Speech Recognition For Medical Team

Xianming Jin

MInf Project (Part 1) Report

Master of Informatics
School of Informatics
University of Edinburgh

2021

Abstract

Speech Recognition has become popular and widely used in many fields nowadays. However, a noisy acoustic environment is still challenging in speech recognition for the medical team. Recently, some researches show that machine learning-based speak recognition system has outperformed the human dispatch for recognising cardiac arrest in emergency calls in terms of speed and sensitivity. However, those works are provided with a large number of the dataset, which would be easier to train the recognition system. In this project, we are provided with a limited amount of emergency call data ($\approx 3 hours$). We would show how to build on from a state-of-art noise-robust speech recognition system with a limited amount of data. As a result, we have achieved a state-of-art system with the performance at around 45% WER using 1.5 hours of a medical dataset for training. Unfortunately, the result of the multi-condition trained noise-robust system is not achieved due to computation issues, but we have shown the approach in details to build up such a system.

Acknowledgements

I would like to express my sincere appreciation to my supervisor Professor Steve Renals, who has tirelessly dedicated his time to the supervision of this project. This word would not have been possible without his guidance, patients, and encouragement.

I would also like to appreciate Andrea Carmantini who have kindly helped us in the coding part of the project. It would not be possible to run even a single experiment without his help at the beginning.

Then, I would like to express my special thanks to Claire Doherty and Jacob Dyer, who helped me back on track at the beginning of the project.

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 in need. I am particularly thankful to Jia Li. I would not have been able to make through several important periods without her.

Table of Contents

Chapter 1

Introduction

1.1 Motivations

Cardiac arrest is a sudden loss of heart function in a person, which causes the victim to lose consciousness and pulse to be stopped. Death could occur within a few minutes if the victim does not receive treatment such as **Cardiopulmonary resuscitation (CPR)**. **Out-of-hospital cardiac arrest (OHCA)** is the situation when cardiac arrest occurs outside of the hospital and is confirmed by the absence of systemic circulation.

In Europe, more than $350,000$ patients are affected by OHCA every year, according to the researches [11], and it is estimated that only 10% on average that patients could survive from OHCA [27]. The survival rate of OHCA patient is strongly associated with early initiating of bystander CPR. A study shows that with attempted resuscitations (e.g. CPR) to 2636 OHCA patients, 39% had signs of **Return of spontaneous circulation (ROSC)** such as breathing, coughing and movements, and 18% survived 30 days [7]. Hence, more than 50% of OHCA patients could or had a higher chance to survive with the resuscitation. This figure is significantly higher than the ones without the resuscitation. Also, there is another study shows that the recognition of OHCA in the emergency calls is positively associated with the provision of bystander CPR, ROSC and 30-day survival [57] and improves the survival rate of OHCA patients. Thus, the emergency medical dispatchers are vital in the process of recognising OHCA, giving CPR instructions to the bystanders and sending out the ambulance. However, recognising OHCA is not as easy as it appears.

A study has shown, only around 0.8% emergency calls (918 out of $108,607$ emergency calls) are OHCA cases, and the emergency medical dispatchers approximately failed 25% times to recognise OHCA cases [3]. Comparatively, the study also shows that a **machine-learning-based framework** had a significantly higher **sensitivity** (True Positive) than the human dispatchers (84.1% vs 72.5%) with slightly lower specificity (True Negative, 97.3%vs 98.8%). Most importantly, the time for the machine-learning-based framework was significantly shorter than the human dispatcher (median 44s vs 54s). The time is vital to OHCA patients as each moment without resuscitation largely decrease the chance of survival.

The machine-learning-based framework requires to process the audios (phone calls) and transforms them into textual representation without prior editing or transcription. The textual representation of the audio is then used to be analysed and predicted. The process of transforming the audio representation to the textual representation is called **Speech Recognition**. Hence, speech recognition plays an essential role in the machine-learning-based framework. The performance of the machine- learning-based framework largely depends on the quality of the textual representations, hence the performance of Speech Recognition. Thus, ***building up a well-performed speech recognition system can help the medical team to improve the survival rate of OHCA patients***. Moreover, the recognition system can also be used to identify other patients' conditions or prioritise the calls for call handlers. Overall, Speech Recognition could be useful for medical teams. However, **Speech Recognition for medical teams** is not simple. There are many challenges for the recognition [53]:

- Limited data available – the size of data can be used for training a Speech Recognition System is much smaller compared to other fields. The limitation is might due to reasons such as ethic and privacy.
- Acoustic environment – noisy and may be reverberant.
- Multi-accent,multi-lingual speakers.
- Limited language model and lexicon in the domain.
- Speaker diarization in emergency teams.

The challenges listed above makes the recognition task hard, and many of the challenges are still difficult to be tackled even nowadays. In this project, we will focus on building a noise-robust system to minimise the degradation of performance caused by the noisy acoustic environment.

1.2 Aims & Contribution

The overall aim of this project is building on from a state-of-art Speech Recognition system and adapt it for medical teams, in terms of improving the performance of the system on the medical data.

There are two objectives in this project while achieving the aim of this project. Due to the limitation of the medical dataset, our first objective is investigating the importance of in-domain dataset. We investigate the difference between training a model using more not-well-matched data and using more well-matched dataset. In this work, we found that using more not-well-matched data would degrade the performance, whereas using well-matched dataset significantly improved the performance.

The second objective is to investigate building a noise-robust system using noise training approach. In principle, the noise training should increase the noise-robustness in the system, which improves the performance. However, due to some accidents happened to the computation environment, we were not able to finish the training, although we have all other things ready for the training. The procedure of noise training is described in Chapter 5. Although we have not achieved the second objective, our final

model has a reasonable performance which is considered as having achieved the aim of the project.

Overall, in this paper, we would first introduce the background knowledge required for the reader to understand the Speech Recognition system in chapter 2. It includes an introduction to basic GMM-HMM based and DNN-HMM based Speech Recognition system. We would also introduce the possible way to improve the GMM-HMM based and DNN-HMM based system. Lastly, we would explain how to build a state-of-art noise-robust ASR system.

In chapter 3, we would discuss in details about the dataset available for the project, and tasks of this project. This chapter would also introduce some related work to the project.

In chapter 4, we would demonstrate the implementation of speech recognition by using **Kaldi**. The experiments and results evaluation of the Speech Recognition system would be shown in Chapter 5. Lastly, the conclusion, limitation and further would be made in Chapter 6.

Chapter 2

Background

Automatic speech recognition (ASR) is a process of transforming audio representation to the textual representation, without necessarily understanding the meaning of what was spoken by speakers. ASR has been studied for many years, and it is recently developed rapidly with the exponential growth of computation power and large data. Traditionally, the technology has been used for many applications such as voice dialling, interactive voice response, gaming, voice-control-based applicants, robotics [29]. Nowadays, the ASR has been advanced to more challenging tasks, such as voice searching, digital assistance and interaction with mobile devices like Siri, and even more challenging task such as ours: speech recognition for the medical team.

Noisy is any undesired disturbances acting on the intended speech signal. **noise-robust** is characteristic of a system to maintain its excellent performance under noisy conditions. Those new applications of ASR require ASR to be robust to all source of real-world noise and other acoustic distortions. However, it is still challenging to reliably recognising speech in a realistic acoustic environment. The noise-robust has become increasingly crucial for ASR to work in much more challenging acoustic environment than in the past.

In this chapter, we would first introduce the basic GMM-HMM approach for building an ASR system. Then we would introduce the DNN and TDNN acoustic model. Lastly, we would discuss how to build a noise-robust ASR system and data augmentation approach.

2.1 GMM-HMM System

Traditionally, we build an ASR system based on Gaussian Mixture Model and Hidden Markov Model (GMM-HMM). GMM-HMM refers to an acoustic model in ASR. It is called GMM-HMM as GMM is used as an emission probability in HMM. In this section, we would look at the steps of how such an ASR is built. Figure 2.1 shows high-level architecture of the ASR system. Note that the End-to-End model has different high-level architecture from the structure shown in Figure 2.1, which [26] models directly model from the input feature to the output sequence. It largely deduces the

training time and decoding time, which is desirable. However, this model requires a large amount of data, which does not fit to our task, so we would not consider the End-to-End approach.

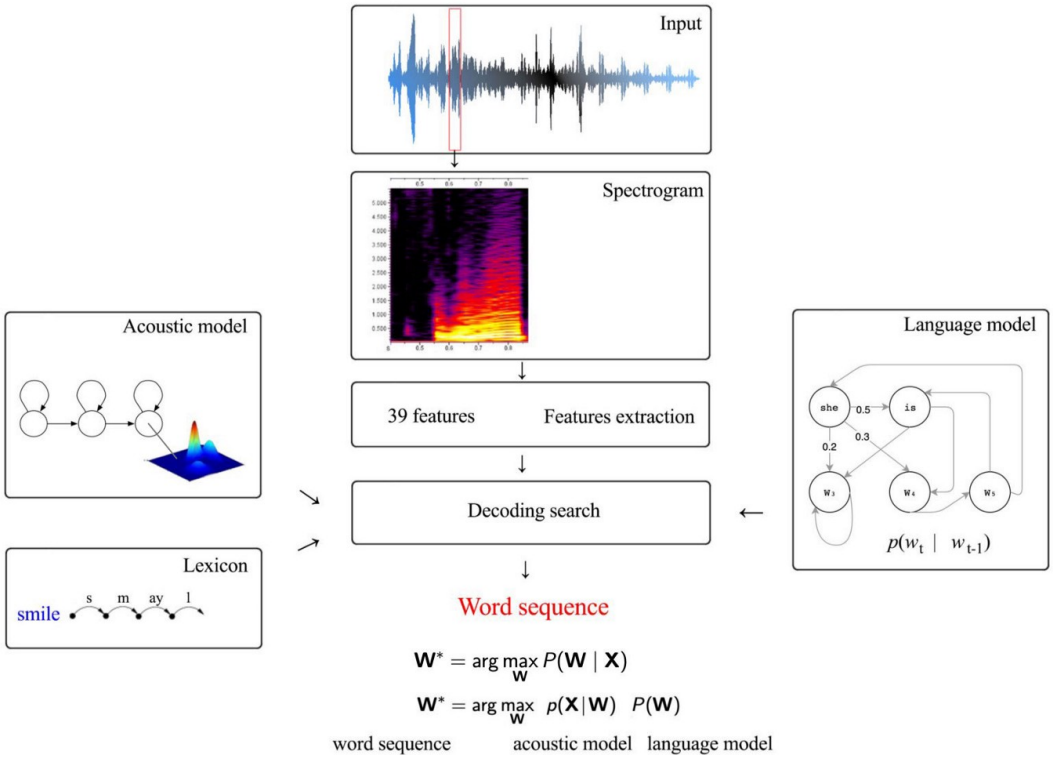

Figure 2.1: A high level overview of how ASR works [20]

ASR is a process of transforming audio singal to the text. In a more technique terms, the goal of ASR s predicting the most probable word sequence **W** given the observation **X**. However, **X** is not the input signal, but feature vector which is extracted from the input signal. In mathematical representation, it is maximising the posterior probability of word sequence **W** given feature vectore **X**:

$$\hat{W} = \underset{w \in \mathscr{L}}{\operatorname{argmax}} P_{\Lambda,\Gamma}(W|X) \tag{2.1}$$

where Λ and Γ are the acoustic model and language model parameters respectively, and $\mathscr{L}$ is the Lexicon. By using the Bayes' rule:

$$P_{\Lambda,\Gamma}(W|X) = \frac{P_{\Lambda}(X|W)P_{\Gamma}(W)}{P(X)} \tag{2.2}$$

Hence, equation 2.1 could be reformed as:

$$\hat{W} = \underset{w \in \mathscr{L}}{\operatorname{argmax}} P_{\Lambda}(X|W)P_{\Gamma}(W) \tag{2.3}$$

where $P_{\Lambda}(X|W)$ is the acoustic model likelihood and $P_{\Gamma}(W)$ is the language model likelihood.

The acoustic model is about modelling a sequence of feature vectors given a sequence of phones instead of words. The language model expresses how likely a given string of word sequence is.

In this section, we would explain how to extract features from the input signal, how to use the HMM model introduced as an acoustic model, and define the language model and lexicon.

2.1.1 Feature extraction

We extract features from the signal and create and dense representation of the content, so we could learn the important information from it without affecting much by the noise. Feature extraction is used in the acoustic model as it creates the feature vector **X**. The figure 2.2 shows a pipeline how feature (MFCC) is created.

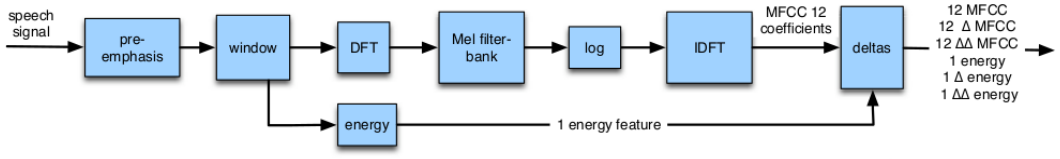

Figure 2.2: Extracting a sequence of 39-dimensional MFCC feature vectors from a quantized digitized waveform [20]

There are many acoustical features types such as MFCC, FBANK and PLP. The acoustic feature we use is Mel-frequency cepstral coefficient (MFCC), which is the most commonly used in ASR, as it aims to characterise the shape of the vocal tract, which corresponding to the spectral characteristic of sound produced.

We firstly use filters to preemphasis the signal to boost the high energy frequency, making the high formants more available so that the acoustic model can detect phone better. We do short-term analysis by Windowing that assuming the signal is not changing in a short period. We use the Discrete Fourier Transform (DFT) to the window to get a power spectrum. Then we apply the Mel Filter bank to get the filterbank features that is the sum of energy in each filter. The Mel filters are filters equally spaced on the Mel scale, which relates the human perceived frequency to measured frequency, allowing features to have a better match to human hearing. We take a logarithm of the filterbank since the human response to signal level is the logarithm. We compute the cepstrum using inverse (DFT) to separate the filter and the source, resulting in 12 cepstral coefficients (delta) where each coefficient represents the information of the vocal tract filter solely. We add feature, energy, which is the total energy of a frame. Energy correlates with phone identity, which is useful for phone detection. Hence we have 13 features for a frame. As the signal changes from frame to frame, for each of 13 features, we also need velocity and acceleration feature. Therefore, in total, we could have 39 features.

2.1.2 Acoustic Model and lexicon

Lexicon is simply a list of words, with pronunciation for each word expressed in a phone sequence. We use Markov Chain to model all the possible sequence of phones.

Recall from equation 2.3, $P_\Lambda(X|W)$, the likelihood of the sequence of feature vector given a sequence of phones, could be approximated using acoustic model and lexicon. HMM could be used as an acoustic model shown in figure 2.3. The top nodes are the HMM states. Traditionally, we use three HMM states to model a phone, enforcing the minimum duration of the phone. Given a trained HMM, we align the observations with HMM states by finding the most likely internal state path, shown in figure 2.4:

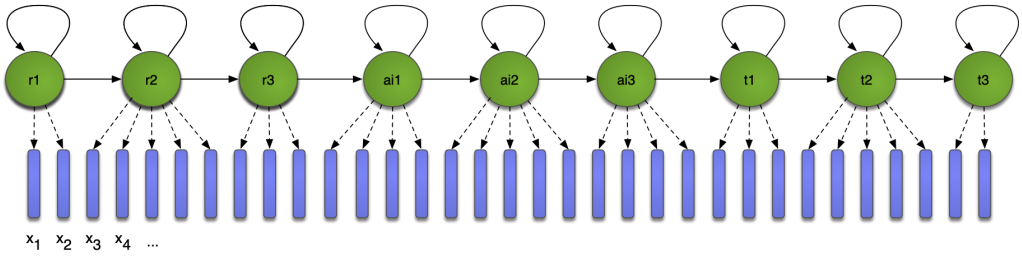

Figure 2.3: HMM used as acoustic model for ASR [19]

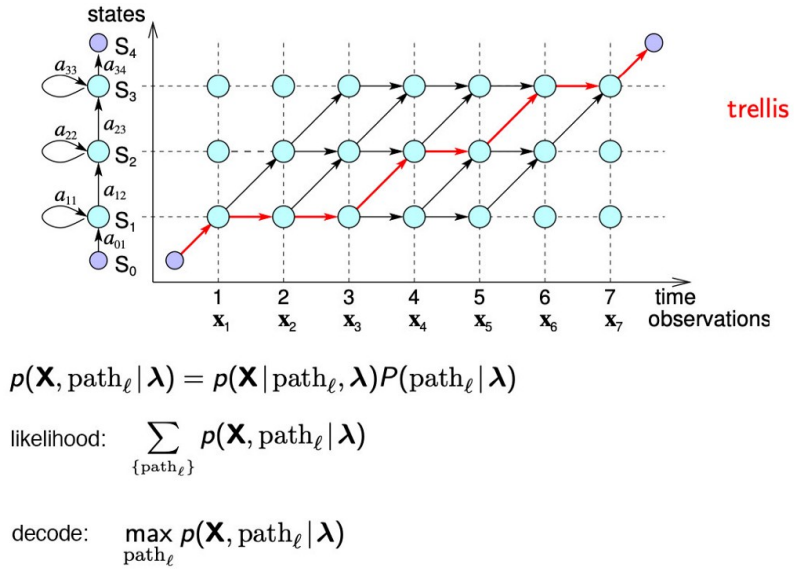

Figure 2.4: Find the best path of internal state sequence, given trained HMM [19]

The above examples show the acoustic model of monophone, assuming each phone is independent of others. However, in real life, sound changes according to the context surrounding the words. It is because of the articulation decadents on the surrounding phone. Hence, we need a more sophisticated acoustic model. In our experiment, we would use **triphone** model to model the dependency between the phones. The structure of the triphone model is similar to the monophone model, but the HMM state differs, as shown in figure 2.5. It results in much more HMM states and much more parameters

required for the acoustic model. We use smoothing or parameter sharing methods to decrease the number of parameters [54] efficiently. In Kaldi, we use Decision Tree to cluster the states to deduce the number of parameters [41].

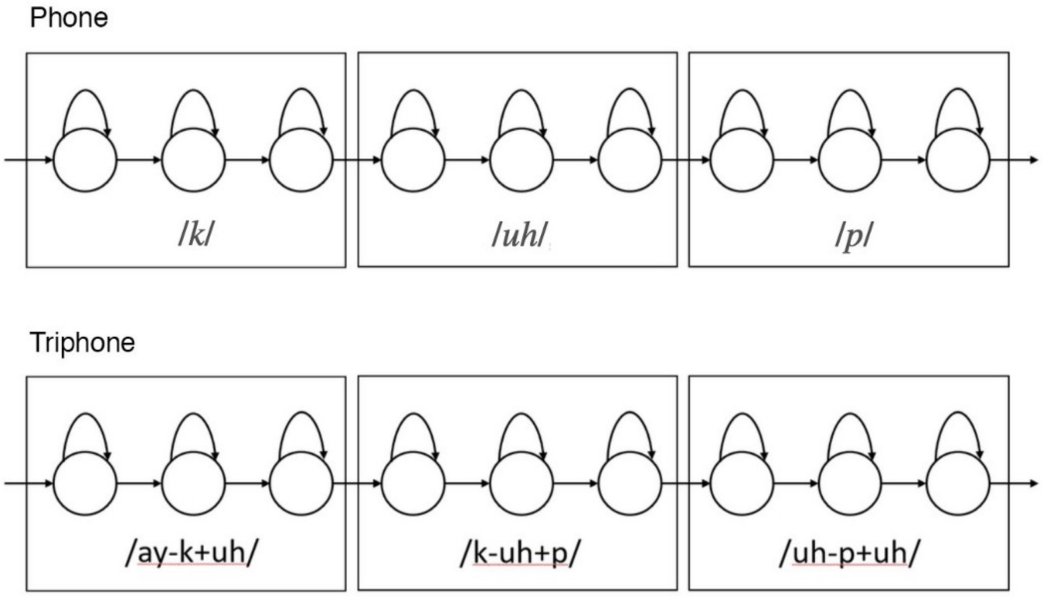

Figure 2.5: A figure shows HMM state of monophone and triphone model [19]

2.1.3 Language Model

A language model calculates the likelihood of sequence of words $P(w_1, w_2, .., w_n)$. It is included in the decoding to improve the accuracy as we assume the audio is grammatically and semantically correct. N-gram language model meaning the current word depends on the N-1 previous words, modelling the dependency between the words. For example, unigram language means that the current word is independent of others, and the bigram model means the current word is dependent only on the previous word. The likelihood calculated by the language model is also called prior likelihood as it simply counts the appearance of words sequence in the corpus:

$$P(w_i|w_{i-n+1}, ..., w_{i-1} = \frac{C(w_{i-n+1}, ..., w_i)}{C(w_{i-n+1}, ..., w_{i-1})} \tag{2.4}$$

For large N, the corpus may not contain the word sequence, and we need to apply the smoothing method [33] or backoff model to allocate the probability to unseen words sequence [22].

2.1.4 Decoder

Finally, given the acoustic model and the language model, the decoder searches the most likely word sequence given the observation X. Find best sequence could be done using Viterbi Algorithm [20]. Kaldi solves the searching by using word lattices.

In Kaldi, Weight Finite State Transducers (WFSTs) [38] is used for training and decoding algorithm for building the GMM-HMM system. The decoding is performed by using HCLG graph:

$$HCLG = H \circ C \circ L \circ G$$

where WFST of individual H,C,L,G are defined as:

	TRANSDUCER	INPUT SEQUENCE	OUTPUT SEQUENCE
G	WORD-LEVEL GRAMMAR	WORDS	WORDS
L	PRONOUNCIATION LEXICON	PHONES	WORDS
C	CONTEXT-DEPENDENCY	CD PHONES	PHONES
H	HMM	HMM STATES	CD PHONES

Table 2.1: a table showing different component represent as WFSTs, with their input sequence and output sequence. By combing transducers HCLG, it results in a transducer that maps from HMM states to word sequence

Finally, the performance of the decoding is measure by using Word Error Rate (WER) [20]:

$$WER = \frac{S+D+I}{N} * 100\% \tag{2.5}$$

where S,D and I stands for substitutions,deletions and insertions respectively, and N is the number of words in reference transcription.

2.2 Improves the GMM-HMM system using Speaker adaptation

For pattern classification, if the training data is not representative of the test data, the mismatch between the datasets would degrade the performance of the classification. Hence, for ASR, if the training set does not represent the new speakers in the test set, the accuracy could be largely decreased. The solution to it is **adaptation**. Adaptation allows adapting the acoustic model to be more closely match the new speakers by using a small amount of data from the target speaker. [13]. In short, we want our speaker-independent system to have the performance close to the speaker-dependent system by using the speaker adaptation system. Although this might not be quite related to noise-robust, it is important to know as fMLLT adapted feature and alignment of data from SAT would be feed into the DNN-HMM model for training. The reason why using an alignment of data from SAT model is that it has more accurate alignment according to some researches, though there are many researching on GMM-free DNN training.

2.2.1 LDA and MLLT feature transform

One approach to adapt the system is to do a linear transform of parameters in the acoustic model. It is shown to be effective for the case when the adaptive data is limited. Maximum likelihood linear regression (MLLR) is one of linear transform approach that map an existing model set into a new adapted model sets that the likelihood of the

adaptation data is maximised. There are two main types of MLLR: unconstrained and constrained MLLR. The equation of unconstrained MLLR shown below []:

$$\hat{(um)} = A^{(s)}\mu^{(m)} + b^{(s)}; \quad \hat{\Sigma(sm)} = H^{(s)}\Sigma^{(m}H^{(s)T} \tag{2.6}$$

where s indicates the speaker. W=[bA] and H are the transformation matrix applied to the mean and covariance, respectively. If Σ is diagonal:

$$\mathcal{N}(y;\hat{(um)},\hat{\Sigma(sm)}) = \frac{1}{|H^{(s)}|}\mathcal{N}(H^{(s)-1}y;H^{(s)-1}A^{(s)}\mu^{(m)} + H^{(s)-1}b^{(s)},\Sigma^{(m)}) \tag{2.7}$$

The term unconstrained meaning there are no constraints between the transformation applied to the mean and covariances, while the constrained MLLR meaning the two transformations are constrained to be the same. This results in a linear transformation in feature-space, so constrained MLLR is also called fMLLR. fMLLR is often used in speaker adaptive training (see in 2.2.2).

Linear Discriminant Analysis (LDA) is a machine learning technique using to reduce the dimension of the feature space. The formula for LDA could be found in [12]. It is usually combined with MLLR to adapt to the system. Detail of using LDA and MLLR could be found in [41] .

According to some researches, LDA and MLLT feature transform could decrease the WER (explain later) HMM-based system up to 25% in some of the test data [49].

2.2.2 Speaker Adaptive Training

Another approach for speaker adaptation is by speaker-adaptive training (SAT), shown in figure 2.6. SAT is often used to avoid 'waste' a large number of parameters encoding the variability between speakers rather than the variability between spoken words which is the true aim [13]. In SAT, a transform is estimated for each individual training speaker, and then the canonical model is estimated given all the transforms. We train it on the fMLLR adapted features.

2.3 Hybrid DNN-HMM system

As our computation power developed, deep learning techniques and deep neural networks [34] are widely used in many fields, including ASR. Hybrid DNN-HMM model is a state-of-art ASR model which outperformed the GMM-HMM model [17]. It is called DNN-HMM as it uses Deep Neural Network (DNN) to replace the GMM for calculating the emission probability in the HMM model (acoustic model). Figure 2.7 shows an architecture of the hybrid DNN-HMM system. In short, DNN is used as an acoustic model, which takes input feature, maps to a vector of likelihoods of the feature in each hidden state in HMM. DNN refers to the neural network with many layers (deep). The multi-layers of DNN generate arbitrary boundary between different classes (states) that helps DNN to learn the patterns in the input features.

For a DNN with N layer, the output of the n_{th} layer can be written as:

$$o^{n+1} = \sigma(z(o^n)), 0 \leq n < N \tag{2.8}$$

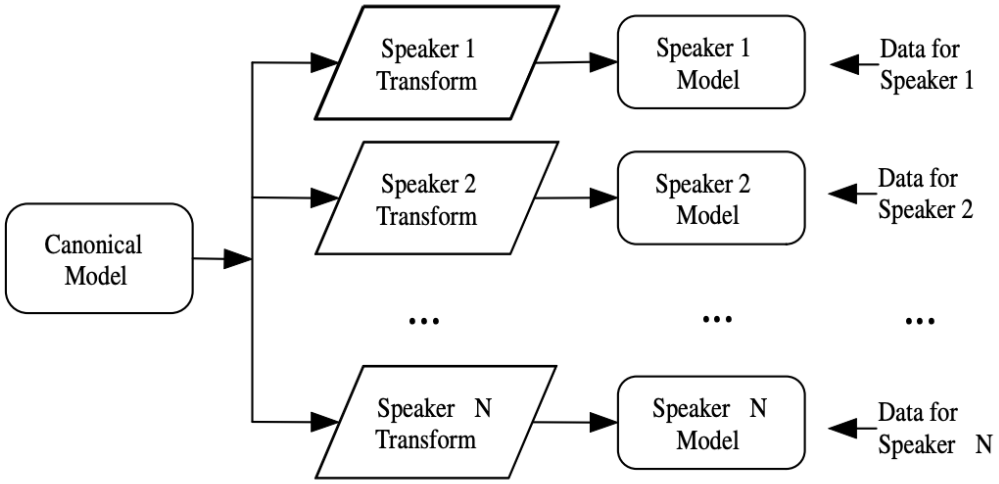

Figure 2.6: A figure shows architecture of speaker adaptive training [13]

$$z(o^n) = W^n o^n + b^n \tag{2.9}$$

where σ is a non-linear activation function, which transforms input non linearly, W^n and b^n are the weight matrix and bias in the layer n. The Weight matrix and bias could be trained using forward and backward computation[34]. The dimensions of W and b are MxN, Nx1 respectively, M is the dimension of the output, e.g. the number of hidden states, and N is the dimension of input, e.g. MFCC, fMMLR features.

2.3.1 TDNN and other DNNs

There are many different architectures for neural networks nowadays. The traditional DNN architecture, shown in figure 2.7, is a typical feed-forward neural network. The information is passed from a layer to the next layer without going backwards, and the links between nodes do not form a cycle. It was the first and most straightforward type of neural network, but it is not used in practice nowadays in ASR field as it has many drawbacks. For example, it cannot model the context information well as it is memoryless (the information in previous input is not passed to the next input).

A Recurrent Neural Network addresses the memoryless problem. In RNN, the connection between the nodes forms a directed graph along the temporal sequence, shown in figure 2.8. Thus, it allows each layer in the model also dependents on the past event. This makes it applicable for tasks such as ASR. However, RNN has problems like gradient vanishing, where information get rapidly lost.

Long Short Term Memory (LSTM) is a type of RNN which address the gradient vanishing problems, and it is capable of learning long-term dependencies which make RNN learn the patterns across time. The figure 2.9 show the difference between standard RNN and LSTM. In short, common LSTM consists four units: cell, input gate, output gate and forget gate, where the cell is used for remembering values over an arbitrary time interval, and the gates are used for manage the information flow in the

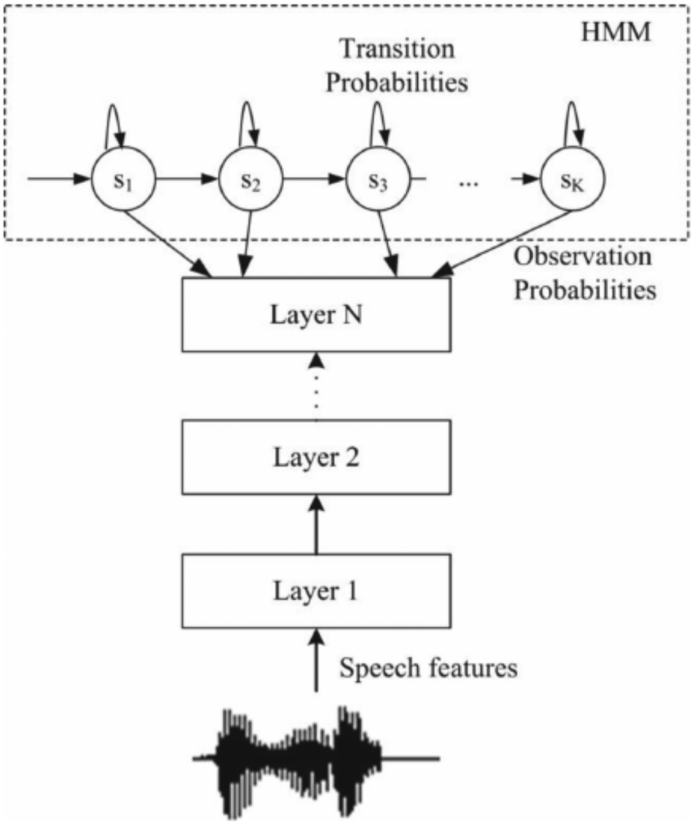

Figure 2.7: A figure shows architecture of the hybrid DNN-HMM system [23]

cells.

Also, Convolutional neural network (CNN) is applicable for ASR. The details of architecture for CNN is found in [34]. In general, CNN has advantages of shift-invariant or space invariant. CNN is mainly used in End-to-End ASR, which is not applicable to our case.

Time delay neural network (TDNN) is another popular neural network used in ASR. It is because TDNN classifies the pattern with shift-invariant, meaning it avoids the need to determine the beginning and endpoints of a sound to classify it. Also, TDNN models the context information for the speech signal, meaning the long-range context information of the speech signal is utilise. Figure 2.10 show an example of the TDNN structure (with sub-sampling)[37].

Both TDNN and DNN are feed-forward neural networks. The difference between TDNN and the standard DNN is the way they process a wider temporal context. Standard DNN learns an affine transform for the entire temporal context. Whereas in TDNN architecture, the initial transforms are learnt on narrow contexts, and deeper layers process the hidden activations from a wider temporal context. Therefore, the higher layers can learn wider temporal relationships. Each layer in a TDNN operates at a different resolution, which increases as the layer becomes deeper in the network[37].

The transformations in the TDNN are tied across time steps. During back-propagation,

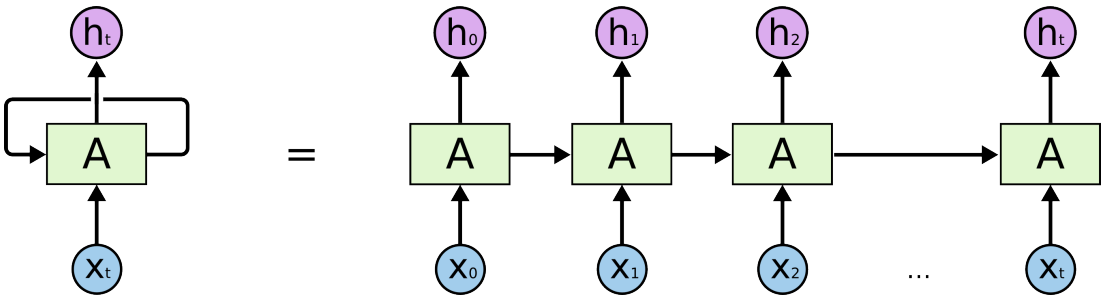

Figure 2.8: A figure shows an example architecture of RNN [36]

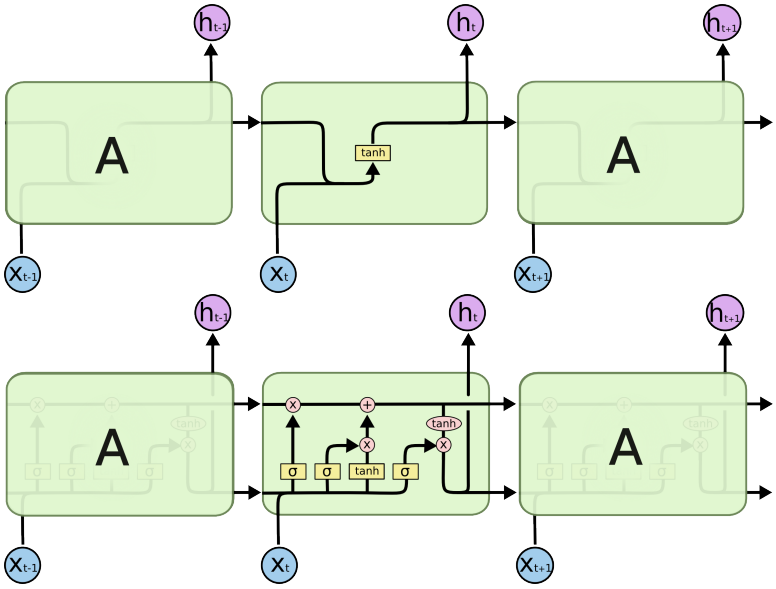

Figure 2.9: A figure shows the difference between standard RNN (top) and LSTM (bottom) [36]

due to tying, the lower layers of the network are updated by a gradient accumulated over all the time steps of the input temporal context. Hence, it forces the lower layers of the TDNN to learn shift-invariant feature transforms.

Both TDNN and LSTM are used in the state-of-art ASR system. TDNN has advantages of easier for training compares to LSTM. However, LSTM has shown to have higher performance on SwitchBoard dataset [14]. We are using TDNN in this project because it is much easier for us to build TDNN and TDNN system is state-of-art (through the existing recipe in Kaldi).

2.3.2 LF-MMI Model

Lattice Free-Maximum Mutual Information (LF-MMI) Model, also known as chain model, is a type of DNN-HMM model. Previous DNN-HMM model use frame-level cross-entropy [20] to train the DNN. Due to the nature of dependencies between frames, it is better to have utterance-level critical rather than frame-level criteria. LF-MMI model uses Maximum Mutual Information (MMI) as sequence discriminative training criteria. The term ”sequence” means taking into account the utterance as

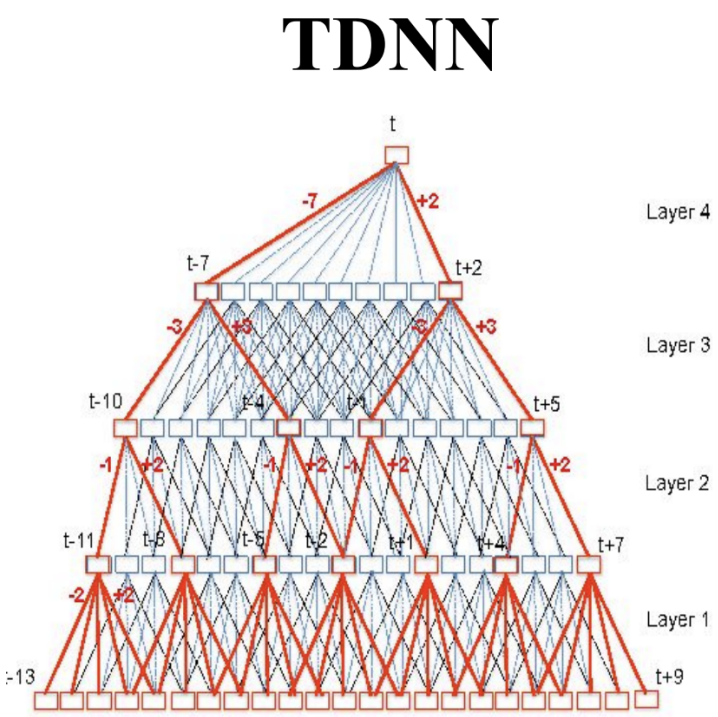

Figure 2.10: A figure shows an example architecture of TDNN neural network[37]

whole instead of frame-level before, and term ”discriminative” means minimising objective function using the gradient-based method to optimise the criteria of the task. The MMI defines as [43]:

$$F_{MMI}(\theta) = \sum_{r=1}^{R} log \frac{P_\Lambda(O_r|M_{w_r})P_\Gamma(w_r)}{\sum_{\hat{w}} P_\Lambda(O_r|M_{\hat{w}})P_\Gamma(\hat{w})} \tag{2.10}$$

where M_w is the HMM corresponding to the word w. More details of MMI is in [39].

Recall from section 2.1.4, GMM-HMM system solves the decoding by using word lattices. LF-MMI model is called lattice-free because it does not need to compute the lattice. Instead, it requires GPU to fit a denominator graph [43] to compute the MMI. In order to fit the graph, we need firstly use a phone level language model instead of the word-level language model.

The advantage of Chain Model is that the performance of Chain model is slightly better (about 5% relatively better) then those of conventional DNN-HMM systems and the system is about three times faster to decode [42]. This is an important property relate to our task as we need to decode the phone calls as fast as possible.

2.4 State-of-art noise-robust ASR

The noise conditions in ASR is mainly described by "signal to noise ratio" (SNR). The equation for SNR shown below[2]:

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{2.11}$$

where P stands for the average power, which is the area of the absolute squares of the time-domain samples divided by the signal length. Due to the fact that many signals have a very wide dynamic range, we often express the power using the log decibel scale:

$$P_{dB} = 10\log_{10}(P) \tag{2.12}$$

$$SNR_{dB} = P_{signal,dB} - P_{noise,dB} \tag{2.13}$$

The higher value of SNR means less noise affected the signal. With higher values of noise, the variance in the input feature would increase, and result in a worse performance of ASR.

2.4.1 CMVN

A large number of researches has been carried out for building noise-robust ASR in the past 30 years. Cepstral Mean and Variation Normalisation (CMVN) has been shown as an efficient technique for state-of-art noise-robust ASR [58]. CMVN could be computed per utterance or speaker (Kaldi uses per utterance option) on the extracted features by subtracting the mean followed by division on the variance of the cepstral feature. We then add the energy feature to the normalised feature extracted, followed by velocity and acceleration feature.

The robustness of CMVN could be explained by its capability of decreasing the difference between the training and test set caused by the noise [1, 45]. Furthermore, it is shown in [45] that CMVN could also decrease the difference between the feature representation and the background noise.

2.4.2 DNN based acoustic model

In ASR, GMMs are widely used to characterise the distribution of speech in the cepstral domain. It is important to know that the effect of noise is addictive in the spectral domain when the underlying model is Gaussian. The equation below simulates the noise speech in the cepstral domain, assuming zero distortion on channel [28]:

$$y = x + log(1 + exp(n - x)) \tag{2.14}$$

where y is the noise speech, x is the clean speech and n is the noise.

Many studies [28] show that with an increase in noise means value (decreasing in SNR), the variance in noise speech increase and results in much worse performance in ASR. Therefore, using GMM to model noise speech might not be a good solution.

Alternatively, it is shown by [48] that ASR using DNN based acoustic model could easily match the state-of-art performance of GMM based acoustic model without any explicit noise compensation. It is because the non-linear layer in DNN is shown to be more robust to variation in input feature [28].

2.4.3 Multi-condition training

A natural way to deal with the noise in the acoustic environment is using multi-condition training. It trains the acoustic model with all noisy speech and hopes one of the trained noise would appear in the testing set. It might sound silly, but it has been shown [18] to be an effective method to largely improve the performance of the ASR when dealing the noisy speech. There are two disadvantages of multi-condition training: firstly, it is tough to enumerate all the possible noise types and SNRs that testing environment would have; secondly, the model (GMM) trained with multi-condition data would have wide distribution since it needs to model all the acoustic environment.

Fortunately, those drawbacks could be easily solved using the DNN acoustic model, shown in [48]. DNN model naturally has more power for noise pattern learning as DNN model focus on the high complex phone/state boundary. It can address more grinding noises and deal with heterogeneous noise patterns, meaning different noise of different magnitude level could be learned simultaneously [60].

2.4.4 Supervector, UBM and iVector

Supervector is high dimensional vector that concatenates the mean vectors of the GMM for the speaker. The supervector could be represented as:

$$m_u = n_0 + Tw_u \tag{2.15}$$

where m_u and m_0 are the supervector [8] for the utterance u and the Universal Background Model (UBM). w_u is the iVector (identity vector) for utterance u and T is the total variability matrix, which is combined from the speaker subspace and channel subspace. The total variability matrix T could be estimated by using EM algorithm [9] and iVector could be estimated by mean of the posterior distribution of w_u given utterance u and T.

A universal Background Model (UBM) is a GMM trained from a large amount of data using the maximum likelihood criterion [16]. It is traditionally used as a speaker verification system to represent the speaker-independent feature characteristics, which is compared against a model of speaker-specific features characteristics.

An iVector is a low-dimensional fixed-length vector which is extracted from the feature vectors. It contains rich information about the speaker and acoustic environment of the corresponding segment. Hence, it helps the ASR system to be both speaker adapted and noise-robust. It is usually used as an additional input along with the acoustic feature to the DNN acoustic models. iVector is initially used for speaker verification [8], but now it is widely used in many areas in speech processing tasks [21, 52], including building a state-of-art noise-robust system.

2.4.5 High-Resolution MFCC and PCA

The MFCC feature used in the DNN model is high-resolution MFCC, which is not same as the MFCC in GMM-HMM system. It is extracted in a similar way to the MFCC but without cepstrum truncation. The intention of it is to provide raw speech features of spectrogram to the DNN, since DNN has ability to learn features from the input. However, it is not fed into the DNN directly. Instead, it is used to get the fMLLR adaptive feature, which is feed into the DNN. Also high-resolution MFCC is used for extracting iVectors.

Principal component analysis (PCA) technique is usually applied to high-resolution MFCC before the iVectors extraction. PCA is used to emphasize variation in the feature space and bring out strong patterns in a dataset. It is often used in machine learning to make data easier to learn.

2.5 Data augmentation

Collecting data is undesirable as it is usually expensive. Data augmentation is a strategy that allows us to modify the original data, which significantly increase the diversity of the training data without actually collecting data. Data augmentation techniques are now commonly used in neural network training [47].

As our medical data is limited, we think data augmentation is a good approach to increase the diversity of training dataset and improve the performance.

2.5.1 Speed and Volume perturbation

Speed and Volume perturbation [24] is a data augmentation technique now commonly used in speech recognition. Naturally, people could speak the same speech with different speed and volume, and we do not want the speed to affect our recognition. By augmenting the speed and volume of the audio, we generate more devise data allowing our model to learn the speed and volume patterns.

Sox [1] is generally used as the function for modifying speed and volume of the signal. As shown in [24], they achieved a relative improvement of 4.8% in their task with speed perturbation. The speed factors used were 0.9, 1.0 and 1.1, so it results in the triple size of the training data. For volume perturbation, each audio is scaled with a random value drawn from a uniform distribution [0.125, 2];

2.5.2 Noise injection

It has been known for three decades that imposing noises to the input signal can improve the generalization of neural networks [50]. ”Noise injection” is another type of data augmentation techniques, which is mixing the original training data with some sampled noise from real life.

[1] http://sox.sourceforge.net/

Sox is also generally used to mix the original signal with the noise. Noise is usually injected with a ratio proportional to the original signal. SNR is used to indicate the value of the noise injected. Lower SNR value means more noise injected.

As the study shows, adding a small magnitude of noise in the input behaves similarly as introducing some regularization techniques in the objective function [31]. With noise injection, the training prefers an optimal solution at which the objective function is less sensitive to the noise [15]. There is 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 [44].

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.

Chapter 3

Dataset, Tasks and Related Work

3.1 Dataset & Tasks

In this project, we would use three datasets for building the ASR system. The first dataset is the medical dataset provided by the NHS, consisting 50 recordings (8kHz) of approximately 3 hours of emergency calls and the corresponding transcriptions. The speakers are medical dispatcher and the bystander of patients. The language spoken is Scottish English. Without violating the privacy issues, we have not scrutinised into the transcription or listened to the recording, but we deduce that the acoustic environment is noisy by some emergency calls example shown on youtube (real 999 Emergency Calls). As this dataset is limited, we cannot directly train an ASR system using the dataset. However, we would use half of them for further training to investigate the importance of "in domain" dataset. Hence, we split the dataset evenly into a training set and a testing set with 25 recording each and a length of $\approx$ 1.5 hour each. Note that exact time of each recording is unknown, but they are approximately equal as they have same number of recording, although the length of each recording is not same. The splitting process is random which ensures the both dataset are unbiased, using a Kaldi script[1] and set the parameter cv_spk_percent to 50. Also, we assume that there are only two speakers in one recordings, and in total 100 speakers, although same medical dispatcher could appear in multiple recordings. There are and 19725 words in the medical testing set, and this figure would be uesd when calculating the WER.

The second dataset we would use is AMI Meeting corpus [4]. The AMI Meeting Corpus is a multi-modal data set consisting of approximately 100 hours of meeting recordings (16 kHz). The language spoken is English, but the speakers are mostly non-native speakers. It is the primary dataset we would use 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. 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

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

recordings of $\approx$ 11.5 testing set. SDM is split into 135 recordings of $\approx$ 77 hours of the training set, 18 recordings of $\approx$ 11.5 hours of the validation set, and 16 recordings of $\approx$ 11.5 hours testing set. Also, note that the number of the speaker is unknown in a conversation, but we use a 'brain-dead' assumption that assumes every 30 seconds in IHM dataset is a new speaker, so 10492 speakers in IHM training dataset, 1197 speakers in the validation set and 1166 speakers in the testing set. Similarly, we use 30 seconds assumption in SDM1 dataset, and there are 10198 speakers in the SDM1 training dataset, 1171 speakers in the validation set and 1147 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 provided, and it is a large corpus that is suitable for training DNN. Most importantly, it has existing Kaldi recipe that builds a state-of-art ASR system using AMI corpus.

The last dataset we will is MUSAN dataset. Musan is originally 11 GB corpus, consisting of music, speech and noise. Due to the computation environment provided, we only downloaded the noise part of MUSAN, which is approximately 1GB size of noise data and consists of 930 different type of 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 sampled from real life. It is used as the noise injected into the original data (Emergency medical data and AMI data).

The task of this project is building on from a state-of-art ASR system and adapt it to the medical team. We would focus on building a noise-robust system to minimise the degradation in performance due to the noisy acoustic environment and thereby improves the performance. We would firstly build a baseline ASR system using the AMI corpus. The performance of the baseline model is evaluated on the testing set of emergency medical data. Then, we would investigate the importance of using more data: will the performance be improved simply using more data or more data that matches the testing set. Later, we would investigate the effect of noise training by injecting noise into original data.

The evaluation metric we will use is standard WER explained in section 2.1.4:

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

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. In our experiment, N = 19725.

3.2 Related Work

As we have mentioned section 1, there has been studying [3] showing that 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 motivation of our project, but their architecture of the model would not be followed as the mag-

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

nitude of available medical data is very different, although they have not mentioned details of their ASR model at all in their report.

The aim of our project is building a noise-robust ASR system. Building a noise-robust system is a popular topic that has been studied over many years. Noise training (multi-condition training) in DNN is our approach to building a noise-robust system. It is a popular and effective approach shown in many works [60]. The advantage of it has been discussed in section 2.4.3.

Another related work is the DAE approach [59, 30], which is trying to recover the original clean signal. Both of noisy training and DAE corrupt the DNN input by randomly sampled noise. This approach is not chosen as we think the noise training approach is simpler to implement and shown to be effective.

There is another multi-condition training [62] related to our approach, in the sense that both approaches are training the DNN with speech signal in multi-condition. The difference is that we obtain the multi-condition speech by injecting noisy into clean speech, whereas they obtain the multi-condition speech from real life. Their approach does not fit our situation as available data is limited in this project.

Also, there is another related work which is a study of racial disparities in ASR [25]. As the result of the study, the author found that even state-of-the-art ASR systems that developed by big companies such as Apple, Google and Amazon have racial disparities in ASR, with WER of 0.35 for black speakers compared with 0.19 for white speakers. The author has proposed a strategy of using the more devised training dataset, consisting of African American Vernacular English to address the problem. It inspired us for using a large English dataset of a mixture of the speaker to minimise the racial disparities, as English is spoken by many different racial.

Chapter 4

ASR Implementation using Kaldi and Slurm

4.1 Kaldi

Kaldi [1] is an open-source ASR toolkit firstly proposed by in 2001 [40], and it is used for teaching and researching. It has a forum which allows all users to propose their questions or bugs (rarely) they found in the code. The author and other experts are responsible for answering questions and maintenance. Kaldi is active, as it reflects the state-of-art researches. Also, the implementation of some of the state-of-art ASR system is building on the top of Kaldi. Hence Kaldi also contains the scripts for building a state-of-art ASR system. Readers should understand three components in Kaldi for building the ASR system [55]:

- **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. To access them from anywhere, we set (inside the file path.sh) an environment variable KALDI ROOT to point to the Kaldi installation and add this to the system path.
- **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 is corresponding to a version of a particular study or experiment carried in the past. It is contained in the **egs** folder in Kaldi directory, and it always contains a script **run.sh** which consisting sequence of scripts needed to run through for building a corresponding ASR system of the study in one go.

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

In this work, as we have mentioned earlier, we are building from the state-of-art ASR system and adapt it to the medical team. Therefore, the entire implementation of the ASR system is based on a single recipe of **ami (version s5b)**[2] with modifications to some of the scripts.

The AMI recipe reproduces the model in [56] that uses CMUDict [35]. CMUDict is an open-source pronouncing dictionary of English, 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 language model used in AMI recipe is built from fisher transcripts [6, 5]. Version s5b simply means a simplified streamline of original one [3].

4.2 Baseline model

The baseline model for our ASR is built on top of the AMI recipe. We are training two models which are based on different training dataset: individual headset microphone and single distant microphone dataset. The main steps of building each model are followed as:

1. Prepare the training dataset, language model(Fisher Corpus), and lexicon (CMU-Dict) using run_prepared_shared scripts[4]); Note the language model and lexicon are chosen mainly due to the convenience as they are already in use with AMI recepe. The original AMI recipe has achieved excellent results with the Fished Corpus and CMUDict.

2. Split the training dataset into two training dataset where one contains 98% (about 75.5 hours), and another contains 2% (about 1.5 hours). It is done for comparison later for investigating the importance of more data. It is done by using an existing script (/utils/subset_dir[5]) to split the data, by setting the parameter "num-utt" to 106000. The script randomly seleted 106000 sentences from the original training dataset (IHM training set contains 108221 sentences and SDM1 training set contains 107047 sentences). The remaining sentence would be used as extra AMI training data.

3. Extract MFCC features from AMI training dataset and applies the Cepstral Mean Variance Normalisation techniques to MFCCs. Note that signal in the training dataset is 16 kHz, but the signal in our medical dataset is 8KHz. Therefore, we need to modify the configuration when making MFCC features [6] to allow the downsampling training dataset to 8KHz in order to make the model works on the 8KHz medical data. Also, another approach would be using a script [7] that uses

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

[5] https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/utils/subset_data_dir.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/

sox (/utils/data/resample_data_dir.sh) to resample the original signal at 8 kHz;

4. Build a monophone model without delta feature and align the training dataset. Note that training dataset is aligned each time after a new model is built;
5. Build a triphone model with delta feature;
6. Build a triphone model with delta and delta-delta; features;
7. Build a triphone model with LDA and MLLT;
8. LDA+MLLT+SAT;
9. Clean the training dataset by removing the bad alignments;
10. Augment the training dataset by using speed perturbation techniques. We produce three versions of each audio with speed factors 0.9, 1.0, and 1.1. 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.
11. Apply the PCA to high-resolution MFCC; build UBM and extract iVectors;
12. Train the final DNN-HMM model. The DNN-HMM model is using the fMML-adapted feature and iVectors extracted from high-resolution MFCC of the training data for the training process; the decision tree and alignment are obtained from the SAT GMM system. The type of DNN is TDNN. The type of DNN-HMM model is Chain Model. Phone level Language Model (LM) is used instead of word level, and no LM backoff as backoff introduces many states which is undesirable. The objective function used in training is MMI. In mathematical representation, we find an acoustic model Λ which maximise the MMI:

$$\underset{\Lambda}{\text{argmax}} \sum_{r=1}^{R} log \frac{P_\Lambda(O_r|M_{w_r})P_\Gamma(w_r)}{\sum_{\hat{w}} P_\Lambda(O_r|M_{\hat{w}})P_\Gamma(\hat{w})} \tag{4.1}$$

The configuration of the hyperparameters such as L2 regularisation parameters (avoid overfitting) in the neural network is kept same[8].

13. Decode the medical validation set by Neural Network based online decoding with iVectors. The unnormalised high-resolution MFCC and iVectors used as input to the neural network. The idea is that iVectors provides information to the neural network about speaker's property, which is proved to be helpful for decoding [41]. Note that the scoring process in origin script requires 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 performance of the two models are compared, and the better model are chosen to be our baseline model in further experiments.

---

resample_data_dir.sh

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

Also, further experiments follow similar procedures as the baseline model. Further investigations are all based on modifying the training dataset. Hence, the architecture of the ASR system is the same across different experiments.

4.3 Computation environment

All the scripts are mainly run on the MLP GPU Cluster of the School of Informatics [9], which is made of multiple GPU boxes (nodes) where each box consists 8 x 1060 Ti GTX GPUs. The Cluster is combined with the use of cluster schedule software **Slurm** [61] to enable the user to submit their jobs when requiring GPUs.

Due to the fact that the Cluster is shared by multiple users, the number of GPUs used in different experiment varied depends on the number of current users. For the best cases where there are only a few current users, we used all 8 GPUs for (use entire node) training the TDNN. The time taken for running one entire experiment is approximately three days. The time taken could be reduced to approximately 1.5 days if the experiment based on the model in the previous experiment.

We are also provided with three CPU servers (Zamora,Tunguska,Lubbock) and one GPU server (Giger)[10] as additional computation environment for running the tasks when the MLP cluster is very busy. Each CPU server consists 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.

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

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

Chapter 5

Experiments & Results evaluation

In this Chapter, we would show the results of the experiments we have achieved. Note that the baseline model is run before others have done, and the noise training approach is decided after seeing the result of the baseline. The noise training was decided after we found that our baseline model performed poorly on the medical data, and we decide to use more data and data augmentation techniques to improve the performance of the model. Noise training, in theory, would build a noise-robust ASR system, and it aims to tackle one of the main challenge: the noisy acoustic environment.

Also, note that the number of experiments may seem to be quite small, as this is a large project, and the experiment is based on a simple existing recipe. However, the medical data was not available until midway of the project, and half of the length of the project is used for building the baseline model by adapting the recipe and fit the recipe to our computation environment.

5.1 Baseline

We built two models which are separately trained using individual headset microphone (IHM) and a single distant microphone (SDM1) dataset. The model with better performance would be chosen as the baseline model. Both models are built from AMI recipe[1].

Also, we actually built two versions for each IHM and SDM1, 16kHz and 8kHZ. 16 kHz IHM and SDM1 model are constructed by using the original AMI recipe without any modification (except some debugging due to change in computational environment). We initially thought the 16 kHz results in a degradation in performance. However, we found that the models could not be used at all for decoding the 8 kHz medical dataset due to the dimensional difference for the input layer in the neural network. Therefore, we do not have the result of 16 kHz models on the medical dataset. One solution to achieve the result is upsampling the medical dataset, but we decided not to do so as upsampling introduces many problems because it "makes up" the data. To build the 8 kHZ model, we change the sample rate to 8000 in configuration (mfcc.conf)

[1] `https://github.com/kaldi-asr/kaldi/tree/master/egs/ami/s5b`

when computing MFCCs and set the parameter allow-downsample to be true. We put results of 16 kHz models and 8 kHz models on AMI corpus in Appendix A.1, as we think they are not very important to our research.

The results of the models on the medical dataset are shown in table 5.1.

MODEL	S	D	I	WER	95%CI
IHM	9106	5957	446	78.63%	[77.63%,79.63%]
SDM1	10235	5763	648	84.39%	[83.40%,85.39%]

Table 5.1: a table showing the WER of models with IHM and SDM1 in 4 s.f. , where S stands for substituion, D stands for deletion, I stands for Insertion. CI is the confident interval. The total number of words in reference is 19725.

The result shows that the IHM model performs better than SDM1 model as it has less ($\approx 6\%$) WER. It has fewer errors in substitution and insertions errors, but higher deletion errors. Higher deletion error in IHM means that IHM model predicted shorter word sequence than SDM1 predicted. We think this might be due to the fact SDM1 is more sensitives to the noises as it needs to predict words spoken at a distance away from the microphone, so it tends to predict more words when encountering the random noise. However, IHM has significantly lower substitution errors than SDM1, which we think IHM does better for identifying a word. Experiment for running SDM1 model was set up as we thought the bystander might use the telephone as a single distant microphone while doing CPR. However, the results show that the telephone might behave in a way more similar to IHM rather than SDM1. As IHM is better than SDM1, we use IHM model as our baseline model.

Also, we show the performance of the baseline models on the AMI validation set and testing set in table 5.2. We think it is a good comparison to see the performance of models on the matched and the less matched dataset.

MODEL	AMI VAL	AMI TEST	MEDICAL TEST
BASELINE	19.3%	19.4%	78.63%

Table 5.2: a table showing the WER of the baseline model on AMI validation, testing and medical testing dataset (4 s.f.)

As the results show, the performance of the baseline model behaved poorly compared to its performance on AMI dataset, as we have expected. However, degradation is huge ($\approx 60\%$). We think there are a few possible reasons for the degradation:

1. The medical dataset is much less matched to the training dataset used for the baseline model. The training dataset is recorded in a meeting scenario, whereas the medical testing dataset is the phone calls made by the bystanders. This motivated us to use some of the medical data for training the model.

2. The medical dataset might be noisier than the AMI dataset, as the acoustic environment is more variant in former. It motivated us to build a noise-robust system.

3. The language model may not fit as the prior distribution of words for the medical dataset is different from the AMI dataset, e.g. words such as ”breath” is more. frequent.

We also have looked at at the WER details where errors are made. We found there are 164 substitution errors of the word ”OK” is predicted as ”OKAY” and 17 substitution errors of word ”Thats” predicted as ”That's”. Therefore, it seems some prediction should be correct but marked wrong. The actual WER is about 1% lower than the WER appeared. It could be a motivation for correcting the transcription, but it could be quite time-consuming to go through the 3-hour transcriptions. As we think the actual WER is likely to be within the confidence interval, we would keep the transcription untouched.

5.2 Train with more data

We investigate the performance of the model that uses more data. Firstly, we show the result of the model using 1.5 hours of extra AMI training. Note that the dataset contains 1.5 hours of extra data is just the original dataset before the split described in step 2 when constructing the baseline.

Then, we show the result of the model using 1.5 hours of medical data. Note that we do realize adding 1.5 hour AMI training data might not be clear to see the effect of using more AMI data, but the amount is chosen to have a better comparison to the use of the medical data. It might be better first to use half of the training data ($\approx$ 33.5 hours) and then add another half to test how performance is affected, but we decided not to do so as we are more interested in the effect of adding more medical data. The medical dataset is split as described in section 3.1 To combine the AMI training set and medical set, we used a scrip called combine_data.sh [2].

Both constructions of the model with extra AMI data or extra medical follow the same steps from step 3 for constructing the baseline model (that is using the AMI recipe by running run.sh script).

MODEL	S	D	I	WER	95%CI
BASELINE	9106	5957	446	78.63%	[77.63%,79.63%]
IHM_AMI	9593	5613	488	79.56%	[78.60%,80.54%]

Table 5.3: a table showing the WER of baseline model and the model trained with more AMI corpus,in 4 s.f. , where S stands for substituion, D stands for deletion, I stands for Insertion. CI is the confident interval. The total number of words in reference is 19725.

As a result shows in table 5.3, the model with extra training data results in a worse performance. There could be a few reasons for it. It could be that the model is overfitting, and the increasing of the AMI data decrease the generalization of the model. The substitution errors and insertion errors increased, but deletion errors dropped slightly.

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

It seems that the model with more AMI data has worse performance for identifying the word.

Therefore, we would not use the model with extra AMI data for further investigation, as it has increasing WER from the baseline model.

MODEL	S	D	I	WER	95%CI
BASELINE	9106	5957	446	78.63%	[77.63%,79.63%]
IHM_MEDICAL	5849	2088	981	45.21%	[43.77%,46.66%]

Table 5.4: a table showing the WER of baseline model and the model trained with more medical data, in 4 s.f. , where S stands for substituion, D stands for deletion, I stands for Insertion. CI is the confident interval. The total number of words in reference is 19725.

The result of adding medical data in training is shown in table 5.4. As a result shows, having 1.5 hours of medical data increased the performance of the model significantly. It is a massive improvement as it drops the WER from 78.63% to 45.21%. It has notable drops in substitution and deletion errors, although the insertion errors increased slightly. It means that the model now is predicting much less word sequence, which is closer to the ground-truth length of the word sequence. Also, it means that now the model has better performance for identifying a word.

We also have looked at the WER details in table 5.5 where substitution errors were made for the model with medical data. The errors are the most frequent (more than 10) substitution errors which are likely to be incorrectly marked. Therefore, the actual errors made by the model might be much less than the appeared errors. The actual WER might be $\approx$3% less than appeared WER. It now might seem to be worth to correct the transcription, but we would leave the transcription for now, as we have only one experiment left. It might be worth to do in further work.

Therefore, as the model improved significantly from the baseline mode, we would use the model with medical data for further investigation.

5.3 Noise Training

We now investigate the performance of the model after injecting noise to the training dataset. The training dataset now consists of both AMI corpus and medical training data. The noise used for injection is from Musan Noises [51]. Note that we only inject the noise as foreground noise. Term foreground noise used to describes some prominent noise, such as alarm and the sound of the phone ring. The script we used for injection follows partial code [line 148 - line 179] extracted from the run.sh of SRE recipe[3]. Overall, the process of noise training using Kaldi is:

1. Use cleaned HMM-GMM model trained with the medical model as the starting point.

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

REF	HYP	FREQUENCY
OK	<UNK>	259
SHES	<UNK>	52
HEEL	HE'LL	32
ITS	IT'S	29
WHATS	WHAT'S	26
SHES	SHE	24
UNTILL	UNTIL	19
SECCOND	SECOND	16
THATS	THAT'S	13
THATS	<UNK>	12
OK	OKAY	12
HES	<UNK>	10
SUM		504

Table 5.5: a table showing some of the top substitution error made for the model with more medical data predicting on the medical testing set. From left to right are the reference transcripts, hypothesis and the frequency of error, respectively.

2. Resample the clean data at 8KHz using a script (/utils/data/resampledatadir.sh [4])
3. Make a copy of clean data (medical data + AMI training data).
4. Inject the noise from Musam Noises dataset at into the clean data to create noisy data. The noise is injected with random SNR. There are four possible SNRs used for injection, which are [15,10,5,0].
5. Combine the noisy data and clean data to create a new training dataset.
6. Follow the same steps from step 10 described in implementation of baseline.

However, we do not have the result for the noise training. As we have mentioned in section 4.3, we are using MLP cluster for running the experiment. However, there are some unknown issues on the server, which leads to the server to be extremely slow. It would take more than one day for simple tasks such as computing the MFCC. Initially, we thought it could be our misbehaviours for using the cluster (it asks the user to put the dataset in the local scratch disk before running the job). However, we found that the cluster remained slow after we have cancelled all our jobs. Also, the speed of cluster remained the same after we copied the entire project into the local scratch disk and ran from the scratch disk.

There were few other available computation environments before. The one was using CPU servers (zamora,tunguska,lubbock) of CSTR. However, the CPU servers are currently having heave CPU tasks running, and we have tried to copy the project onto the server, but the speed remained slow. There is another GPU server which we used to run the model with medical data, but the servers were taken down owing to a cool-

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

ing problem. Also, we tried to use gcloud [5] for computation, but we found that the AMI corpus is unreachable from the gcloud. Therefore, we currently do not have any available computation resources to finish noise training.

It would only take less than two days to complete the noise training, as we have prepared the dataset and code. However, due to many coincident and the deadline of the project, we had to leave the result section out. Although we have not achieved the model for noise training, we deduce that the improvement should be within a few percents according to the observation of other relative works.

5.4 Overall discussion on the experiments

Although we have not achieved the noise training model, which should be our best model in the hypothesis, we have achieved a model with the WER around 45%. We think this is a reasonable result as we were provided with limited available data. A proper well-performed ASR system usually requires more than 100 hours of training dataset in the domain, to cover most of the patterns in the dataset for DNN to learn.

As we have seen in the experiment for the baseline model and the experiment of using more AMI data, the performance of the ASR degraded when moved to a different domain. With using only 1.5 hours of the medical dataset, the performance of the system improved more than 30%. It shows the importance of matched dataset, which answers one of our investigation questions.

There might be worries for overusing of the medical testing dataset, as the testing dataset should be seen only once in principle at the testing time. It might be better to split the original testing dataset into three: training, validation and testing set. However, there is a trade-off between using more training dataset and keep the testing set unseen. We think it is more important to use more in domain dataset to improve the performance of the model rather than keeping the medical dataset unseen. Also, the testing set is not actually been seen many times in our experiments, as there are only a few experiments. We think the negative effect of multiple uses of a testing set is negligible.

[5] https://cloud.google.com/compute/docs/gcloud-compute

Chapter 6

Conclusions & discussions

Overall, in this work, we have achieved the best performance of 45.21% WER for the model trained with AMI corpus and medical dataset, as our final model. Although we have not finished the noise training, we think the improvement would be within a few percents.

We have seen how the performance of the model degrades as the domain changes. Also, we have seen the effect of using more unmatched dataset resulted in a worsed performed model. The importance of matched dataset is demonstrated by 33.42% improvements from baseline to the model trained with the medical dataset, although only 1.5 hours of the medical dataset is used for training. Also, the actual improvements might be even larger if we correct our transcription for the medical dataset.

Although the noise training should make a more noise-robust system, the use of CMVN, DNN and iVectors should have also built a good noise-robust system. It is a pity that we have not finished the noise training, but we could conclude that a model with 45.21% WER is a reasonably good model considering the amount of medical data provided. Also, we think the results of the model might be higher if we are using the medical dataset from a mixture English speaker, as Scottish English is considered to have unique and different characteristics from other English speakers (native or non-native).

6.1 Limitation

One limitation of this work is the computation environment. The project could be finished by the time if the issues that we currently have is solved. Also, it would be faster to debug the coding and carry out more experiment if we are provided with a faster and private computation environment. The gcloud was one of the options for it, but the option is discarded as we think it is too expensive to use. Also, we did not expect all servers simultaneously having troubles to use.

One other limitation is medical dataset. Due to privacy issues, it is tough to get a medical dataset. It requires lots of negotiation between the School and medical teams. The medical data is finally available at the mid of the project, and it was the features drawn from the signals. It took many efforts to grant permission for using the raw

signal. We finally grant permission at the end of March, but it was quite late as we have other tasks in life at that time. We really cannot start further investigations before permission. Also, we could build a model with better performance if we have more of the medical dataset.

6.2 Further work

The experiments carried out in this work is mainly adapting the training dataset and apply the AMI recipe to built the ASR system. The language model is also built using the AMI recipe. However, we think it is worth to use a different language model which is adapted to the medical domain. The frequency of words, such as breath should have more weight than the previous model. We found the Medical corpus [10], which is an English corpus made up of texts collected from the Internet with a focus on the medical domains. It consists of 33 million words in total. We could use the distribution of the corpus as the language model for the system.

It might be worth to ask for the more medical dataset. As we have shown the importance of in-domain data in this work, the model could be improved significantly by using more medical data. We could also apply different data augmentation techniques to increase the quantity of medical data. It might also be worth to try SpecAugment methods. SpecAugment[46] augment the spectrogram of the signal by warping it in the time direction. It helps the DNN to be robust to a partial loss of frequency and small segments. Moreover, it has advantages of saving the computation cost compares to traditional data augmentation methods.

Also, noise injection and multi-condition training were how we traditionally train the model. A study [32] proposed a Noisy Adaptive Training (NAT) algorithm, an analogue to Speaker Adaptive Training we introduced in this work. In principle, NAT ”estimates the pseudo-clean model parameters directly without relying on point estimates of the clean speech features as an intermediate step”. As a result, the study shows relative improvements of 18.83% and 32.02% on Aurora 2 and Aurora 3, respectively. Although we have not finished the noise training, it might be worth to go further for NAT.

Bibliography

[1] Bishnu S Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *the Journal of the Acoustical Society of America*, 55(6):1304–1312, 1974.

[2] MARK BAKER. Chapter 2 - mixed signal test measurements and parameters. In MARK BAKER, editor, *Demystifying Mixed Signal Test Methods*, pages 35 – 61. Newnes, Burlington, 2003.

[3] Stig Nikolaj Blomberg, Fredrik Folke, Annette Kjær Ersbøll, Helle Collatz Christensen, Christian Torp-Pedersen, Michael R. Sayre, Catherine R. Counts, and Freddy K. Lippert. Machine learning as a supportive tool to recognize cardiac arrest in emergency calls. *Resuscitation*, 138:322–329, 05 2019.

[4] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. The ami meeting corpus: A pre-announcement. In *International workshop on machine learning for multimodal interaction*, pages 28–39. Springer, 2005.

[5] Christopher Cieri, David Graff, Owen Kimball, Dave Miller, and Kevin Walker. Fisher english training part 2, transcripts ldc2005t19, 04 2004.

[6] Christopher Cieri, David Graff, Owen Kimball, Dave Miller, and Kevin Walker. Fisher english training speech part 1 speech - linguistic data consortium, 12 2004.

[7] Corina de Graaf, Dominique N.V. Donders, Stefanie G. Beesems, and Rudolph W. Koster. Time to return of spontaneous circulation (rosc) and survival in out-of-hospital cardiac arrest (ohca) patients in the netherlands. *Resuscitation*, 130:e31–e32, 09 2018.

[8] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2010.

[9] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[10] Sketch Engine. English medical corpus from the web — sketch engine, 06 2015.

[11] EuReCaTWO. European registry of cardiac arrest - study two (eureca two), 2017.

[12] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

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

[14] John J Godfrey, Edward C Holliman, and Jane McDaniel. Switchboard: Telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520. IEEE, 1992.

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

[16] J. H. L. Hansen and T. Hasan. Speaker recognition by machines and humans: A tutorial review. *IEEE Signal Processing Magazine*, 32(6):74–99, 2015.

[17] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[18] Hans-Günter Hirsch and David Pearce. The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.

[19] Jonathan Hui. Speech recognition—acoustic, lexicon language model, 09 2019.

[20] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

[21] Martin Karafiát, Lukáš Burget, Pavel Matějka, Ondřej Glembek, and Jan Černockỳ. ivector-based discriminative adaptation for automatic speech recognition. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 152–157. IEEE, 2011.

[22] Slava Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401, 1987.

[23] Irina Kipyatkova and Alexey Karpov. Dnn-based acoustic modeling for russian speech recognition using kaldi. pages 246–253, 08 2016.

[24] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[25] Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R Rickford, Dan Jurafsky, and Sharad Goel.

Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14):7684–7689, 2020.

[26] Gakuto Kurata, Kartik Audhkhasi, and Brian Kingsbury. Ibm research advances in end-to-end speech recognition at interspeech 2019, 10 2019.

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

[28] Jinyu Li, Li Deng, Reinhold Haeb-Umbach, and Yifan Gong. *Robust automatic speech recognition: a bridge to practical applications*. Academic Press, 2015.

[29] Jinyu Li, Li Deng, Reinhold Haeb-Umbach, and Yifan Gong. Chapter 1 - introduction, chapter 2 - fundamentals of speech recognition. In Jinyu Li, Li Deng, Reinhold Haeb-Umbach, and Yifan Gong, editors, *Robust Automatic Speech Recognition*, pages 1 – 7, 9 – 40. Academic Press, Oxford, 2016.

[30] Andrew Maas, Quoc V. Le, Tyler M. O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y. Ng. Recurrent neural networks for noise reduction in robust asr. In *INTERSPEECH*, 2012.

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

[32] Yajie Miao, Hao Zhang, and Florian Metze. Speaker adaptive training of deep neural network acoustic models using i-vectors. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(11):1938–1949, 2015.

[33] Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38, 1994.

[34] Michael A. Nielsen. Neural networks and deep learning, 2018.

[35] The School of Computer Science at Carnegie Mellon University. The cmu pronouncing dictionary, 2019.

[36] Christopher Olah. Understanding lstm networks – colah's blog, 08 2015.

[37] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[38] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlíček, Y. Qian, K. Riedhammer, K. Veselý, and N. T. Vu. Generating exact lattices in the wfst framework. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4213–4216, 2012.

[39] Daniel Povey. *Discriminative training for large vocabulary speech recognition*. PhD thesis, University of Cambridge, 2005.

[40] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society, 2011.

[41] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[42] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755, 2016.

[43] Desh Raj. On lattice free mmi and chain models in kaldi, 05 2019.

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

[45] Robert Rehr and Timo Gerkmann. Cepstral noise subtraction for robust automatic speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 375–378. IEEE, 2015.

[46] Daniel S. Park and William Chan. Specaugment: A new data augmentation method for automatic speech recognition, 04 2019.

[47] Daniel Seita. 1000x faster data augmentation, 06 2019.

[48] Michael L Seltzer, Dong Yu, and Yongqiang Wang. An investigation of deep neural networks for noise robust speech recognition. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 7398–7402. IEEE, 2013.

[49] Nickolay Shmyrev. Training an acoustic model with lda and mllt feature transforms, 04 2017.

[50] Jocelyn Sietsma. Neural net pruning-why and how. In *Proceedings of International Conference on Neural Networks, San Diego, CA, 1988*, volume 1, pages 325–333, 1988.

[51] David Snyder, Guoguo Chen, and Daniel Povey. MUSAN: A Music, Speech, and Noise Corpus, 2015. arXiv:1510.08484v1.

[52] Mehdi Soufifar, Marcel Kockmann, Lukáš Burget, Oldřich Plchot, Ondřej Glembek, and Torbjørn Svendsen. ivector approach to phonotactic language recognition. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[53] Stephanie. The current challenges of speech recognition, 10 2019.

[54] Steve Young. A review of large-vocabulary continuous-speech. *IEEE Signal Processing Magazine*, 13(5):45–, 1996.

[55] Sam Sučík. Prosodic features in spoken language identification, 2019.

[56] Pawel Swietojanski, Arnab Ghoshal, and Steve Renals. Hybrid acoustic models for distant and multichannel large vocabulary speech recognition. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 285–290. IEEE, 2013.

[57] Søren Viereck, Thea Palsgaard Møller, Annette Kjær Ersbøll, Josefine Stokholm Bækgaard, Andreas Claesson, Jacob Hollenberg, Fredrik Folke, and Freddy K. Lippert. Recognising out-of-hospital cardiac arrest during emergency calls increases bystander cardiopulmonary resuscitation and survival. *Resuscitation*, 115:141–147, 06 2017.

[58] Olli Viikki and Kari Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1-3):133–147, 1998.

[59] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[60] Shi Yin, Chao Liu, Zhiyong Zhang, Yiye Lin, Dong Wang, Javier Tejedor, Thomas Fang Zheng, and Yinguo Li. Noisy training for deep neural networks in speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):2, 2015.

[61] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.

[62] Dong Yu, Michael L Seltzer, Jinyu Li, Jui-Ting Huang, and Frank Seide. Feature learning in deep neural networks-studies on speech recognition tasks. *arXiv preprint arXiv:1301.3605*, 2013.

Appendix A

Additional Data

Model	AMI Val	AMI Test
IHM_8kHZ	19.3%	19.4%
SDM1_8kHZ	35.1%	39.0%
IHM_16kHZ	18.6%	18.1%
SDM1_16kHZ	31.9%	35.7 %

Table A.1: a table showing the WER of the IHM and SDM1 model that trained in 8 kHZ and 16 kHZ on AMI validation, testing dataset (4 s.f.)