consistent improvement, as shown in [Nguyen et al., 2018]. Also, the study [Peddinti et al., 2015] shows that their system achieved 1.5% relative improvements in WER for the system used volume perturbation compared to the only used speed perturbation.

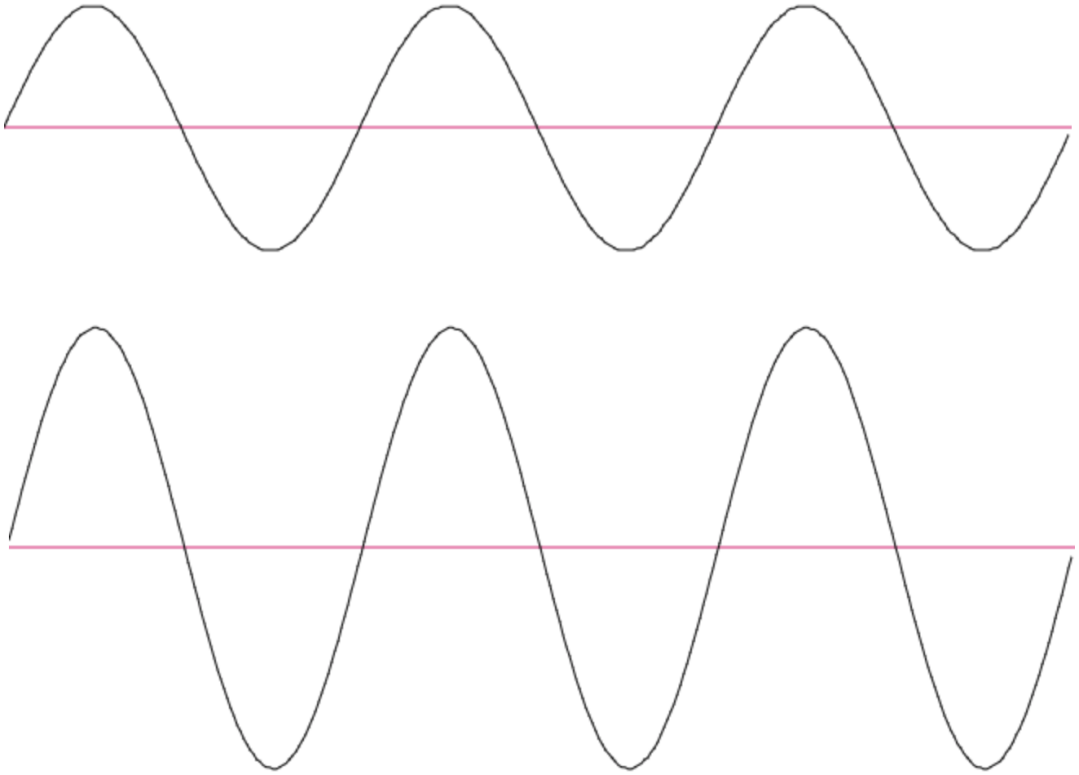

Figure 2.2: A naive example for volume perturbed signal of sin wave. The top figure represents y=sin(x) and the bottom figure represents y=2sin(x).

2.5 Noise Injection

It has been known for three decades that imposing noises to the input signal can improve the generalisation of neural networks [Sietsma, 1988]. "Noise injection" is another type of data augmentation technique that mixes the original training data with some sampled noise from real life. For example, if we have a clean speech track $x^{(i)}$ and a noise track $\xi^{(i)}$, then we can form the noisy speech track:

$$\hat{x}^{(i)} = x^{(i)} + \xi^{(i)} \tag{2.9}$$

It simulates the audio recorded in a noisy environment. However, this approach is risky when clean speech length is long. For example, if the clean speech has 1000 hours, it requires a unique noise track spanning roughly 1000 hours. Otherwise, say only 10 hours of repeating noise, the DNN could memorise the noise track and "subtract" it out from the noisy speech. Thus, instead of using a single long noise source $\xi^{(i)}$, we use a large number of short clips (that is easier to be collected in real life) and treat them as separate sources of noise before superimposing all of them:

$$\hat{x}^{(i)} = x^{(i)} + \xi^{(i)}_{(1)} + \xi^{(i)}_{(2)} + \ldots \quad . \tag{2.10}$$

Sox is also generally used to mix the original signal with the noise. Figure 2.3 is an example of using Sox to inject the noise. Noise is usually injected with a ratio

proportional to the original signal. Signal to Noise Ratio (SNR) is used to indicate the value of the noise injected. A lower SNR value means more noise injected.

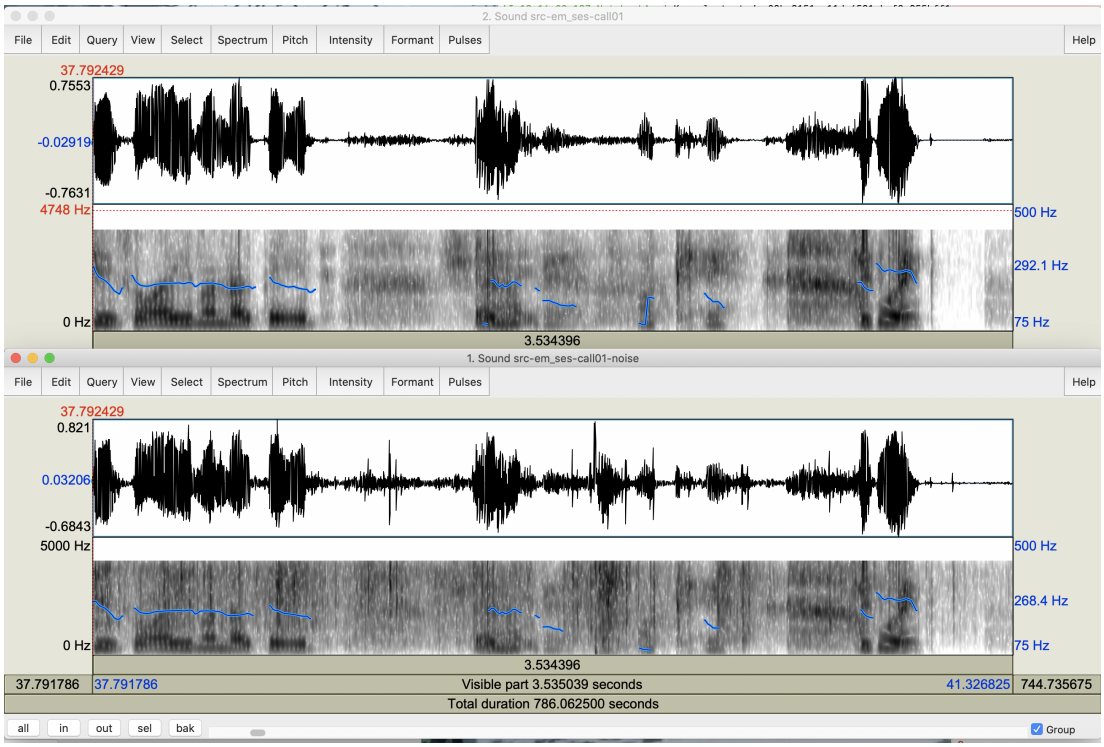

Figure 2.3: Noise is injected to a clean medical audio (top) resulting a noise audio (bottom)

As the study shows, adding a small magnitude of noise in the input behaves similarly to introducing some regularisation techniques in the objective function [Matsuoka, 1992]. With noise injection, the training prefers an optimal solution at which the objective function is less sensitive to the noise [Grandvalet and Canu, 1995]. Another study that showed the noise injection is closely related to some other generally used techniques, such as sigmoid gain scaling and target smoothing by convolution [Reed et al., 1995].

This approach has two advantages: firstly, the noise pattern introduced could be learned, and it shares the idea of multi-condition training; secondly, the perturbation caused by the noise data could improve the generalisation of the ASR system.

2.6 SpecAugment

SpecAugment is a state-of-art data augmentation published by Google Brain [Park et al., 2019] in December 2019. It is inspired by the success of augmentation in both the speech and vision domains, and it operates on the log mel spectrogram of the input audio instead of raw audio like other data augmentations. The core idea of SpecAugment is to treat the spectrogram as if it were an image, where the x-axis of the image is the time while the y-axis is frequency.

SpecAugment has three basic policies to augment the data:

- Time warp – In short, given a time step τ, time warp select a random point on the time axis passing through the centre of images within the time steps $(W, \tau - W)$, squeeze or stretch the data at the chosen point by a distance w which chosen from a uniform distribution from 0 to a time warp parameter W, in a randomly chosen direction by using interpolation techniques.
- Frequency Masking - A frequency channel $[f_0, f_0 + f)$ are masked, where f is first chosen from a uniform distribution from 0 to the frequency mask parameter F, and f_0 is chosen from $[0, v - f)$. v is the number of mel frequency channels.
- Time Masking - t consecutive time steps $[t_0, t_0 + t)$ are masked, where t is first chosen from a uniform distribution from 0 to the time mask parameter T, and t_0 is chosen from $[0, \tau - t)$. An upper bound is introduced on the time mask so that the time mask cannot be wider than p times the number of times steps.

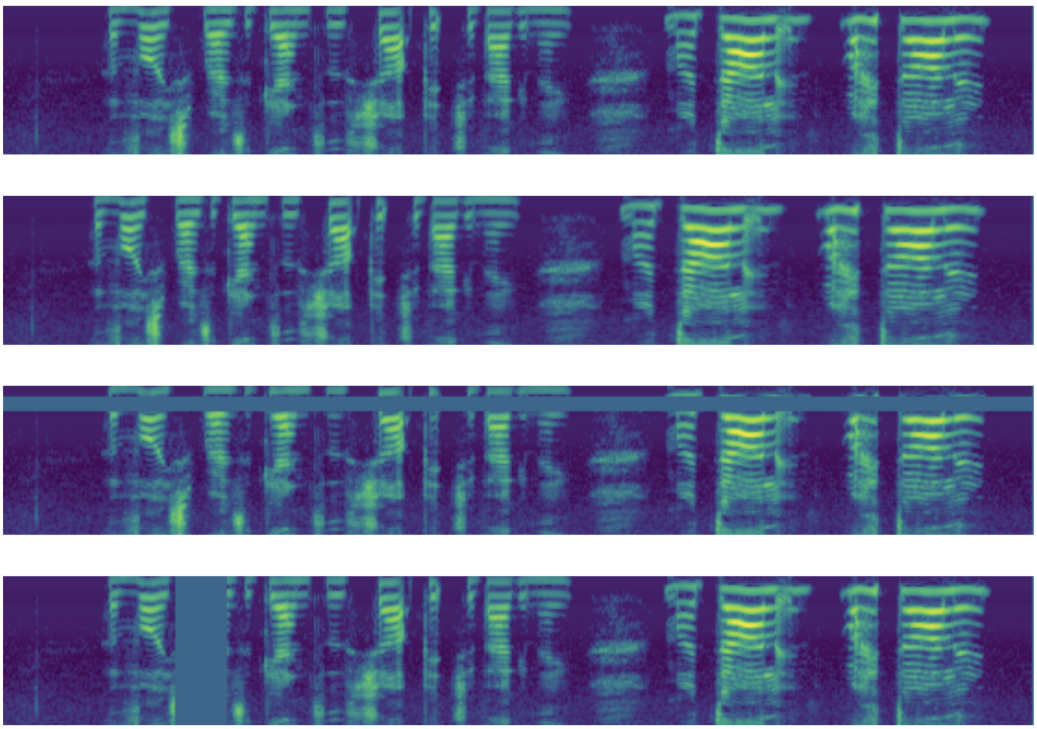

Figure 2.4: Augmentations applied to the base input, given at the top. From top to bottom, the figures depict the log mel spectrogram of the base input with no augmentation, time warp, frequency masking and time masking applied [Park et al., 2019].

The Figure 2.4 illustrates each augmentations applied to a single input.

The basic augmentation policies could be combined used on a single input. For example, the author of Park et al. [2019] introduced four new augmentation policies by combining the basic policies for their tasks.

SpecAugment has shown to be a practical approach to improving the WER on LibriSpeech and Switchboard, and it has the advantages of not generating new raw data, which saves the computational cost. Also, Park et al. [2019] showed that the ASR

system's performance trained used simple handcrafted policies surpassed the performance of hybrid systems, even without a language model's aid. Most importantly, SpecAugment has the advantages of converts ASR from an over-fitting to an under-fitting problem, and the performance could be gained by using bigger networks and a longer training process. However, this could also be seen as a disadvantage as it could take much longer for the model to fit the data. This would require even more time when the dataset is large, although Park et al. [2020] showed specAugment yielded more considerable improvements on large scale datasets than time-tested and other more sophisticated augmentation methods.

Chapter 3

Task, Dataset and Related Work

3.1 Task

This project's task is to build on the speech recognition models in our previous project [Jin, 2020] to find the best performance model using various data augmentation techniques. The effects of each data augmentation techniques would be investigated and the effects of different combinations between various data augmentations. In particular, the following data augmentation would be investigated:

- Speed Perturbation
- Volume Perturbation
- Noise Injection (Noise Training)
- SpecAugment

Note that VTLP and Tempo Perturbation would not be investigated due to the time limit. The decision was also made because some research [Geng et al., 2020, Ko et al., 2015] already show that the speed perturbation system performs better than VTLP and Tempo Perturbation.

The performance of the model would be evaluated on the medical testing set using standard evaluation metric Word Error Rate (WER) [Jurafsky and Martin, 2009]:

$$WER = \frac{S+D+I}{N} * 100\%,$$

where S stands for substitution error, D stands for deletion error, I stands for insertion error, and N stands for the number of words in the reference.

The lower WER means a better performance of a model as it has fewer errors on the predictions. The performances of different models would be compared by comparing the WER of the models. A model with the lowest WER would be selected as the final model in this project.

3.2 Dataset

3.2.1 Medical Dataset

In this project, the same three datasets would be used as in our previous project [Jin, 2020]. The first dataset is the medical dataset provided by the NHS, consisting of 50 narrowband recordings of approximately 3 hours of emergency calls and the corresponding transcriptions. The speakers are medical dispatcher and the bystander of patients. As this dataset is limited, it would not be possible to train an ASR system using the dataset directly. A significant improvement in the performance was achieved by using half of the whole medical dataset in the previous project. The importance of the "in domain" dataset would be further discussed in this project. As a recap, the dataset was split evenly into a training set and a testing set with 25 recordings each and a length of $\approx$ 1.5 hours each (1.6 hours for training and 1.4 hours for validation set to be more precisely).

The splitting process is done randomly. In this work, Kaldi script[1] is used for the random split by setting the parameter cv_spk_percent to 50. The random split ensures both datasets are unbiased. Also, it was assumed that there are only two speakers in one recording and 100 speakers in total, even though the same medical dispatcher could appear in multiple recordings. There are and 19725 words in the medical testing set, and this figure would be used when calculating the WER.

Note that the same testing set would be used for many experiments. This would violate the testing set's nature as the testing set should be kept away and used only once at the end. However, there is a trade-off between maximising the model's performance and the validity of the methods. The testing set is needed to check the models' performance and investigate data augmentation techniques' effectiveness. It could be possible to alternatively split the testing set into a validation set and a smaller set of testing sets, but this would cause both the validation and testing set to be too small. It would add more risks to the methods compared to a larger testing set that was used more than once. Using one larger testing set enables a direct comparison of the performance to the previous project's best model.

3.2.2 AMI Dataset

The second dataset used in the project is AMI Meeting corpus [Carletta et al., 2005]. The AMI Meeting Corpus is a multi-modal data set consisting of approximately 100 hours of wideband meeting recordings. The language spoken is English, but the speakers are primarily non-native speakers. It is the primary dataset that is used for building the ASR system. There are two versions of the dataset where one is recorded using an individual headset microphone (IHM), and another is using a single distant microphone(SDM1). Both versions have approximately 100 hours of recordings, but the number of recording varies. In this project, only the IHM dataset would be used throughout the experiments as our project work [Jin, 2020] had shown that IHM con-

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/utils/nnet/subset_data_tr_cv.sh

stantly performs better than the SDM dataset. IHM dataset has 682 recordings, and it is split into 547 recordings of $\approx$ 77 hours of the training set, 72 recordings of $\approx$ 11.5 hours of validation set and 63 recordings of $\approx$ 11.5 testing set. Also, note that the number of the speaker is unknown in a conversation, but a naive assumption is made which assumes a new speaker every 30 seconds in IHM dataset, so 10492 speakers in the IHM training dataset, 1197 speakers in the validation set and 1166 speakers in the testing set. The split and speaker approximation could be made running the AMI recipe [2] in Kaldi. AMI corpus is used as it is considered to be the best existing dataset available, and it is a large corpus that is suitable for training DNN. Most importantly, there exists a Kaldi recipe that builds a state-of-art ASR system using AMI corpus.

3.2.3 MUSAN Dataset

The last dataset that would be used is the MUSAN dataset. Musan is originally an 11 GB corpus consisting of music, speech and noise. Due to the computation environment provided, only the noise part of MUSAN is downloaded, which is approximately 1GB size of noise data and consists of 930 different noises. The total length of recordings is unknown, but it is not important as they would be randomly selected and used for injection. The different types of noise were all sampled from real life.

3.3 Related Work

As mentioned in chapter 1, the study in Blomberg et al. [2019] shows the speech recognition technology has been used, combined with machine learning framework, to predict whether the emergency call is the case of cardiac arrest. This is the project's primary motivation for building a speech recognition for the medical team, but their architecture of the model could not be followed as the magnitude of available medical data is very different, although they have not mentioned their ASR model's details at all in their report.

In our previous project [Jin, 2020], we found that using a small amount of medical data could significantly improve the performance of the model. However, collecting more medical data is difficult due to many reasons, such as privacy. Thus, data augmentations are motivated in this work as those techniques could generate synthetic data for increasing the variety of the data. Many related works showed using data augmentation techniques could improve the performance of the models.

Geng et al. [2020] carried out a systematic investigation of different data augmentation techniques (i.e.VLTP, Tempo and Speed Perturbation) and had shown Speed Perturbation improved the absolute WER reduction of 1.51% over the best-published system on UASpeech at the time. However, the performance was evaluated on the disordered speech recognition, which is a different type of speech recognition system in this work – Large Vocabulary Continuous Speech Recognition (LVCSR). Due to the intrinsic differences in the systems, the effects of Speed Perturbation could vary. In comparison, similar experiments were carried out for investigating the effects of

[2] https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/run.sh

VTLP, Tempo and Speed Perturbation on LVCSR in Ko et al. [2015]. Speed Perturbation, which emulates both VTLP and Tempo Perturbation, was shown to have larger WER improvement again than the other two. However, the performance was evaluated on the Gale Mandarin test set, and the model used more than 100 hours of Gale Mandarin training data. This differs from the scenario in this paper as the test set is not in the same domain as the training set, and only a small amount of in-domain data is used. In Gauthier et al. [2016], the paper investigated the Speed Perturbation on two lower resources languages: Hausa and Wolof. However, the paper showed that the Speed Perturbation only had minor improvements (0.6%) on Wolof's test set and no improvements on Hausa. Although many studies have shown that Speed Perturbation usually improves ASR models' performance, Speed Perturbation's amount of effects remains interested in this work.

Volume Perturbation is another commonly used data augmentation technique as Speed Perturbation. As mentioned in section 2.4, many studies had shown the Volume Perturbation has consistent improvements on the ASR system [Nguyen et al., 2018] and had relative 1.5% improvements on the system that uses Volume Perturbation, comparing to the system that uses Speed Perturbation only [Peddinti et al., 2015]. However, the effects of combining Volume Perturbation with other data augmentation techniques other than Speed Perturbation remain unknown. Also, in Peddinti et al. [2015], the effect of using Volume Perturbation solely was not investigated. Those effects would be investigated and discussed in this work.

Building a noise-robust ASR system is a popular topic that has been studied over many years. Noise Injection, also known as Noise training, is also a popular and effective data augmentation techniques for building a noise-robust system shown in many works. The advantage of noise training has been discussed in section 2.5. In Yin et al. [2015], the paper proposed the noise training approach for DNN-based ASR. The paper concluded that the noise pattern was effectively learned, and the generalisation capability of the learned DNN could be improved by injecting a moderate level of noise in the training data. It also claimed that the noise training approach could effectively learn multiple types of noises and noise injection at a moderate range of SNRs deliver further gains in the performance. Also, in Ko et al. [2017], the paper showed that a considerable improvement could be obtained in the close-talking scenarios by combining clean and reverberated training data. The effects of the noise training would be further confirmed in this work.

One other related work to the noise injection is the Denoising Autoencoder (DAE) approach [Vincent et al., 2008, Maas et al., 2012], trying to recover the original clean signal. Both noisy training and DAE corrupt the DNN input by randomly sampled noise. This approach is not considered in this work as the noise training approach is simpler to implement and effective.

There was another multi-condition training [Yu et al., 2013] related to the noise training in this work, in the sense that both approaches are training the DNN with speech signal in multi-condition. The difference is that the multi-condition speech in this work is obtained by injecting noise into clean speech, whereas [Yu et al., 2013] obtained the multi-condition speech from real life. Their approach is not feasible in this work as the

available data is limited in this project.

After the proposes of SpecAugment in [Park et al., 2019], there are growing researches on the effects of SpecAugment. SpecAugment is also applied in other research domains such as Machine translation [Bahar et al., 2019] and Automatic Speaker Verification System [Faisal and Suyanto, 2019]. As mentioned previously, SpecAugment is a data augmentation technique that operates on the feature level of the data and shown to have significant improvements on the performance [Park et al., 2019, 2020]. However, the effects of combining SpecAugment and other traditional data augmentation techniques such as Speed Perturbation remain unknown.

Lastly, another related work was a study of racial disparities in ASR [Koenecke et al., 2020]. As the result of the study, the author found that even state-of-the-art ASR systems developed by big companies such as Apple, Google and Amazon have racial disparities in ASR, with a WER of 0.35 for African-American speakers compared with 0.19 for white speakers. The author has proposed using the more devised training dataset, which consists of African American Vernacular English, to address the problem. It inspired us to use a dataset with a good diversity of English speakers, consisting of native and non-native speakers.

Chapter 4

ASR Implementation using Kaldi

As same as our previous work [Jin, 2020], Kaldi [Povey et al., 2011] is again used for the implementation in this work. As a recap, Kaldi and the baseline's implementation details are introduced again in this chapter.

4.1 Kaldi

Kaldi[1] is an open-source ASR toolkit that is intended for the uses of teaching and researching. Many of the state-of-art ASR systems are implemented on the top of Kaldi. Thus, Kaldi also contains the scripts for building many state-of-art ASR systems. As shown in Sučík [2019], Readers should understand the following three components in Kaldi for building the ASR system:

- **Kaldi Binaries** is a set of functions mostly written in C++ that perform particular tasks (e.g. computing MFCC and CMVN). These binaries can be bind together, which makes Kaldi powerful and flexible. The binaries are stored in someplace else than the working directory. An environment variable KALDI ROOT is set (inside the file path.sh) to point to the Kaldi installation, and this is added to the system path to access them from anywhere.
- **Kaldi scripts** consists of multiple Kaldi Binaries, and it is the high-level implementation of a particular task, such as data augmentation, model training or decoding. Kaldi provides default hyperparameters in the scripts, which experimentally shown to perform well. The script could be adapted to fit into the choice user prefers.
- **Kaldi recipes** consists of Kaldi Binaries and Kaldi scripts, and it corresponds to a version of a particular study or experiment carried in the past. It is contained in the **egs** folder in the Kaldi directory, and it always contains a script **run.sh** which consisting a sequence of scripts needed to run through for building a corresponding ASR system of the study in one go.

[1] `http://kaldi-asr.org/`

As mentioned above, this work would continue to use the ASR system's implementations from the previous work. Therefore, the entire implementation of the ASR system is again based on a single recipe of **ami (version s5b)**[2] with modifications to some of the scripts. Version **s5b** simply means a simplified streamline of original one [3].

4.2 Dictionary

The AMI recipe reproduces the model in [Swietojanski et al., 2013] that uses CMUDict [Carnegie Mellon University, 2019], CMUDict is an open-source dictionary for North America English that contains more than 134,000 words and their pronunciations, which is suitable for uses in speech technology and is maintained by the Speech Group in the School of Computer Science at Carnegie Mellon University.

The dictionary has mapping from words to their pronunciations in the ARPAbet phoneme set [4], a standard Phoneme Set for English pronunciation. The current phoneme set contains 39 phonemes, vowels carry a lexical stress marker:

- 0 — No stress;
- 1 — Primary stress;
- 2 — Secondary stress;

Below shows a few examples from the Phoneme Set:

Phoneme	Example	Translation
AH	hut	HH AH T
ER	hurt	HH ER T
NG	ping	P IH NG

4.3 Language Model

The language model used in the AMI recipe is built from both AMI corpus and Fisher English Training Speech Part 1 Transcripts [Cieri et al., 2004a,b].

The Fisher transcripts represent the first half of a conversational telephone speech collection, which contains the time-aligned transcript data for 5,850 complete conversations, each lasting up to 10 minutes.

The AMI recipe trained two separate n-gram language models for both AMI corpus and fisher transcripts first, then apply Interpolation [Pusateri et al., 2019] and Pruning [Goodman and Gao, 2000] to yield a better-mixed language model as the final language model for the ASR system. The N-gram order is 3 by default during the training.

[2] `https://github.com/kaldi-asr/kaldi/tree/master/egs/ami/s5b`
[3] `https://github.com/kaldi-asr/kaldi/tree/master/egs/ami/s5`
[4] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

4.4 Baseline model

The baseline model in this work is the model with the best performance in the previous work, which is built on top of the AMI recipe. The main steps of building the baseline would be the same as our previous project [Jin, 2020]:

1. Prepare the AMI training dataset, language model (AMI+Fisher Corpus), and lexicon (CMUDict) using run_prepared_shared scripts[5]); Note the language model and lexicon are chosen mainly due to the convenience as they are already in use with AMI recipe. The original AMI recipe has achieved excellent results with the Fished Corpus and CMUDict.
2. Extract MFCC features from AMI training dataset and apply the Cepstral Mean-Variance Normalisation techniques to MFCCs. Note that the training dataset's signal is sampled at 16 kHz, but the medical dataset's signal is sampled at 8KHz. Therefore, a modification to the configuration is needed when making MFCC features [6] for downsampling the training dataset to 8KHz and making the model works properly on the 8KHz medical data. Additionally, another downsampling approach could be using a script [7] that uses sox (utils/data/resample_data_dir.sh) to resample the original signal at 8 kHz;
3. Build a monophone model [Jurafsky and Martin, 2009] without delta feature and align the training dataset. Note that the training dataset is aligned each time after a new model is built;
4. Build a triphone model with delta feature [Jurafsky and Martin, 2009];
5. Build a triphone model with delta and delta-delta; features;
6. Build a triphone model with LDA and MLLT [Povey et al., 2011];
7. LDA+MLLT+SAT [Gales et al., 2008];
8. Clean the training dataset by removing the bad alignments;
9. Combine the medical dataset together with the AMI dataset using combine_data.sh [8]. Note that the MFCC feature and high-resolution MFCC features [Jurafsky and Martin, 2009] of the dataset were already provided.
10. Augment the training dataset by using Speed Perturbation with speed factors 0.9, 1.0, and 1.1, which triples the size of the original training dataset. For volume perturbation, each audio is scaled with a random value drawn from a uniform distribution [0.125, 2]; Compute the high-resolution MFCC of the augmented dataset.

[5] https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/run_prepare_shared.sh

[6] https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/conf/mfcc.conf

[7] https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/utils/data/resample_data_dir.sh

[8] https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/utils/combine_data.sh

11. Apply the PCA to high-resolution MFCC; build UBM and extract i-vectors [Dehak et al., 2010, Karafiát et al., 2011];

12. Train the final DNN-HMM model. The DNN-HMM models i-vectors extracted from high-resolution MFCC of the training data and MFCC features for the training process; the decision tree and alignment are obtained from the SAT GMM system. The type of DNN is TDNN [Peddinti et al., 2015]. The type of DNN-HMM model is Chain Model[Povey et al., 2016]. The objective function used in training is MMI [Raj, 2019]. The configuration of the hyperparameters such as L2 regularisation parameters (avoid overfitting) in the neural network is kept the same[9].

13. Decode the medical test set by Neural Network based online decoding with i-vectors. The unnormalised high-resolution MFCC and i-vectors used as input to the neural network. The idea is that i-vectors provides information to the neural network about the speaker's property, which is proved to be helpful for the decoding [Povey et al., 2011]. Note that the original script's scoring process requires the STM file, which is not provided for medical data. Hence we replace the original scoring script (/local/score.sh) by another scoring script (/steps/score_kaldi.sh).

The baseline model's WER performance is 45.21%, and it is trained on 235.5 hours (77 hours of AMI + 1.5 hours of medical data) after Speed Perturbation and Volume Perturbation. This is chosen to be the baseline model because a better WER model than the best performing model in the previous work is desired at the end of the investigations.

The further experiments would follow similar procedures as the baseline model. All of the further investigations except the SpecAugment are based on modifying the training dataset for the DNN training. This means that only steps after step 8 would be needed for each experiment in this work.

4.5 Computation environment

All the scripts are mainly run on the MLP GPU Cluster of the School of Informatics [10], which is made of multiple GPU boxes (nodes) where each box consists of 8 x 1060 Ti GTX GPUs. The cluster is combined with the use of cluster schedule software **Slurm** [Yoo et al., 2003] to enable the user to submit their jobs when requiring GPUs.

The time taken for training the whole baseline model was approximately three days. It would take about one day for the training procedures from step 1 to step 8. For the DNN training after step 8, it would cost approximately two days for training 235.5 hours of data for 15 epochs.

Three CPU servers (Zamora,Tunguska,Lubbock) and one GPU server (Giger)[11] are

[9] https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/local/chain/tuning/run_tdnn_1j.sh

[10] http://computing.help.inf.ed.ac.uk/teaching-cluster

[11] http://www.cstr.ed.ac.uk/internal/howtos/computersupport/

also provided as the additional computation environment for running the tasks when the MLP cluster is bustling. Each CPU server consists of 4x8-core AMD Opteron 6325 CPUs with 512GB RAM and plenty of available scratch disk. The Giger server consists of 4x GTX 980 with 32GB Memory. Commands **rsync** is used to move the files between the servers. Fortunately, as the MLP cluster is not very busy during the investigations, only CPU servers are often used in this work as many computations rely heavily on the CPU. It would take much longer to run the CPU tasks on the MLP cluster. For example, computing the high-resolution MFCC features for the noise injected data would take over two days on the cluster but approximately less than 12 hours on the CPU server.

Chapter 5

Experiments & Results evaluation

5.1 Plain model

As mentioned earlier, the baseline model is the model with the best performance in the previous work. The baseline model is a model that contains both AMI and medical dataset and has Speed Perturbation, and Volume Perturbation applied. As a result, the baseline model achieved a large improvement of WER (78.63% to 45.21%) from the baseline model in our previous work [Jin, 2020]. This improvement is contributed by both the usages of in-domain data and data augmentation. However, the individual effects of the usage of in-domain data and each augmentation technique are unknown. In order to have a better insight into how each augmentation affects the performance, it would be essential to have a model that only uses the medical data but has no augmentation techniques applied. The result is shown in Table 5.1:

MODEL	BASELINE	PREVIOUS BASELINE	PLAIN
WER	45.21%	78.63%	49.00%

Table 5.1: a table showing the WER of the baseline, previous baseline and plain model on the medical testing dataset (4 s.f.)

As Table 5.1 shows, the usages of 1.5 medical data significantly increased the performance from 78.63% to 49.00%, comparing the model that uses only AMI corpus but with Speed Perturbation and Volume Perturbation applied. This demonstrates the importance of the in-domain data. A much more significant improvement could be expected if more in-domain data is available.

As discussed above, collecting more data would not always be optional. The baseline model that used both Speed Perturbation and Volume Perturbation improved the performance from 49.00% to 45.21%. Almost 4% of improvements by solely using Speed Perturbation and Volume Perturbation, without having the cost of collecting any new data, could be regarded as solid evidence that data augmentations are an effective approach for improving the model's performance when only limited data available.

However, the effects of each augmentation technique are unknown yet. It could even be

the case that one augmentation is improving the performance while another is degrading. We need to investigate further how Speed Perturbation and Volume Perturbation separately affect the model's performance, shown in the following sections.

Note that some readers could be confused about the reasons why the plain model was not chosen as the baseline in this work instead of the model with the best performance in our previous work. The decision is made because a better performance model than our previous work is intended in this work, rather than a better performance model than the plain model. Also, some readers could argue that the baseline should be the simplest achievable model, but the model from the previous work is indeed the simplest achievable model as it already exists without carrying out any experiments.

5.2 Speed Perturbation and Volume Perturbation

For investigating the individual effects of Speed Perturbation and Volume Perturbation, two experiments needed to be carried: the model uses Speed Perturbation solely, and the model uses Volume Perturbation solely. However, due to the different mechanisms in two augmentation techniques, Speed Perturbation yields more data for the training. If both augmentations happened to improve the performance, but with larger improvements using Speed Perturbation, the results could be biased as Speed Perturbation has more data than Volume Perturbation. The improvements could be due to the amount of data used instead of the augmentation technique's effects. Therefore, to make a better distinguishment, an additional experiment is carried out for Volume Perturbation by manually clone the original dataset to make the size three times larger, so it is comparable to the Speed Perturbation. The increased dataset would then be augmented using Volume Perturbation by scaled with a random value drawn from the uniform distribution [0.125, 2]. The speed factors for the Speed Perturbation are 0.9,1.0,1.1. The results are shown in Table 5.2:

Model	Amount of data (hours)	WER
Previous baseline	77*3 = 231	78.63%
Baseline	(77+1.5)*3 = 235.5	45.21%
Plain	(77+1.5) = 78.5	49.00%
Speed Perturbation only	(77+1.5)*3 = 235.5	45.27 %
Volume Perturbation only	(77+1.5) = 78.5	48.55%
Volume Perturbation (x3)	(77+1.5)*3 = 235.5	79.56%

Table 5.2: a table showing the WER of the baseline, previous baseline, plain model and the models using only Speed Perturbation and Volume Perturbation, on the medical testing dataset (4 s.f.)

The Table 5.2 shows that Speed Perturbation improved the plain model from 49.00% to 45.27%, which is very close to the baseline performance. This suggests that Speed Perturbation contributes the most in the improvements from the plain model to the baseline. Volume Perturbation also improves the performance but relatively much smaller than Speed Perturbation. It has approximately 1% relative improvements from the plain to the model that uses Volume Perturbation only. It also has minor improvements

when combining Speed Perturbation and Volume Perturbation (from the model uses Speed Perturbation only to the baseline model). Surprisingly, it seems that overfitting occurred in the model that uses Volume Perturbation with increased data. The performance dropped from 49% to 79.56%, which is even worse than the previous baseline (78.63%).

Few things can be deduced from the results above. Firstly, the approach of manually increasing data size by cloning and applying with the Volume Perturbation is not feasible, as it causes overfitting. Secondly, Speed Perturbation seems to be a more powerful augmentation technique than the Volume Perturbation in terms of "improving the performance". There might be few reasons for both improvements by using Speed Perturbation and Volume Perturbation:

- AMI data is the recordings of conversations in the meeting scenarios, whereas medical data is the recordings of emergency calls. Thus, the speaking rates could tend to be slower in the AMI data than medical data. As the model is trained mostly by the AMI data and Speed Perturbation could increase AMI data's speaking rates by a factor of 1.1, this might make AMI data match better to the in-domain data in terms of speaking rates. Thus, it results in a large performance improvement.
- The i-vector-based TDNN system is based on the mean shift captured in the i-vector, as the neural network needs to learn the necessary normalisation based on this. However, the AMI data are recorded with headsets, and the recordings are phone calls in medical data. There is a low variance in i-vector w.r.t mean shifts in the well-recorded audio dataset. Therefore, performing volume perturbation of the training data could emulate the mean shifts in the MFCC domains. This enables the neural network to learn the normalisation better and thus improves the performance.

Although Speed Perturbation improves the performance more than Volume Perturbation, Volume Perturbation has the advantage of simplicity. Volume Perturbation does not increase the data's increasing size, which means the training time would remain almost the same by applying Volume Perturbation, although the MFCC features need to be re-computed after applying the Volume Perturbation.

Due to the factor that Speed Perturbation is the main contributor to the improvements for the baseline model, further research is carried out to investigate the Speed Perturbation in more detail.

5.3 Analysis of Speed Perturbation Results

As discussed above, the large improvement using Speed Perturbation could be due to the differences in speaking rates between AMI data and the medical data. Speed Perturbation is explored more in this section. Speaking rates are expressed in terms of words per seconds (WPS) in this work:

$$WPS = \frac{\text{Total Words}}{\text{Total Length}} \tag{5.1}$$

WPS could be expressed in different units. For example, WPS in the unit of a recording would count the total words that appeared in the recordings and divide the number by the recording's total duration.

For the interest of knowing the distribution of speaking rates in AMI and medical dataset, two histograms of WPS are plotted in Figure 5.1. The average speaking rates are also computed and shown in table Table 5.3.

DATASET	MEAN (25 RECORDINGS)	STD (25 RECORDINGS)
AMI	3.06	0.40
MEDICAL	3.58	0.23
MEDICAL/ AMI		1.17
DATASET	MEAN (995 SEGMENTS)	STD (995 SEGMENTS)
AMI	3.41	1.37
MEDICAL	3.55	1.00
MEDICAL/ AMI		1.04
DATASET	TOTAL WORDS / TOTAL LENGTH	
AMI	2.86	
MEDICAL	3.57	
MEDICAL/AMI		1.25

Table 5.3: a table showing the mean and standard deviation of speaking rates in AMI and medical dataset. Speaking rates are expressed in the terms of words per second and it could be calculated differently depending on the units: unit of a recording, unit of a segments and in total. The ratios between WPS of Medical and WPS of AMI with different units are also shown in the table.

As shown in Figure 5.1, both plots of histogram show that the speaking rates of medical data are more distributed at the right side of the plots than the AMI data distributed at the left side of the plot. Table 5.3 shows that the average speaking rate of medical data is consistently higher than the AMI data across different units. Also, the standard deviations of WPS for the medical set are consistently smaller than the WPS for the AMI dataset. Hence, it is possible to conclude that speaking rates in the medical data is faster than the speaking rates in the AMI data.

The previous Speed Perturbation uses perturbation factors of 0.9,1.0, and 1.1. However, according to the WPS findings above, the medical data is 1.17 faster than the AMI data in the unit of recording and 1.04 faster in the unit of segments. This opens some questions for researching on Speed Perturbation: **would it be better to use the perturbation factor according to the ratios of mean WPS found (i.e. 1.17 and 1.04), or would it better to use the standard perturbation factor of 1.1? Also, would speed up the audio have better improvements than the slowing down (perturbation factor of 0.9)**?

Therefore, in order to further investigate the Speed Perturbation, the effects of different perturbation factors are explored, and the results are shown in Table 5.4

By looking at the results of Table 5.4, there are few important findings that could be drawn from it:

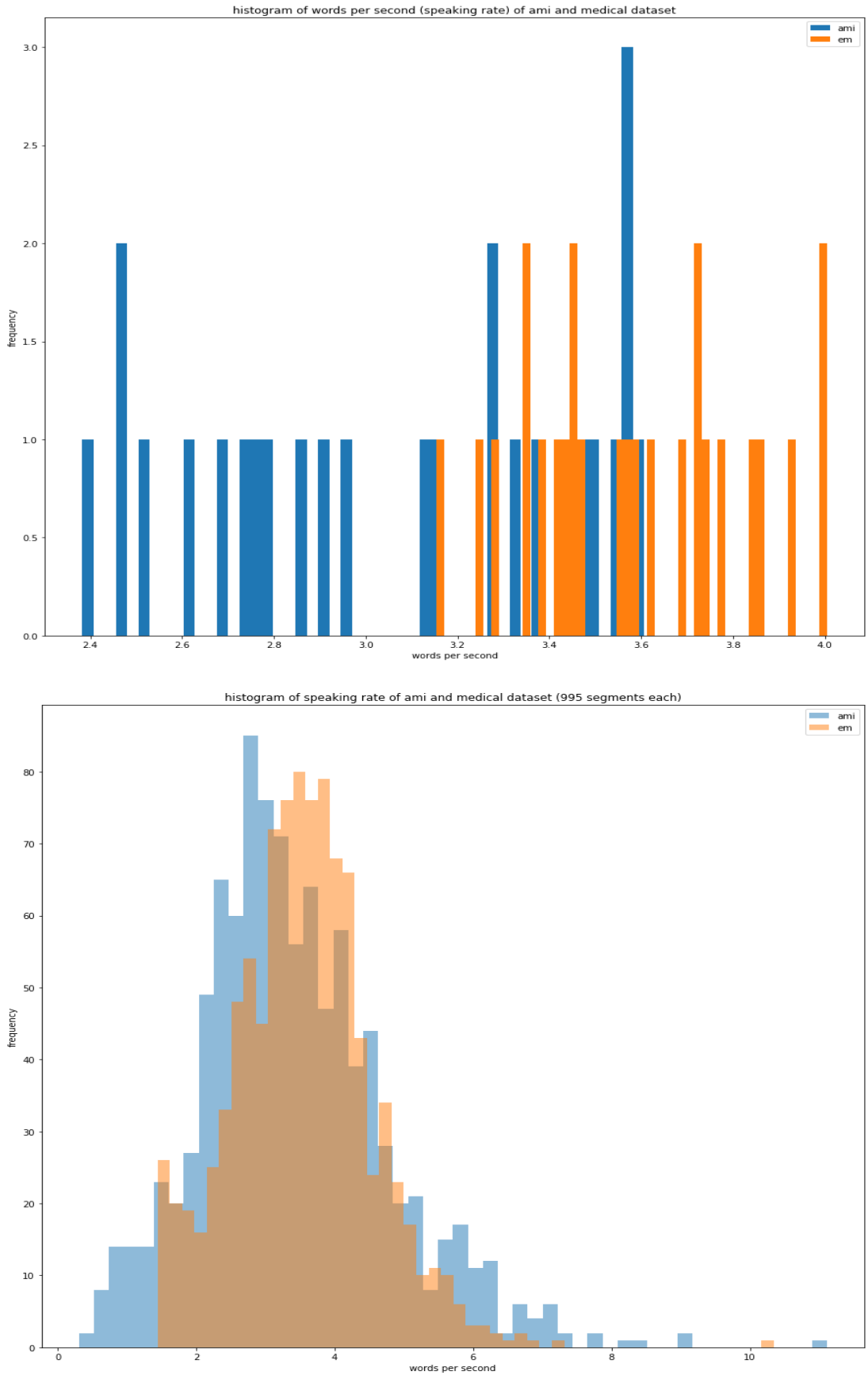

Figure 5.1: Two histograms are showing the speaking rates of AMI and medical dataset expressed in the term of WPS of unit recordings (top) and segments (bottom). As there are only 25 recordings and in total 995 segments for the medical data, 25 AMI recordings and 995 segments are randomly selected from the AMI dataset for computing and comparing WPS.

Model	Speed Perturbation factors	Amount of data (hours)	WER
(1)	(1.0)	(77+1.5)=78.5	48.55%
(2)	(0.9, 1.0)	(77+1.5)x2 = 157	46.57%
(3)	(1.0, 1.1)	(77+1.5)x2 = 157	46.20%
(4)	(1.0, 1.04)	(77+1.5)x2 = 157	46.31%
(5)	(1.0, 1.17)	(77+1.5)x2 = 157	46.34%
(6)	(0.9, 1.0, 1.1)	(77+1.5)x3=235.5	45.21%
(7)	(1.0, 1.1, 1.17)	(77+1.5)x3=235.5	46.10%
(8)	(1.0, 1.1, 1.20)	(77+1.5)x3=235.5	45.37%
(9)	(0.8, 0.9, 1.0, 1.1, 1.2)	(77+1.5)x5=392.5	45.08 %

Table 5.4: a table showing the WER of the models with different Speed Perturbation factors (and Volume Perturbation applied), on the medical testing dataset (4 s.f.). Model (1) is the model with only Volume Perturbation applied and Model (6) is the baseline.

- Speed Perturbation consistently improves the performance of ASR across all different perturbation factors used.
- Solely speeding up the audios consistently improves more than solely slowing down the audio as shown in the results of the models (2), (3), (4) and (5). This could be due to the reasons discussed above: medical data has a faster speaking rate than AMI, speeding up the AMI could better match the training data to the testing data.
- The standard perturbation factor of 1.1 improves more than the ratio found (1.04, 1.17) previously, as shown in the results of the models (3), (4) and (5).
- The standard combinations of perturbation factors (0.9, 1.0, 1.1) that includes both speeding up and slowing down the audios improves better than only speeding up the audios shown in the results of the models (6), (7), (8), although the model (8) has the performance very close to the model (6). This might be because the standard combination generates the synthetic data that mimics the testing data better.
- Using more perturbation factors that generate even more synthetic data would further improve the performance, as shown in the result of the model (9).

Overall, Speed Perturbation is an effective data augmentation technique for improving the performance of the ASR system. Although model (9) achieved the best results of 45.08% comparing to the models with other speed perturbation factors, further experiments would be based on model (6) (the baseline) due to the computation efficiency.

5.4 Noise Injection

The noise injection experiments would be similar according to the procedures described in our previous work [Jin, 2020]. The noise used for the injection is the foreground noise extracted from the MUSAN Noises dataset [Snyder et al., 2015]. Foreground noise means prominent noise such as alarm and the sound of the phone

ring. The script used for the injection is the partial codes [line 148 - line 179] extract from the run.sh of SRE recipe [1]. Overall, the process of noise training could follow as:

1. Resample the clean training data at 8KHz using a script (/utils/data/resample-datadir.sh [2]).
2. Make a copy of the resampled clean data.
3. Inject the noise from Musan Noise dataset into the clean data to create noisy data. The noise injected with a random variable SNR is randomly selected from 4 SNRs of [15,10,5,0].
4. Combine the noisy data and clean data to create a new training dataset.
5. Follow the same steps from step 11 described in the baseline implementation.

The effects of noise injection are investigated together with other data augmentation. The results are shown in table Table 5.5.

MODEL	S.P. (0.9,1.0,1.1)	V.P. [0.125,2]	N.I [15,10,5,0]	DATASIZE (HOURS)	WER
I	✔	✔	✖	(77+1.5)*3 = 235.5	45.21%
II	✖	✖	✖	(77+1.5) = 78.5	49.00%
III	✔	✖	✖	(77+1.5)*3 = 235.5	45.27%
IV	✖	✔	✖	(77+1.5) = 78.5	48.55%
V	✖	✖	✔	(77+1.5)*2 = 157	46.51%
VI	✔	✖	✔	(77+1.5)*2*3 = 471	43.96%
VII	✖	✔	✔	(77+1.5)*2 = 157	46.37%
VIII	✔	✔	✔	(77+1.5)*2*3 = 471	**43.81%**

Table 5.5: a table showing the WER of the models with different data augmentation applied. S.P. stands for Speed Perturbation, V.P. stands for Volume Perturbation and N.I stands for Noise Injection. Model I is the baseline and Model II is the plain model.

The important findings that could be drawn from the Table 5.5 are:

- As shown in the WER of V, using only Noise injection could improve the plain model about absolute 2.49%, which is a large improvement, and it is quite close to the improvement of 3.75% (III) on the plain model that uses only Speed Perturbation.
- Volume Perturbation is compatible with Noise Injection as shown by the improvements from 46.51% (V) to 46.37 % (VII).
- Speed Perturbation works very well together with Noise Injection as shown the improvements from 45.27% (III) and 46.51% (V) to 43.96% (VI). This is a large improvement.

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/sre16/v1/run.sh

[2] https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/utils/nnet/subset_data_tr_cv.sh

- All three data augmentation techniques are compatible and could be combined to improve performance as VIII achieved the best performance so far.

All three data augmentations are compatible because they augment the data in different levels, and they are not conflicting with each other.

Overall, noise training is a practical and feasible approach to improve the model's performance. Also, the advantages of noise training could be underestimated by the results shown above. This is because the recordings are done either by headset or phone calls. Noise training might have larger improvements if the audios are recorded some distances away from the sources and the recording condition is noisier.

5.5 SpecAugment

SpecAugment is the last augmentation that would be investigated in this work. The effects of different policies of SpecAugment could result very differently. However, due to the time limit, the effects of SpecAugment would be investigated using a single policy. The policy is the default policy of the implementation of SpecAugment in Kaldi. This is mainly chosen for the convenient purpose, as it is already implemented and experimentally shown to have good results by other researchers. The policy has a default value for the following variables:

- freq-max-proportion=0.5; freq-max-proportion is the maximum proportion of the frequency space that could be zeroed out
- time-zeroed-proportion=0.2; time-zeroed-proportion is the proportion of time frames that would be zeroed out.
- time-mask-max-frames=20; The maximum length of a zeroed region in the time axis, in frames.

The default policy in Kaldi implementation does not include one basic policy of Time warping introduced in [Park et al., 2019]. This might be due to the conclusion made in [Park et al., 2019], which suggested "Time warping contributes but is not a major factor in improving performance". However, as time-warping has similar effects of Speed Perturbation, combining SpecAugment with Speed Perturbatino would same effects of including all three basic policies.

Notice that MFCC would pass through Inverse Discrete Cosine Transform layer (idct-layer) to transform back to filterbanks representation before SpegAugment. It does not make any senses to take out a band of MFCCs like taking out a band of frequencies for SpecAugment. It would not be corresponding to any physical processes [3].

According to [Park et al., 2020], the paper found that combining SpecAugment and Multistyle Training would degrade the model's performance, which uses only Multi-style Training. Multistyle Training (MTR) is used to augment the input data by combining clean audio with a large library of noise audio using a mixed room simulator.

[3] https://groups.google.com/g/kaldi-help/c/VmF7WwSGSoI/m/ynnd1Ry1AwAJ

Therefore, MTR is similar to the Noise Injection in this work. Would SpecAugment work well with Noise Injection and other augmentations in this work?

A model with only SpecAugment applied is carried out at the first before any further investigations. The result is shown in Table 5.6:

MODEL	BASELINE	PLAIN	SPECAUGMENT ONLY
WER	45.21%	49.00%	49.40%

Table 5.6: a table showing the WER of the baseline, plain and the model with only SpecAugment applied (15 epochs) on the medical testing dataset (4 s.f.)

Surprisingly, the model with only SpecAugment applied degraded the WER by 0.4%. This could be because SpecAugment would turn an overfitting problem into an underfitting problem, and running the epoch number of 15 for 78.5 hours of data might not be sufficient. Thus, models applied by SpecAugment with running more epochs are investigated. Also, to better distinguish between the effects of epoch and SpecAugment, the plain model running with more epochs is investigated. The results are shown in Table 5.7

78.5 hours of data		
Epoch	Plain	SpecAugment
15	49.00%	49.40%
30	47.61%	46.55%
45	47.38%	45.39%
60	-	44.79%
90	-	44.29%

15 Epochs		
Model	Datasize (Hours)	WER
V	(77+1.5)*2 = 157	46.51%
III	(77+1.5)*3 = 235.5	45.27%
(9)	(77+1.5)x5=392.5	45.08%
VI	(77+1.5)*2*3 = 471	43.96%
VIII	(77+1.5)*2*3 = 471	43.81%

Table 5.7: Left table shows WERs of the plain model and the model with SpecAugment applied running with different Epoch number given 78.5 hours of data. The right table shows the WERs of the different model with different data size after the augmentation running 15 Epcohs. V is the model with the only Noise Injection applied. III is the model with only Speed Perturbation applied. Model 9 is the model with only Speed Perturbation applied, using speed perturbation factors (0.8, 0.9, 1.0, 1.1, 1.2). VII is the model with Speed Perturbation and Noise Injection applied, and VIII is the model with all others. The models running with equivalent data size are grouped into the same colour.

As Table 5.7 shown, SpecAugment could eventually improve the model's performance when running with more epochs. The improvements of using SpecAugment becomes more dominants with more epochs as the gap of improvements increases. Note that the plain model only runs up to 45 epochs as SpecAugment starts to outperform the plain model. SpecAugment of Epoch 30 has similar performance to the model with only Noise Injection applied while both models are trained with a similar equivalent amount of data. Also, the SpecAugment of Epoch 45 could achieve a similar performance as the model with only Speed Perturbation applied, and Epoch 90 could achieve similar performance as the model with all other three augmentations applied.

Moreover, it seems that SpecAugment is achieving better performance when running with 60 epochs comparing to the model (9), which is the model with only Speed Perturbation applied using perturbation factors of (0.8, 0,9, 1.0, 1.1, 1.2), while SpecAugment has a less equivalent amount of data for the training than the model (9). This might suggest that SpecAugment is better than Speed Perturbation when running with more data.

Lastly, it seems that the model with SpecAugment has not converged yet with epoch 90. However, further experiments would be expensive as 90 epochs already require almost a week ($\approx$ five days) for the training.

Therefore, SpecAugment is also an effective augmentation technique by itself for improving performance. Now, the effects of combining SpecAugment with other data augmentation would be investigated. However, due to the time limit, SpecAugment would only be combined with the Noise Injection and the current best model. Noise Injection is chosen to be investigated because the outcome could be compared to the results in [Park et al., 2020], which suggested Noise Injection might not be compatible with the SpecAugment. The results is shown in Table 5.8

MODEL	N.I [15,10,5,0]	SPEC.	EPOCHS	DATASIZE (HOURS)	WER
IX	✖	✔	60	(77+1.5) = 78.5	**44.79%**
V	✔	✖	15	(77+1.5)*2 = 157	46.51%
X	✔	✖	30	(77+1.5)*2 = 157	45.28%
XI	✔	✔	15	(77+1.5)*2 = 157	47.57%
XII	✔	✔	30	(77+1.5)*2 = 157	44.80%

MODEL	S.P.	V.P.	N.I	SPEC.	EPOCHS	DATASIZE	WER
VIII	✔	✔	✔	✖	15	471	43.81%
XIII	✔	✔	✔	✔	15	471	**43.67%**

Table 5.8: A table shows the effects of combining SpecAugment and Noise Training only (shown at the top) and combining SpecAugment with the current best model VIII (shown at the bottom).

Some key findings could be drawn from Table 5.8:

- Combining SpecAugment with Noise Injection degrades the performance from 46.51% to 47.57% when running with Epoch 15. This is consistent with the outcome in Park et al. [2020]. However, as the epoch number increased to 30, SpecAugment and Noise Injection starts improving the performance further comparing to the model solely use the Noise Injection by the results shown in X and XII.
- SpecAugment could be a more powerful augmentation technique comparing to the noise training. This is because the model running with 60 epochs using only SpecAugment performs better than the model running with 30 epochs using only Noise Injection while they have the same equivalent amount of data, shown in the results of IX and X. Also, model IX has similar performances (slightly but

not significantly better) to the model with 30 epochs using both Noise Injection and SpecAugment, as shown in IX and XII.

- SpecAugment could be combined with other augmentations to further improves the performance, as shown in VIII and IX.

Although the effects of combining SpecAugment with solely using Speed Perturbation or Volume Perturbation are not investigated, it improves the current best model VIII. Overall, the SpecAugment is a practical approach for improving the ASR system's performance, and it could be concluded as compatible with other data augmentations.

5.6 Final Model

The best choice of constructing the final model for this project would be building a model with all data augmentations applied, with speed perturbation factors of (0.8, 0.9, 1.0, 1.1, 1.2) and running with 90 epochs. However, such a model would take a month to train in the current computational environment.

Therefore, a model with all data augmentation applied running with 90 epochs but with the standard speed perturbation factors of (0.9, 1.0, 1.1) is chosen as the final model. This decision is made based on the fact that SpecAugment works better with more epochs and the improvements of using more epochs for SpecAugment is larger than the improvements of using more speed perturbation factors. Overall, it would take approximately two weeks for training the final model. Two weeks' time cost is very expensive, and the model's performance on the testing set is tracked and early stop the training if the overfitting occurred. The changes in the performance are shown in Figure 5.2. The final model's performance is recorded in Table 5.9 to better compare to other models.

MODEL	S.P.	V.P.	N.I	SPEC.	EPOCHS	DATASIZE	WER
I	✔	✔	✖	✖	15	235.5	45.21%
II	✖	✖	✖	✖	15	78.5	49.00%
XIII	✔	✔	✔	✔	15	471	43.67%
XIV	✔	✔	✔	✔	90	471	**42.53%**

Table 5.9: A table shows the effects of combining SpecAugment and Noise Training only (shown at the top) and combining SpecAugment with the current best model VIII (shown at the bottom).

By analysing the Figure 5.2 and Table 5.9:

- Fortunately, overfitting did not occur during the training over the long periods. It seems that further improvements could be gained with even more epochs.
- The final model could improve more than absolute 1% WER from the previous best model XIII. It is able to improve approximately absolute 2.68% WER from the baseline and about absolute 6.47% from the plain model. The improvement is huge, considering the fact that no extra in-domain data is used.

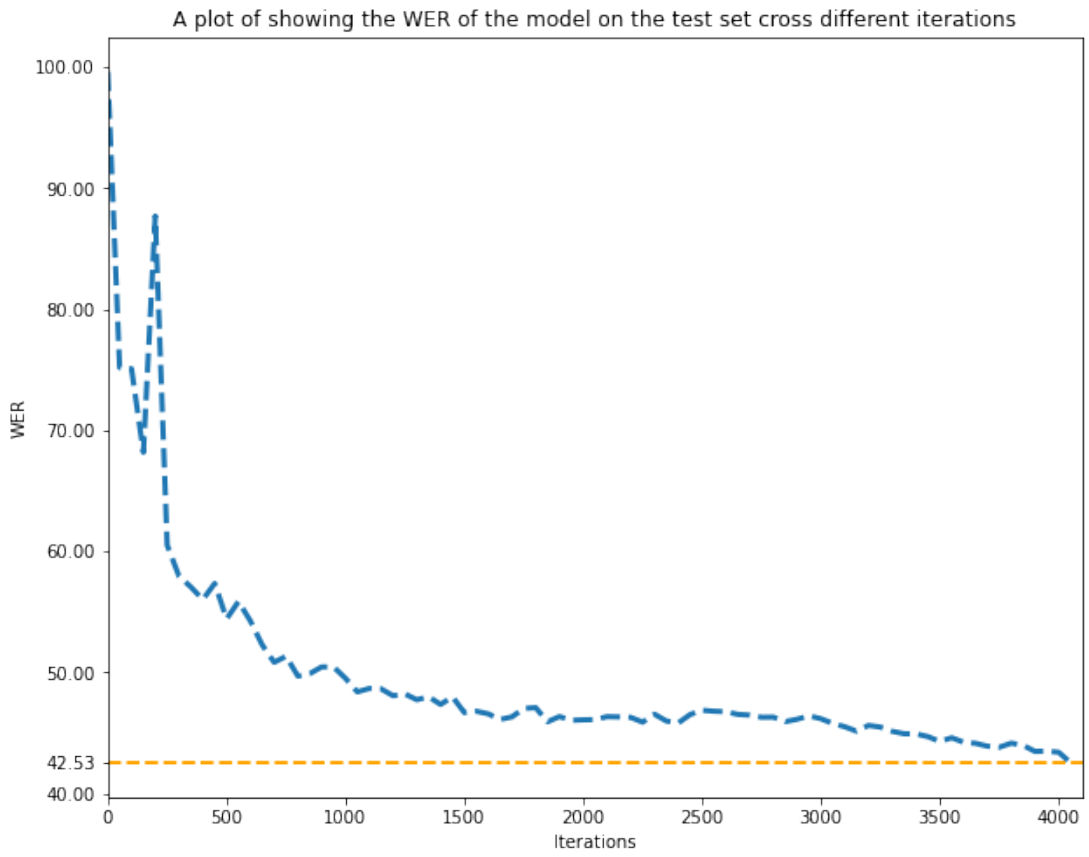

Figure 5.2: Noise is injected to a clean medical audio (top) resulting a noise audio (bottom)

Overall, a final model with a significant improvement from the baseline is achieved at the end. It is very risky and expensive to train a model for more than two weeks, considering only 78.5 hours of original data. It might be a better choice to gradually increase the epoch number for the model with all augmentations. However, this would cost more than a month to achieve the results. There is a trade-off between the risk and the chance of getting a model with the best performance. The decision was also made since overfitting did not occur for training the model with only SpecAugment applied with 90 epochs, thus overfitting is less likely to occur with more synthetic data.

5.7 Overall discussion on the experiments

There are many experiments carried out in this project. Some major outcomes should be emphasised in this section using Table 5.10.

The first and the most important outcome is that we found all the data augmentations investigated in this work could improve the ASR system' performance, although Volume Perturbation consistently has tiny improvements.

Secondly, in terms of the magnitude of the improvements on the ASR system using single augmentation, the rank of the effectiveness is:

MODEL	S.P.	V.P.	N.I	SPEC.	EPOCHS	DATASIZE	WER
I	✔	✔	✖	✖	15	235.5	45.21%
II	✖	✖	✖	✖	15	78.5	49.00%
III	✔	✖	✖	✖	15	235.5	45.27%
IV	✖	✔	✖	✖	15	78.5	48.55%
V	✖	✖	✔	✖	15	157	46.51%
X	✖	✖	✔	✖	30	157	45.28%
SPEC15	✖	✖	✖	✔	15	78.5	49.40%
SPEC30	✖	✖	✖	✔	30	78.5	46.55%
SPEC45	✖	✖	✖	✔	45	78.5	45.39%
SPEC60	✖	✖	✖	✔	60	78.5	44.79%
SPEC90	✖	✖	✖	✔	90	78.5	44.29%
VI	✔	✖	✔	✖	15	471	43.96%
VII	✖	✔	✔	✖	15	157	46.37%
XI	✖	✖	✔	✔	15	157	47.57%
XII	✖	✖	✔	✔	30	157	44.80%
VIII	✔	✔	✔	✖	15	471	43.81%
XIII	✔	✔	✔	✔	15	471	43.67%
XIV	✔	✔	✔	✔	90	471	**42.53%**

Table 5.10: a overview table showing the WER of the models with different data augmentation applied and running with different epochs and datasize. S.P. stands for Speed Perturbation, V.P. stands for Volume Perturbation and N.I stands for Noise Injection. SPEC. stands for SpecAugment. Model I is the baseline, Model II is the plain model and model XIV is the final model.

$$\text{SpecAugment} \approx \text{Speed Perturbation} > \text{Noise Injection} > \text{Volume Perturbation}.$$

SpecAugment is ranked approximately the same as Speed Perturbation because the former has a higher potential to improve the ASR system when running with more epochs but improves less than Speed Perturbation when the epoch is small. Note the full SpecAugment should also include time-warping, which is not used in this work when solely investigating SpecAugment. Therefore, the improvements of the SpecAugment could be underestimated.

Thirdly, all the data augmentations could be combined to improve the ASR system further, although more epochs are needed when using SpecAugment. Also, Noise Injection seems to be more compatible with Speed Perturbation than SpecAugment, at least with epochs up to 30, as shown in the experiments results in models VI, XI and XII.

Lastly, the model could have about 1.54% improvement from the baseline and 5.32% of improvements from the plain model on WER using all augmentations in this work without collecting any new in-domain data. Running the model with all data augmentations further from 15 epochs to 90 epochs could further improve more than 1% on

WER. The main drawback to the data augmentations is that it could require more training time for the models. Also, the final model might still be underfitting because there is a large improvement from SPEC60 to SPEC90, and there could be further improvements from SPEC90 to a model with SpecAugment running more than 100 epochs. As model XIV has more data than needed to fit during the training, it would require more training time.

Chapter 6

Conclusions & discussions

Overall, a model with a performance of 42.53% WER is achieved on the medical testing set, as the final model in this work.

Fortunately, all the objectives in this works are achieved in the end. Recall that the two objectives are:

1. investigate the effects of various data augmentation techniques individually.
2. investigate the effects of various combinations of different data augmentation techniques.

By achieving the first objective, it is confirmed now that all data augmentations in this work could improve the ASR system's performance in different magnitude, although using a small amount of in-domain data improves much more. Hence, it is feasible to use data augmentation to boost the model's performance, especially when collecting more data is not possible.

Also, by achieving the second objective, it is now clear that different data augmentation techniques could be combined and further improves the mode's performance.

Considering the amount of in-domain data available in the emergency medical domain and achieving almost absolute 1.54% from the baseline and 5.32% improvements from the plain model on WER using only the data augmentations approach, the project's aim is roughly accomplished, although the current system is far from a good standard ASR system. The performance could be further improved about more than 1% by training the model with 75 extra epochs, and a good standard system is anticipated using similar data augmentation techniques when larger in-domain data is collected.

The main drawback of data augmentation that could be seen in this work is the increased computation cost. Comparing to training a model for two weeks and getting about 7% of improvements from the plain model that runs for 15 epochs, collecting more data is a much efficient way for improving the performance.

6.1 Limitation

One limitation of this work is the computational environment. More experiments could be investigated in time if more powerful GPUs are feasible. For example, it was not cost-effective to further investigate the SpecAugment for running more than 90 epochs as it would take too long for the current computational environments.

One other limitation is the medical dataset. Due to privacy issues, it would not be possible to collect more data, as it requires a substantial amount of negotiation between the School and the medical teams. The ASR system's performance would be expected to be a lot better than the current one because only 1.5 hours improved the previous baseline performance from 78.63% to 49.00%. Also, due to the limitation in the dataset, no validation set is used in this work. The testing set is used frequently in this work, which is not appropriate in general.

6.2 Further work

It is worth trying the SpecAugment running more than 100 epochs for further work until the model converges or sees overfitting occurs. Also, the effects of combining SpecAugment and Speed Perturbation alone could be further investigated. SpecAugment could also be further explored by investigating different policies.

Also, it might be worth asking for more medical data. This would enable us to have a better insight into the effects of using different data augmentations. So far, it is found that the improvements of using Speed Perturbation are similar to the improvements of using SpecAugment. However, [Park et al., 2020] suggested that SpecAugment is more superior to others in their works to improve performance. A better comparison could be made to validate their outcomes if the data size is no longer an issue.

Lastly, it is worth trying other data augmentations as there are many of them. Noise Injection is the traditional approach for training a noise-robust ASR system. A study Kalinli et al. [2010] proposed a Noisy Adaptive Training (NAT) algorithm, an analogue to Speaker Adaptive Training introduced in our previous work[Jin, 2020]. In principle, NAT "estimates the pseudo-clean model parameters directly without relying on point estimates of the clean speech features as an intermediate step". The paper tested NAT on both Aurora 2 and Aurora 3 dataset. Aurora 2 consists of digitally adding noise, and Aurora 3 consists of connected digit strings recorded in realistic car environments. As a result, the study shows relative improvements of 18.83% and 32.02% on Aurora 2 and Aurora 3, respectively.

Bibliography

P. Bahar, A. Zeyer, R. Schlüter, and H. Ney. On using specaugment for end-to-end speech translation. *arXiv preprint arXiv:1911.08876*, 2019.

S. N. Blomberg, F. Folke, A. K. Ersbøll, H. C. Christensen, C. Torp-Pedersen, M. R. Sayre, C. R. Counts, and F. K. Lippert. Machine learning as a supportive tool to recognize cardiac arrest in emergency calls. *Resuscitation*, 138: 322–329, 05 2019. URL https://www.resuscitationjournal.com/article/S0300-9572(18)30975-4/pdf.

J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, et al. The ami meeting corpus: A pre-announcement. In *CMI-MLMI*, pages 28–39. Springer, 2005.

T. S. o. C. S. Carnegie Mellon University. The cmu pronouncing dictionary, 2019. URL http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

C. Cieri, D. Graff, O. Kimball, D. Miller, and K. Walker. Fisher english training speech part 1 speech - linguistic data consortium, 12 2004a. URL https://catalog.ldc.upenn.edu/LDC2004S13.

C. Cieri, D. Graff, O. Kimball, D. Miller, and K. Walker. Fisher english training part 2, transcripts ldc2005t19, 04 2004b. URL https://catalog.ldc.upenn.edu/LDC2005T19.

N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2010.

J. Driedger and M. Müller. A review of time-scale modification of music signals. *Applied Sciences*, 6(2):57, 2016.

E. Eide and H. Gish. A parametric approach to vocal tract length normalization. In *ICASSP*, 1996.

EuReCaTWO. European registry of cardiac arrest - study two (eureca two), 2017. URL https://www.erc.edu/news/european-registry-of-cardiac-arrest-study-two-eureca-two.

M. Y. Faisal and S. Suyanto. Specaugment impact on automatic speaker verification system. In *2019 ISRITI*, pages 305–308, 2019.

M. Gales, S. Young, et al. The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304, 2008.

E. Gauthier, L. Besacier, and S. Voisin. Speed perturbation and vowel duration modeling for ASR in Hausa and Wolof languages. In *Interspeech 2016*, San-Francisco, United States, Sept. 2016. URL `https://hal.archives-ouvertes.fr/hal-01350057`.

M. Geng, X. Xie, S. Liu, J. Yu, S. Hu, X. Liu, and H. Meng. Investigation of data augmentation techniques for disordered speech recognition. 2020.

M. Giurgiu and A. Kabir. Comparison of vocal tract length normalization technique applied for clean and noisy speech. In *ICTSP*, pages 351–354. IEEE, 2011.

J. Goodman and J. Gao. Language model size reduction by pruning and clustering. In *ICSLP*, 2000.

Y. Grandvalet and S. Canu. Comments on" noise injection into inputs in back propagation learning". *IEEE Transactions on Systems, Man, and Cybernetics*, 25(4): 678–681, 1995.

N. Jaitly and G. E. Hinton. Vocal tract length perturbation (vtlp) improves speech recognition. In *ICML WDLASL*, volume 117, 2013.

X. Jin. Speech recognition for medical team. Master's thesis, University of Edinburgh, May 2020.

D. Jurafsky and J. H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009. ISBN 0131873210.

O. Kalinli, M. L. Seltzer, J. Droppo, and A. Acero. Noise adaptive training for robust automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(8):1889–1901, 2010.

M. Karafiát, L. Burget, P. Matějka, O. Glembek, and J. Černockỳ. ivector-based discriminative adaptation for automatic speech recognition. In *ASRU*, pages 152–157. IEEE, 2011.

T. Ko, V. Peddinti, D. Povey, and S. Khudanpur. Audio augmentation for speech recognition. In *ISCA*, 2015.

T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *ICASSP*, pages 5220–5224. IEEE, 2017.

A. Koenecke, A. Nam, E. Lake, J. Nudell, M. Quartey, Z. Mengesha, C. Toups, J. R. Rickford, D. Jurafsky, and S. Goel. Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14):7684–7689, 2020.

T. Lancet. Out-of-hospital cardiac arrest: a unique medical emergency. *The Lancet*, 391:911, 03 2018.

Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong. Chapter 1 - introduction, chapter 2 - fundamentals of speech recognition. In J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong, editors, *Robust Automatic Speech Recognition*, pages 1 – 7, 9 – 40. Academic Press, Oxford, 2016. ISBN 978-0-12-802398-3. URL http://www.sciencedirect.com/science/article/pii/B9780128023983000015.

Li Lee and R. C. Rose. Speaker normalization using efficient frequency warping procedures. In *ICASSP*, volume 1, pages 353–356 vol. 1, 1996.

A. Maas, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, and A. Y. Ng. Recurrent neural networks for noise reduction in robust asr. In *INTERSPEECH*, 2012.

K. Matsuoka. Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):436–440, 1992.

T. S. Nguyen, M. Sperber, S. Stüker, and A. Waibel. Building real-time speech recognition without cmvn. In *SPECOM*, pages 451–460. Springer, 2018.

D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

D. S. Park, Y. Zhang, C.-C. Chiu, Y. Chen, B. Li, W. Chan, Q. V. Le, and Y. Wu. Specaugment on large scale datasets. In *ICASSP*, pages 6879–6883. IEEE, 2020.

V. Peddinti, D. Povey, and S. Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *ISCA*, 2015.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. In *ASRU*. IEEE Signal Processing Society, Dec. 2011. IEEE Catalog No.: CFP11SRW-USB.

D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755, 2016.

E. Pusateri, C. Van Gysel, R. Botros, S. Badaskar, M. Hannemann, Y. Oualil, and I. Oparin. Connecting and comparing language model interpolation techniques. *arXiv preprint arXiv:1908.09738*, 2019.

D. Raj. On lattice free mmi and chain models in kaldi, 05 2019. URL https://desh2608.github.io/2019-05-21-chain/.

R. Reed, R. Marks, and S. Oh. Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. *IEEE Transactions on Neural Networks*, 6(3):529–538, 1995.

D. Seita. 1000x faster data augmentation, 06 2019. URL https://bair.berkeley.edu/blog/2019/06/07/data_aug/.

J. Sietsma. Neural net pruning-why and how. In *ICNN*, volume 1, pages 325–333, 1988.

D. Snyder, G. Chen, and D. Povey. MUSAN: A Music, Speech, and Noise Corpus, 2015. arXiv:1510.08484v1.

S. Sučík. Prosodic features in spoken language identification, 2019. URL `https://project-archive.inf.ed.ac.uk/ug4/20191566/ug4_proj.pdf`.

P. Swietojanski, A. Ghoshal, and S. Renals. Hybrid acoustic models for distant and multichannel large vocabulary speech recognition. In *ASRU*, pages 285–290. IEEE, 2013.

W. Verhelst and M. Roelands. An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech. In *ICASSP*, volume 2, pages 554–557. IEEE, 1993.

S. Viereck, T. P. Møller, A. K. Ersbøll, J. S. Bækgaard, A. Claesson, J. Hollenberg, F. Folke, and F. K. Lippert. Recognising out-of-hospital cardiac arrest during emergency calls increases bystander cardiopulmonary resuscitation and survival. *Resuscitation*, 115:141–147, 06 2017. URL `https://www.sciencedirect.com/science/article/pii/S0300957217301570`.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICPS*, pages 1096–1103, 2008.

S. Yin, C. Liu, Z. Zhang, Y. Lin, D. Wang, J. Tejedor, T. F. Zheng, and Y. Li. Noisy training for deep neural networks in speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):2, 2015.

A. B. Yoo, M. A. Jette, and M. Grondona. Slurm: Simple linux utility for resource management. In *JSSPP*, pages 44–60. Springer, 2003.

D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide. Feature learning in deep neural networks-studies on speech recognition tasks. *arXiv preprint arXiv:1301.3605*, 2013.

Y. Zhou, C. Xiong, and R. Socher. Improved regularization techniques for end-to-end speech recognition. *arXiv preprint arXiv:1712.07108*, 2017.