SPEAKER RECOGNITION USING DEEP NEURAL NETWORKS WITH REDUCED COMPLEXITY

by

Vidya Thanda Setty, B.E.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
December 2018

Committee Members:

Vishu Viswanathan, Chair

George Koutitas

Semih Aslan

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

DEDICATION

To the almighty God “Sri Krishna” for giving me the strength to finish my thesis.

To my parents Padma and Thanda Setty for believing in me*;* Ashwin Venkatesh for his motivation and support; family for their continuing support; to all my friends for their significant advice throughout the completion of my thesis.

ACKNOWLEDGEMENTS

I would like to sincerely thank my thesis advisor, Dr. Vishu Viswanathan for his immense support, patience and mentoring. This work would not have been possible without his involvement and guidance. His valuable advice and positive appreciation helped me complete my thesis successfully.

I would like to thank my thesis committee members for their encouragement and insightful advice throughout my thesis.

I would like to thank the engineers from NASA Johnson Space Center's Human Computer Interface branch, G. Salazar, A. Romero, and D. Juge, for suggesting the problem addressed in this thesis as a potential application in the International Space Station.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviation	Description
DNN	Deep Neural Network
ANN	Artificial Neural Network
MFCC	Mel-Frequency Cepstral Coefficient
PLP	Perceptual Linear Prediction Coefficient
CNN	Convolutional Neural Network
LSTM	Long Short-Time Memory
RNN	Recurrent Neural Network
ReLu	Rectified Linear Unit
CPU	Central Processing Unit
GPU	Graphical Processor Unit
SVM	Support Vector Machine
HTK	Hidden Markov Toolkit
CNTK	Computational Network Toolkit
SLS	Sequential Layer Specific
HMM	Hidden Markov Model

GMM	Gaussian Mixture Model
UBM	Universal Background Model
SNN	Shallow Neural Network
OBD	Optimal Brain Damage
DSSM	Deep Structured Semantic Model
TIMIT	Texas Instruments Massachusetts Institute of Technology
HTIMIT	Handset Texas Instruments Massachusetts Institute of Technology
LDC	Linguistic Data Consortium
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio

ABSTRACT

The goal of this research is to develop a small footprint text-independent speaker recognition system for a closed set of a relatively small number of speakers (e.g., 15-20). The problem was inspired by its potential application to the International Space Station (ISS) to determine which astronaut is speaking at a given time. In this research, the so-called Direct DNN based approach is used in which the output layer posterior probabilities are used to determine the identity of the speaker. Consistent with the small footprint design goal, a baseline DNN model was developed with just enough hidden layers and enough hidden units per layer, thereby reducing the total number of parameters, and by careful design to avoid the common problem of overfitting and to optimize algorithmic aspects including context-based training, activation functions, regularization, and learning rate. This baseline model was evaluated on two commercially available databases, clean speech TIMIT and multi-handset speech database HTIMIT, and on noise added TIMIT database that we created using four types of noises at three different signal-to-noise ratios (SNRs). The speaker recognition accuracy of the baseline is 100% for TIMIT, 96.75% for HTIMIT, and 100%, 98.75% and 98.125% for noise added TIMIT database at 20 dB, 10 dB and 5 dB SNR, respectively. This demonstrates that the baseline system has an error-free performance in relatively clean speech and a robust performance under telephone handset variability and in acoustic background noise. The baseline model has a total of 2.4M parameters. The rest of the work was devoted to

reducing the complexity of the DNN system by reducing the number of parameters without causing significant loss in performance. Initially, we used an adaptive pruning method where the parameters of all the layers are pruned simultaneously and the pruned system is retrained. The performance of this technique was evaluated on all the above-mentioned speech databases. We then developed a novel and enhanced pruning technique called Sequential Layer Specific (SLS) pruning. The SLS pruning technique performs pruning sequentially in multiple stages and in a layer-specific manner, followed by retraining after each pruning stage, while ensuring no or only minor performance loss in each pruning stage. The SLS pruning technique is significantly more effective than the adaptive pruning technique in terms of both model complexity reduction and speaker recognition performance loss. For the SLS pruned model, the speaker recognition accuracy is 100% for TIMIT database with 31X complexity reduction; 94.75% for multi-handset database HTIMIT with 4.5X complexity reduction. For noise added TIMIT database, with 1.7X complexity reduction there is no additional drop in speaker recognition accuracy relative to the baseline DNN in both 5 dB and 10 dB SNR and a 99.37% accuracy is achieved with 3X complexity reduction in 20 dB SNR. For cases where the speaker recognition accuracy is less than 100%, a higher "accuracy" is obtained using the "Top-two" performance metric in which recognition success is declared if the correct speaker lies in the top two choices predicted by the DNN model.

I. INTRODUCTION

1.1 Motivation

The topic of this thesis research is DNN-based text-independent, closed-set speaker recognition for a relatively small number of users (e.g., 15-20). Further goals are to achieve a lower complexity DNN and study the robustness of the system to acoustic background noise and telephone handset variability. The above-stated problem and associated goals were inspired by the requirements of the NASA Johnson Space Center (JSC) for their application to the International Space Station (ISS) [1]. We are in discussion with the NASA-JSC engineers from the Human Interface Branch in Houston. A low complexity solution with a low power requirement and a small footprint is important for ISS application.

Deep neural networks can learn complex functions using a large number of hidden layers, which provides the "depth" to the network. In some cases, fewer layers may also be capable of learning complex functions using the same number of parameters as "deep" models. So, it is not necessary to have deeper networks for all applications [2]. The advantage of having multiple layers is that they can learn features at various levels of abstraction. DNNs, in general, require a lot of data for training. Insufficient amount of data used to train the network may fail to produce reliable performance under test conditions. However, it is possible to improve the performance of a system using a suitable algorithm for training [2]. Applications such as search engines and Facebook and iPhone image searching tasks use deep learning. In these cases, providing sufficient data for training is not a problem because there are millions of users every day. However, DNNs with a complex design may not be necessary for a speaker recognition application

involving a closed-set of a relatively small number of users (e.g., 15-20) and a limited training database. In this case, it is possible to use a DNN with a smaller number of parameters. Instead of treating DNN as a black box, it is a goal of this research to use the knowledge of machine learning in configuring the DNN with just enough parameters.

1.2 Background

Technologies such as image recognition, speech recognition, speaker recognition and other biometric systems use, among other things, mathematical computations, pattern matching, and machine learning. In recent research, traditional methods like Hidden Markov Models (HMM) are being replaced by machine learning technology. The main scope of this thesis research is speaker recognition using Deep Neural Networks (DNNs). The characteristics of speech signal such as vocal tract structure, voice pitch, speaking style, etc., play an important role in speaker recognition. Early work on physiological components of speech production was published by Gunnar Fant in 1960, and this has led to extensive advances in speech research [3].

DNNs have been used recently with great success in many areas. Open-source toolkits such as Kaldi (a flexible speech recognition toolkit from Microsoft that supports discriminative training and linear transforms), Tensor Flow (software from Google that helps in building and training of neural networks) and CNTK (Computational Network Toolkit from Microsoft that trains deep learning algorithms) have been developed using machine learning [4] [5] [6]. Machine learning is a technology where the system acts like the human brain and makes decisions by learning from observed data. For instance, when a person sees an object the brain learns the features of the object and in the future, the brain recognizes that object based on the features it learnt. In the same way, in machine

learning technology the machine learns the features of the object and performs a recognition task. Earlier speaker recognition systems have used Hidden Markov Model (HMM) and Gaussian Mixture Model (GMM) [7]. Applications using an Artificial Neural Network (ANN) use a single layer of non-linear hidden units to predict HMM states. The research work described in [8] shows how the replacement of GMM by DNN gives improved results in a speaker recognition system. Several organizations including IBM, Google and Microsoft have succeeded in using DNNs for acoustic modeling of speech [9]. The speech signal has features including pitch, pitch jitter, and spectral parameters such as MFCCs (Mel Frequency Cepstral Coefficients) and PLPs (Perceptual Linear Prediction coefficients). These are some of the features that can be used in DNN for training the networks [9]**.**

1.3 Speaker recognition

Speaker recognition determines the identity of the person based on the characteristics of their voice. Speaker verification, on the other hand, is the process of verifying the claimed identity of the speaker. Speech recognition differs from speaker recognition as it recognizes *what* is being said and not *who* said it [10].

The research in speaker recognition has been increasing as speech-related applications are becoming popular. Speaker recognition can be classified into text-dependent and text-independent systems. In text-dependent speaker recognition, the speaker is prompted to say the text on which the system has been trained; in the text-independent case, the speaker can say *any* text and the system should be able to recognize the speaker. Text-independent speaker recognition is a more difficult problem than text-dependent speaker recognition because the former system must be trained on all possible

text contexts [10].

The speech spectral features such as MFCCs and PLPs can be used to train the DNN models. Voice activity detectors are used to remove the silence or noise frames from the speech signal [11]. In general, a context of +/- 5 or +/- 10 frames around the current frame is used to obtain a stacked vector of features to train the DNN model. DNN can be used as a feature extractor or a classifier. In the so-called indirect DNN approach, DNN bottleneck features are used as features for a secondary classifier like i-vector [12]. In the direct DNN approach, the frame level DNN posteriors from the output layer are combined by simply averaging to produce a single decision to recognize the speaker [12]. The so-called d-vectors are extracted from the last hidden layer and are used for speaker modeling. The d-vector based classifier has been shown to be robust to additive background noise and to outperform the i-vector classifier [13].

1.4 Deep Neural Networks

A Neural Network is a type of machine learning. The artificial neural network (ANN) is a feed forward network and is extensively used in pattern recognition [14]. The types of neural networks include Convolutional Neural Networks (CNNs), Long- Short-Term Memory (LSTM) networks, Recurrent Neural Networks (RNNs), and Deep Neural Networks (DNNs) [15]. This research is focused on DNN.

DNN is a multi-layer perceptron with three or more hidden layers and uses the stochastic gradient descent algorithm to initialize and update the weights used in all the layers [12]. Stochastic gradient descent uses a single *learning rate* throughout the training process. The Adam optimization algorithm is an extension of stochastic gradient descent, which optimizes the learning rate based on the first and second order moments (i.e, the

gradient mean and element-wise squared gradient, respectively). Adam has been shown to produce significantly better results for image and natural language problems [16].

A DNN with an input layer, an output layer and three hidden layers is shown in Figure 1. Each node of a hidden layer is the sum of the product of the inputs (x) and the weights (w). The weights are the connections or links between nodes. The nodes are sometimes referred as neurons. The learning rate defines the rate at which the weights are updated. One complete sweep over the training data is called an *epoch*. The training data is divided into *batches*; a number of batches are required to complete one epoch and are also referred as iterations. An epoch consists of many such iterations. A single neuron output is,

$$z = (x*w) + bias,$$

where the bias assists the activation function to better fit the data [17]. The node output is then subjected to a non-linear complex function such as sigmoid, softmax, Rectified Linear units (ReLu), Leaky ReLu, tanh, sine, logistic [18]. The softmax function is used at the final output layer to produce a probability vector. We initially used the sigmoid activation function but later found that the ReLu activation function is more suitable for our problem.

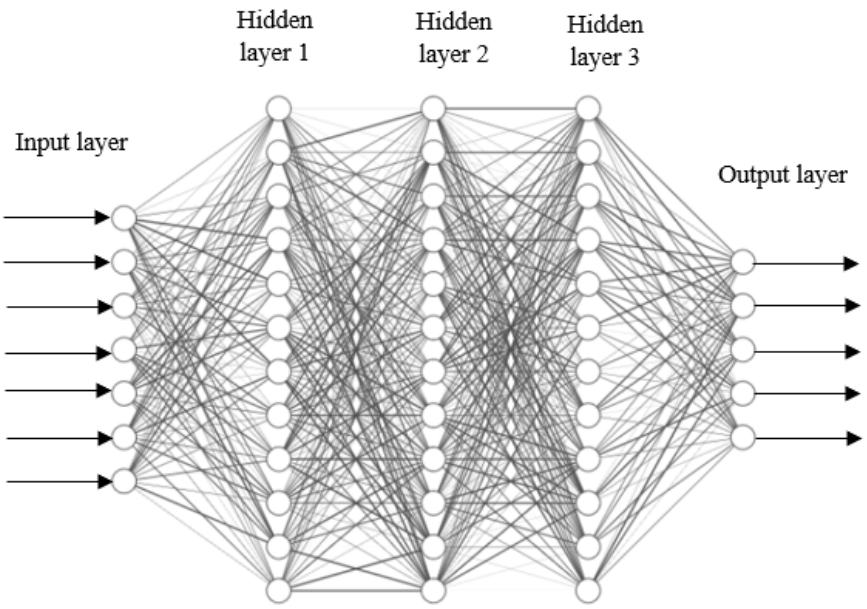


Figure 1. Deep Neural Network with 3 hidden layers

Sigmoid function

The sigmoid activation function maps the real-valued number to between 0 and 1. It is sometimes used in the output layer to render negative numbers to 0 and positive numbers to 1. The sigmoid function suffers from a vanishing gradient problem, and it is also computationally expensive [18]. The sigmoid function is graphically represented in Figure 2.

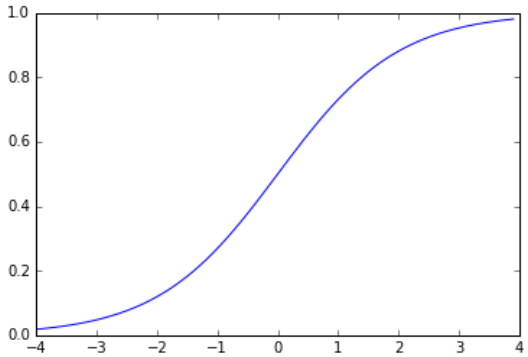


Figure 2. Sigmoid Activation Function

ReLu function

The Rectified Linear unit activation function outputs 0 for an input 'x' if x is less than 0, and it outputs x for x greater than 0. This activation function doesn't suffer from the vanishing gradient problem in the positive region, but it does in the negative region [18].

The Leaky ReLu addresses the vanishing gradient problem in the negative region. However, many researchers use ReLu and have reported satisfactory results [18]. The graphical representation of ReLu function is shown in Figure 3.

$$A(x) = max(0, x)$$

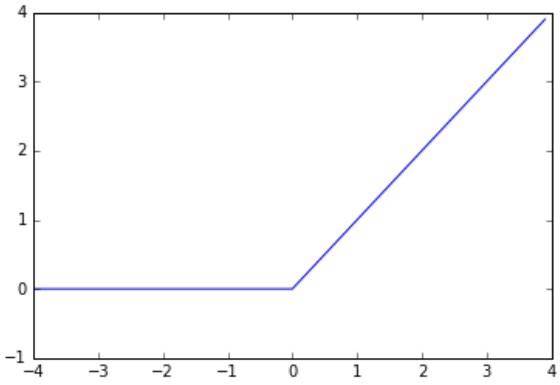


Figure 3. ReLu activation function

1.5 Research objectives

In current research, DNNs are treated generally as a black box by giving extensive data for training. This thesis research is focused on how to reduce the complexity of the DNN that matches available resources. Any computer with a high-power CPU may be sufficient for training the networks. However, an add-on GPU (Graphical Processor Unit) from Nvidia, for example, or a GPU enabled computer will

enable the training to be done faster. We used an NVIDIA GEFORCE 940MX GPU enabled laptop.

A fully connected DNN delivers high accuracy but it requires high computational complexity and large storage. The objective of this research is to design a low complexity DNN reducing power consumption, storage, computational resource and latency, without significant performance loss. Different approaches are investigated to discard unnecessary parameters and the network is retrained using only the retained parameters. The complexity reduced DNN results in a smaller footprint solution requiring lower compute power and less storage.

The research was focused on developing a baseline DNN configuration with just enough hidden layers and hidden units. Further work was carried out to find out a suitable activation function, learning rate, and weight-updating algorithm. After a brief search and advice from experts, we decided to use the CNTK Toolkit for modeling the neural network, as it is efficiently implemented for addressing speech-specific problems. The commercially available clean speech database TIMIT was chosen because it is widely used by speech researchers [19] [20]. The research is also extended to explore the robustness of the DNN solution over different conditions including telephone handset variability and types and levels of acoustic background noise. The effect of handset variability on the DNN solution is studied using the multi-handset database HTIMIT [21] [22] and the robustness to noise is studied using a noise added TIMIT database, which we created using four types of noise at three different signal-to-noise ratios.

The goal of this research is to use DNN not only to predict the speaker identity but also to determine and retain only the essential connections of the network.

The fully connected neural network is trained and then using a suitable algorithm the parameters that are not necessary are discarded, and as a result DNN with a reduced complexity is obtained. The simplified DNN is then *retrained* to get performance close to that of the initial, fully connected DNN. Performance results from both cases are compared in various conditions using TIMIT, HTIMIT, and noise-added TIMIT databases. There might be less frequent cases where parameters considered as unnecessary might turn out to be relevant; so, it is important to analyze each layer carefully before deciding what parameters to discard [23].

The network becomes sensitive after *pruning* to discard selected weights and could degrade the performance of the system. The algorithm we have developed for discarding parameters is novel and effective as compared to other pruning techniques, as it is resistant to sensitivity and it allows smooth learning of parameters, thereby allowing the network to perform the speaker recognition task reliably. The pruning takes place sequentially in multiple stages ensuring that there is no huge drop in accuracy in each stage; this novel design makes the algorithm unique and effective. Thus, the resulting simplified DNN uses less computational resource and storage without significant loss in speaker recognition accuracy.

1.6 Thesis Outline

The rest of the report is arranged as follows. A focused literature review on speaker recognition methods prior to DNN as well as the methods using DNN for speaker recognition, including the complexity and overfitting problems of the DNN models is provided in Chapter II. Then we describe in Chapter III the proposed approach of using DNN for small footprint speaker recognition. This chapter includes a discussion of the

chosen commercially available speech databases and open source software used in this research. In Chapter IV, we describe the development of the baseline DNN system and its performance results on various databases. In Chapter V, we describe the adaptive-threshold pruning method, followed by a novel Sequential Layer Specific pruning technique. Performance results on various databases are also provided for both pruning methods. In Chapter VI we describe the final, small footprint DNN solution and its performance on various databases. Chapter VII contains conclusions and future work. A list of references is included at the end.

II. LITERATURE REVIEW

2.1 Speaker recognition methods prior to DNN

Earlier speaker recognition systems have used HMM (Hidden Markov Model), SVM (Support Vector Machine) and GMM (Gaussian Mixture Model). SVM is one of the discriminative classifiers. The SVM *binary* classifier models the decision boundary between two classes as a *separating hyperplane.* SVMs use "supervectors" as input which is obtained by combining several small dimensional vectors into a single higher-dimensional vector [24]. GMM is a stochastic model that has become the *de facto* reference method in speaker recognition [24]. The speech signal contains linguistic and speaker-specific information. Gaussian components are used to represent the speaker-dependent spectral shapes, and the Gaussian mixtures are used to model the arbitrary densities [7]. The long-term averages of acoustic features are used to average out the phonetic variations and preserve only speaker-specific information. A similar approach is used in a Gaussian classifier and has been used for text-independent speaker recognition. The averaging process requires the speech utterances to be long enough to obtain the long-term averages; otherwise it may lead to loss of speaker specific information [7]. For text-independent speaker recognition, the HMM-based speech recognizer was used for segmentation at the front end in an attempt to improve the performance, but this resulted in a huge computational complexity and only a minor increase in accuracy. Gaussian mixture densities were used for speaker identification [7]. Telephone handset variability in the speech database causes performance degradation in speaker recognition. The telephone channel effects are non-linear in nature and these effects are coupled with

speaker specific information and make it difficult to isolate and remove them from the features, thereby degrading the system performance [25].

The reference [26] describes two approaches: Sparse speaker representation approach and discriminative regularization approach to train the Universal Background Model. These approaches use expectation-maximization algorithm to train the Universal Background Model. A Vector Quantization (VQ) technique maps the vectors from a large vector space to a finite number of clusters. The center of each cluster is called the codeword. The acoustic features are used to generate a VQ codebook. Speakers can be discriminated based on the location of the centroids in a cluster. Vector quantization is not efficient for a large database and hence neural network techniques can be used to improve the speaker recognition performance [27].

The paper [11] describes speaker identification system using Gaussian mixture speaker models. The author of this research work achieved 82.8% accuracy for a closed-set (100 speakers) speaker recognition using the Switchboard database. The GMM based speaker recognition system requires extensive resources to achieve state-of-the-art performance. The feature sub-sampling discussed in [25] increases the training speed. The performance of the recognition system can be increased by increasing the inter-speaker variability in the UBM data [25]. The GMM model performance will be degraded due to the channel or microphone conditions of the speech data [25]. Joint Factor Analysis is used to compensate for the effects due to the channel variability [24]. Voice activity detectors are used to remove the silence or noise frames from the speech signal and get significant improvement in the performance [28]. The enhanced GMM

based systems such as Joint Factor Analysis, Eigenchannel, etc., make the system rather complex and not suitable for small footprint implementation.

Hidden Markov Models are used to represent the probability distribution over the sequence of observations. They are widely used in speech applications. Prior to the introduction of MFCCs, Linear Prediction Coefficients (LPC) were used as feature vectors for the intended recognition tasks. LPC represent the resonance property of vocal tract and these are the main feature type used in HMM classifiers. HMMs are used to build an acoustic model for each speaker. In the recognition phase, these models are compared against the target speaker models. Vector quantization is used to compress the feature vectors. HMM combined with VQ gives significant improvement in performance in both text-dependent and text-independent speaker recognition rather than using VQ alone [29].

2.2 DNN based speaker recognition

Over the last few years, deep learning has gained wide acceptance in speech processing research. Several organizations including IBM, Google and Microsoft have succeeded in using DNNs for acoustic modeling of speech [9]. DNN can be used as a feature extractor or as a classifier [12].

2.2.1 Indirect DNN approach

The indirect method is one of the two approaches in designing a speaker recognition system where the trained DNNs are used to extract the features and then these features are used to train a secondary classifier for the intended speaker recognition task. The DNN that was trained for a different purpose could be adapted and used for a

different task. For instance, the DNN trained for automatic speech recognition could be used for speaker recognition or language recognition. The DNN output posterior probabilities and bottleneck features are used as features for the secondary classifier like i-vector [12]. A voice activity detector is used to extract speech-only segments. The 600-dimensional i-vectors are extracted from stacked mean feature vectors and are length normalized. The mean over the training data is used as a target model for language or speaker recognition task [12]. Figure 4 shows the DNN architecture used in the indirect DNN approach.

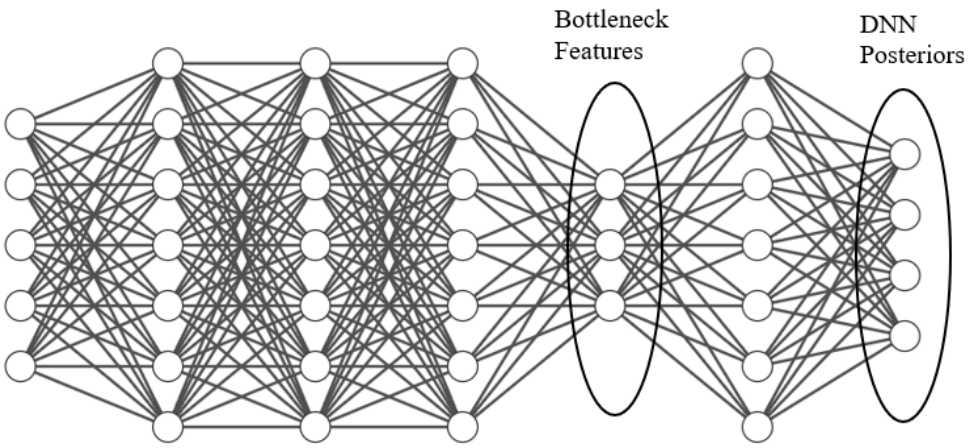


Figure 4. DNN architecture used in the indirect DNN approach

The DNN is trained at the frame level to classify speakers. The features from the last hidden layer are extracted and used for speaker modeling. The average of these feature vectors over the training database is called as deep vector or d-vector, and it is considered as speaker model. From a test speech utterance, the d-vector is extracted by passing it through the DNN layers and then compared with the saved target model to perform the recognition task. The d-vector classifier provides a robust performance in acoustic background noise. The d-vector approach has been shown to outperform the i-vector approach [13].

2.2.2 Direct DNN approach

In the direct DNN approach, the trained neural networks are used as a classifier to recognize the speaker. The input layer in the direct approach represents the dimension of the input spectral features, followed by 3 or more hidden layers and an output layer. The dimension of the output layer is equal to the number of speakers the system is designed to identify. The frame level DNN posteriors from the output layer must be combined by simply averaging over the test utterance [12].

For a small footprint, low resource application it is possible to use DNN to train and predict the speaker without using a secondary classifier. The use of a secondary classifier itself requires additional computational resource, which is not suitable for a small footprint system. The performance of the system may be improved slightly by increasing the number of hidden layers, but it comes at the cost of increased complexity. During testing, the frame level aggregated DNN posteriors are averaged to produce a single decision [12]. The direct DNN architecture for speaker recognition is shown in Figure 5.

The mini-batch stochastic gradient descent algorithm speeds up the training process by processing the training data in small batches. DNN learns the basic features of the speaker in the lower layers and learns complex features in the higher layers [30]**.** The text-independent speaker recognition system is a relatively complex problem as it requires all possible speech contexts for training. To achieve a robust performance, the system should be trained in all operating conditions including background noise and telephone handset variability.

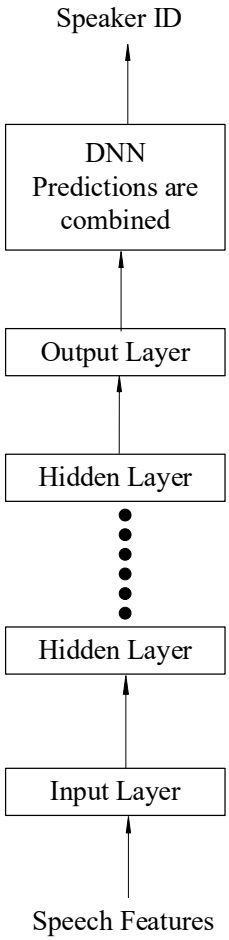


Figure 5. Direct DNN approach

2.3 Complexity of Deep Neural Network models

A complex DNN with a huge number of parameters requires extensive training data. Sometimes the number of parameters in the neural network exceeds the number of data samples available for training. Reducing the number of parameters in a neural network also reduces the CPU resources required to train. The DNN algorithmic aspects include the activation function and learning or updating algorithms. Speech recognition using the TIMIT database and image recognition using the CIFAR-10 database have been developed using shallow feedforward networks with parameters less than deeper-layer DNN, with performance comparable to deeper-layer DNN [2]. The just-cited work uses a

model compression method to train the model. Model compression is a technique where synthetic labeled data is obtained by passing the unlabeled data through a complex model (DNN) and this synthetically labeled data is used to train shallow neural networks (SNN). SNN is then trained to learn the same function that was already learned by the complex model. This shows that SNN could also learn a complex function similar to DNN [2].

The technique described in [31] reduces the number of parameters in the neural network based on the fact that the optimized weights of the neural networks tend to be structured. The technique of discarding less important parameters from the neural network with only a small performance loss is called as pruning [32]. Two types of structured pruning described in [32] are regularization-based pruning and importance-based pruning. In the regularization based pruning method, weights are divided into different groups based on their importance and each group uses a different regularization parameter resulting in zeroing out the unimportant parameters [32]. For image application, the proposed approach described in [32] achieved 4X speedup and 0.8% accuracy loss for AlexNet. The greedy network pruning called as fisher pruning [33] yields a 10x speedup. Fisher pruning initially focused on investigating performance change by removing a single parameter and then greedily removing the parameters until there is no significant drop in accuracy [33]. This technique reuses the already computed base model's gradient information, which makes it easier to implement.

DNN is trained to learn both connections as well as weights. The pruning technique is compared to a mammalian brain. During the early stage of a child's development process the synapses are created and then the little-used ones are gradually pruned in the later stages [34]. The iterative pruning and retraining of network ensure

only a small loss in accuracy but a huge reduction in the complexity of the network. L2 regularization gives better results compared to L1 regularization. [34]. The iterative pruning technique outperforms the single-step aggressive pruning. Image recognition implemented using the MNIST database and AlexNet show that pruning reduces computational resource without significant loss in accuracy. This technique reduces the number of parameters in AlexNet from 61M to 6.7M with no loss in accuracy [34]. The optimal brain damage (OBD) based pruning outperforms the magnitude-based pruning. The OBD based pruning cuts the network complexity by half, facilitating the use of limited training data [35].

2.4 Overfitting problems with deep neural network models

The multiple non-linear hidden layers learn complex features of input. However, fully connected DNN models are prone to overfitting [36]. The increase in the complexity of the network can overfit the data but may fail to generalize for test data that is not seen under training. The condition where DNN model yields poor performance in both training and validation data is referred as an underfitting problem; hence it is important to make sure that the network yields good performance on training data before dealing with the overfitting problem [37].

Early stopping is one of the techniques used to address the overfitting problem. The technique involves careful observation of the training process by analyzing the evaluation metrics and stopping updating of weights as the average error on the validation set starts increasing. The practical validation error curve usually has two or more local minima, so it is important to stop training at the smallest local minima [36]. Figure 6 shows the overfitted, underfitted and appropriate fit neural network models.

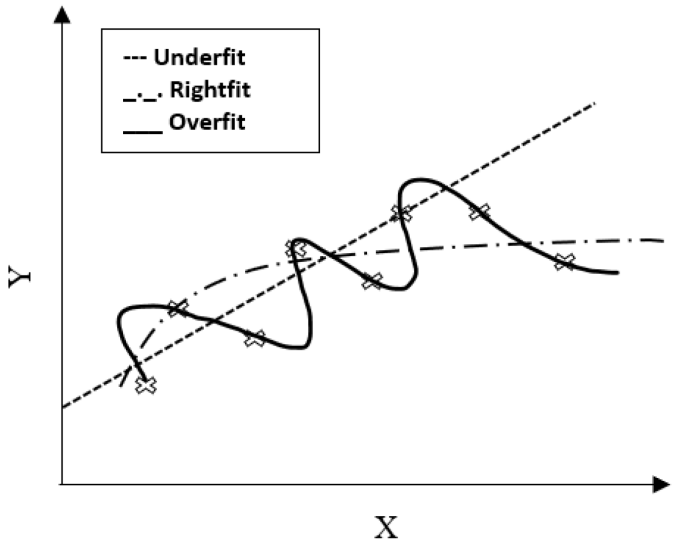


Figure 6. Overfit, Underfit and appropriate fit of Neural Network

Regularization also helps in overfitting. L1 and L2 are the two types of regularization technique. The regularization parameter is a hyperparameter added to the cost function to reduce overfitting. The most commonly used regularization is L2 regularization; also known as weight decay, it acts as a scale factor that drives the weights towards zero and reduces the error rate. However, imposing large regularization sometimes fails the learning algorithm like gradient descent [38]. Experimental results on VGG-16 [39] and AlexNet [40] architectures show regularization helps in increasing the accuracy but suddenly exhibits 1% drop in accuracy for a large regularization. Delayed Strong Regularization is also one of the techniques used to addresses the gradient diminishing problem in learning method [38].

The so-called dropout technique significantly reduces the overfitting problem and outperforms regularization techniques. This technique significantly improves the performance in speech, image and document classification problems [37]. The limited training data and extensive training yield an overfitted model. A complex neural network

requires a large amount of training data. When the available training data is limited it is possible to use the dropout method. It is a technique of temporarily removing certain hidden units during training. The choice of removing the hidden units from the network is done randomly. For instance, 50% dropout removes half of the hidden units. It is possible to use a different dropout rate for each hidden layer [37]. High momentum is a technique used in neural network to speed up the learning process. There might be cases where large momentum and learning rate make the network weights to grow very large; in such scenarios it is possible to use Max-norm regularization technique to address this problem. Using dropout along with max-norm regularization, decaying learning rates and high momentum provide a significant improvement in performance [37]. Experimental results on TIMIT, MNIST, CIFAR-10, and ImageNet databases have all shown the effectiveness of using dropout in combatting the overfitting problem. However, many researchers use dropout alone and have reported satisfactory results. [41].

The Gaussian dropout described in [37] multiplies the output of the hidden units by Gaussian noise. This dropout method increases the training time by 2-3 times the original training without dropout [37]. The experimental results on the TIMIT database using a 6-layer network shows that this dropout technique decreases the phone error rate from 23.4% to 21.8% [37]. It is important to remember that the dropout method doesn't decrease the complexity of the network as the nodes of the neural network are just ignored in that particular updating step during training but are not removed from the network. Figure 7 shows the fully connected network with and without dropout.

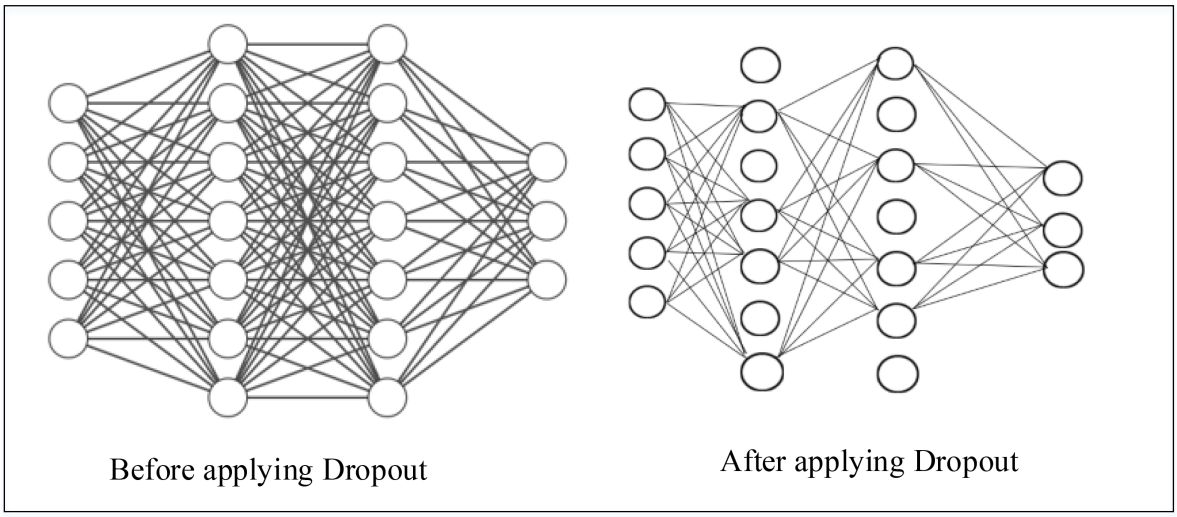


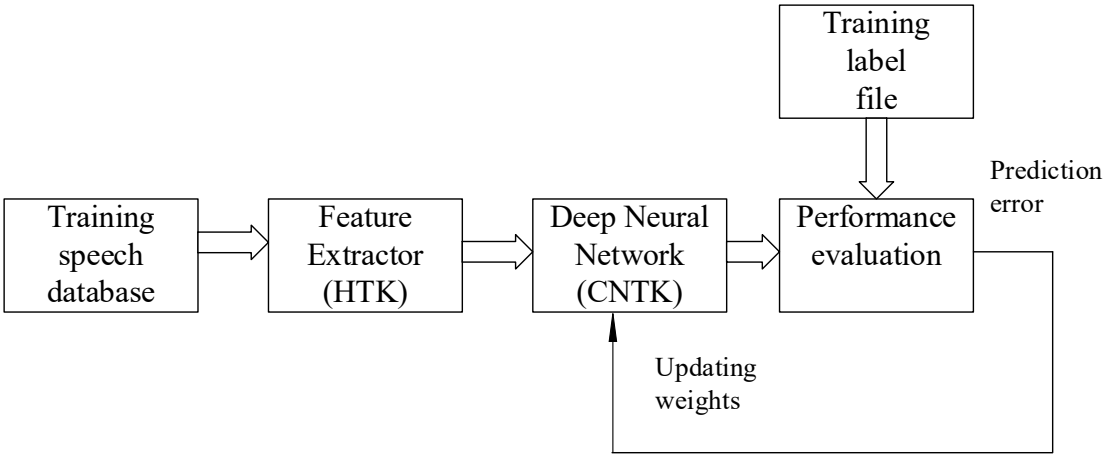
Figure 7. Dropout technique

III. PROPOSED APPROACH

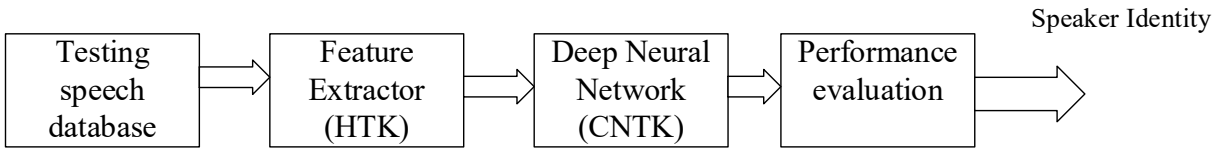
3.1 Direct DNN based solution

The proposed approach of this research is illustrated in Figure 8, where the top diagram shows the training process and the bottom diagram shows the testing process. For training and testing, we used two commercially available databases, clean speech TIMIT database and multi-handset database HTIMIT, and a noise added TIMIT database we created using four types of noise at three different signal-to-noise ratios (SNRs). We used 80% of the data for training and 20% of the data for testing. We used the HTK Toolkit as feature extractor to extract the MFCCs from the speech data. A stacked vector of MFCC features is used as input to the DNN. We used CNTK toolkit to develop the DNN model and the direct DNN based approach for the intended speaker recognition, which is explained later in this chapter. In training phase, the prediction error from the DNN output is used to update the weights and train the DNN model. The trained DNN model is used in testing phase, and the DNN posterior probabilities are used to determine the identity of the speaker.

With this proposed approach, the following tasks were carried out in this research: (1) developing a baseline DNN system; (2) evaluating the performance of the baseline system over the three speech databases mentioned above; (3) investigation of complexity reduction techniques; and (4) evaluating the performance of the final complexity reduced system over the same three speech databases. The rest of the chapter explains the details of HTK, CNTK, and speech databases used in this research.



(a) Training



(b) Testing

Figure 8. Block diagram of the proposed approach

3.2 HTK TOOLKIT

To extract MFCC feature vectors, the speech signal is pre-processed using a pre-emphasis filter and then divided into frames. A Hamming window is applied to each frame to reduce the spectral leakage. Each frame is then processed using Fast Fourier Transform and the resulting frequencies are linearly spaced into a Mel-frequency filter bank. Finally, a logarithmic transformation followed by cosine transformation is applied to generate Mel-Frequency Cepstral Coefficients [24]. We calculated MFCCs using the Hidden Markov Toolkit [42]. A configuration file is first created, specifying the required parameters like window size, sampling rate, etc., along with a text file specifying the

location of the .wav file and .mfc file and then use HTK's HCopy tool to extract the MFCC feature vectors, using the following command.

HCopy -T 1 -C config_wav2mfc -S convert.scp [42]

The 39-dimensional MFCC features are extracted from the speech signal. The feature vector of each frame has 12 MFCCs, 1 normalized energy, 13 delta, and 13 delta-delta coefficients. CNTK expects as input HTK format MFCC files and label files. It expects the start and end of the frame in the label file. We used HVITE tool [42] to create the label files containing information about the start and end of the frames. The generated label files are then formatted using a python script for use in CNTK.

3.3 CNTK TOOLKIT

CNTK is one of the popular open-source deep learning software toolkits [5]. It is used to implement deep learning architectures, such as Convolution Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Deep Structured Semantic Models (DSSM), and Deep Neural Networks (DNNs). CNTK supports Windows and Linux platforms. Also, it supports popular data such as plain text, speech, images, and binary [5]. CNTK trains deep learning models faster than other available open-source deep learning libraries like TensorFlow, Theano, Caffe, etc., [43]. The compatibility of CNTK library with C++ and Python allows the user to customize the built-in algorithms. This library can be efficiently implemented on a CPU, single or multiple GPU [5]. We chose to use CNTK as it is efficiently implemented for speech applications. CNTK is 2 to 4 times faster than TensorFlow [43]. CNTK is best suited when the resource availability is limited. One can use Brain Script or Python to implement their neural network models. CNTK supports sparse data computation but

doesn't support sparse weight computation. All DNN experiments described in this thesis were performed on an NVIDIA GEFORCE 940MX GPU enabled laptop, using the Microsoft Windows operating system.

CNTK requires as input MFCC files and the corresponding label files of the speech data to train the DNN model. As mentioned above, we used the HTK toolkit to extract features from the speech data. The CNTK library provides a function to read the HTK format feature and label files.

3.4 Speech databases

DNN requires a lot of data for training if it has a large number of parameters. Our research is focused on small footprint closed-set speaker recognition. Further goals are to develop a reduced complexity DNN and demonstrate robust performance under telephone handset variability and in acoustic background noise. After investigating commercially available databases and with advice from experts in the field, we chose TIMIT database for DNN-based speaker recognition system development and HTIMIT database to evaluate its performance robustness to handset variability. Also, we created a noise-added TIMIT database, by adding four types of noise to clean TIMIT speech at three different signal-to-noise ratios, to evaluate the robustness of the system under different acoustic background noise conditions.

3.4.1 TIMIT database

The TIMIT (Texas Instruments Massachusetts Institute of Technology) corpus is widely used for the development and evaluation of speech related applications. It is commercially available from the Linguistic Data Consortium (LDC) [20]. It contains the

recordings of 630 speakers of eight different dialects. The speech was recorded using a high-quality microphone at a sampling rate of 16 kHz [20]. The TIMIT corpus contains phonetically rich speech waveform files for each utterance as well as corresponding phonetic and word transcription files. The corpus contains two dialect sentences (SA), three phonetically diverse sentences (SI) and five phonetically compact sentences (SX) for each speaker. The same two dialect sentences (SA) are spoken by all speakers [20]. The TIMIT database files are in SPHERE format. We used the sox tool [44] to convert the files from SPHERE format to wave format.

As our research is focused on small footprint, text-independent speaker recognition, we selected randomly 20 speakers (10 male and 10 female speakers) representing all 8 dialects.

3.4.2 HTIMIT Database

The HTIMIT database [21] is used in this research to evaluate the effect of telephone handset variability on the performance of our DNN-based speaker recognition system. The HTIMIT database is created by playing a subset of TIMIT database from a loudspeaker though ten different handsets [22]. The TIMIT database is downsampled from 16 kHz to 8 kHz before playing them through the loudspeaker. The ten different handsets were used to collect the HTIMIT database. The ten handsets include nine telephone handsets and a high-quality microphone [22]. Of the ten handsets used, four of them had carbon button microphone (cb1, cb2, cb3, cb4), four of them has electret microphones (el1, el2, el3, el4), one was a portable (cordless) phone (pt1), and the tenth one was a Sennheizer high-quality head-mounted microphone (senh).

To have variable sound characteristics, different transducer designs were chosen

as a criterion to select carbon-button handsets and different transducer grill designs were chosen as a criterion to select different electret handsets. Handsets cb3 and cb4 were selected to evaluate the robustness of the system in poor handset conditions [22]. The HTIMIT database files are also in SPHERE format; we used the sox tool to convert the files from SPHERE format to wave format.

3.4.3 Noise added TIMIT database

To evaluate the robustness of DNN based speaker recognition in acoustic background noise conditions, we created a noisy database by adding noise to the clean TIMIT speech. We selected four different noise types: moving car noise, wind noise, white noise and city traffic noise [45]. The noise files were sampled at 16 kHz.

The noise files were added to the TIMIT speech files at three signal-to-noise ratios: 5 dB, 10 dB, and 20 dB. The noise power for individual noise file was calculated and a fixed scale factor K was then calculated and applied to reduce the energy of the noise signal for a given SNR. Finally, the resulting noise file is added to the clean speech file producing a noise added speech file. We wrote a MATLAB function to add noise at specified SNR to clean speech. We used a MATLAB tool to estimate the SNR of the noise-added speech files and verified that the SNR was, indeed, correct. The mathematical expression to calculate the factor K is given by,

*K = (signal_power/noise_power) * 10^(-(SNR/10))*

IV. DEVELOPMENT OF THE BASELINE DNN SYSTEM

4.1 DNN Configuration

There is no thumb-rule for selecting the number of hidden units to use in DNN. As this research work is focused on designing a closed-set small footprint speaker recognition system, it is our goal to design a model with just enough parameters. Hence, we conducted experiments to find the right number of hidden units per layer for our problem that are applicable to all three databases: TIMIT, HTIMIT, and noise added TIMIT. We initially trained the network on full utterance speech data, but it failed to give acceptable results, as it is not considered as an effective training approach in speech applications [12]. For a relatively small set of speakers (20 in our case), it is expected that we get 100% speaker recognition accuracy for clean speech. However, the full utterance training produced significantly lower performance. Hence, we investigated the context-based training approach, where a stacked set of spectral features (MFCCs) extracted from speech frames over a context window of 5 frames to the left and 5 frames to the right around the current input frame is used as feature vector input to DNN [12]. A 39-dimensional MFCC feature vector is extracted for each frame of the speech signal. With context-window being 11 frames wide, the dimension of the input layer is 429 (=11x39). The output layer dimension is 20, as we are developing a speaker recognition system for 20 speakers.

The classification or prediction error is calculated during training to assist the back-propagation technique in updating the weights. We initially used the stochastic gradient descent (SGD) algorithm to update the weights. SGD uses a single learning rate throughout the training process. We then decided to use Adam optimization, which gave

comparatively better results. Adam optimization algorithm is an extension of SGD, and it optimizes the learning rate based on the first and second order moments of the gradients. Adam is widely used in natural language problems [16]. The learning rate specifies the rate at which the weights are updated. We used the learning rate of 0.001. A larger learning rate increases the training speed, but the gradient descent can overshoot the minimum, thereby failing to converge or even diverging, resulting in ineffective training. We experimented with different learning rates in the exponential range of 0.1, 0.01, 0.001 and found that the learning rate of 0.001 provides good overall performance for our training data. We conducted the following experiments using HTIMIT database.

The hidden layer units are a linear function of their inputs. The activation function transforms the linear hidden units into a non-linear function. We initially used sigmoid as the activation function, but it resulted in the vanishing gradient problem. To address this problem, we switched to the ReLu activation function.

The accuracy of the model increased with the increase in the number of hidden units per layer but reached a plateau; any further increase produced no additional increase in accuracy. There is a risk in that too many hidden units can cause overfitting and hence lower performance over test data.

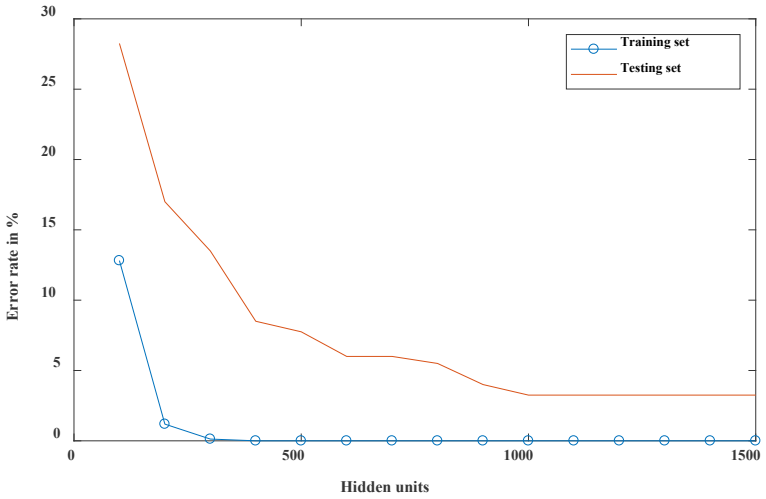


Figure 9. Performance of the network for various hidden number of units per layer

We used 3 hidden layers for this experiment. Seen from Figure 9, using 1000 hidden units per layer produces good performance.

After deciding the number of hidden units per layer, experiments were carried out to find the appropriate number of hidden layers. The performance of DNN can, in general, improve with increase in the number of layers. However, for our problem of speaker recognition for a closed set of 20 speakers, it is sufficient to use just three layers. Anything beyond 3 hidden layers reached a plateau. Let us recall that this research is focused on designing a DNN solution with just enough hidden layers and enough hidden units per layer. Using 3 hidden layers and 1000 hidden units per layer meets this goal.

Neural networks tend to overfit the training data when the available data for training is limited. One can address this issue by adding more data for training such that DNN will be trained on all possible training data and decrease the generalization error for unseen new test data. In our experiments, we observed overfitting and the resulting high generalization error. We used the dropout technique to solve this issue; however, it is

important to use a proper level of dropout as otherwise it will result in underfitting, thereby yielding a lower performance even in training. In order to achieve good overall performance of DNN, we carefully experimented on the use of different dropout rates as illustrated in Figure 10 and found that the dropout rate of 30% is best suited for our problem.

The dropout technique *randomly* ignores certain hidden nodes during training. It is important to note that the dropout technique helps in resolving the problem of overfitting but doesn't help in decreasing the complexity of the network because the nodes and connections of the network are just ignored while training and are not removed from the network.

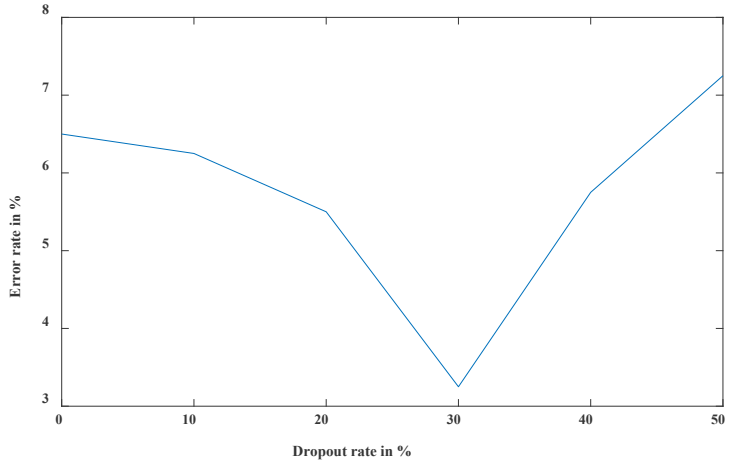


Figure 10. Performance of the network on different dropout rate

One complete sweep over the training data set is called an *epoch*. Using more epochs will help DNN to learn from the prediction errors and progressively optimize the weights, thereby yielding progressively better performance. Training the DNN more than required might result in an overfitted network.

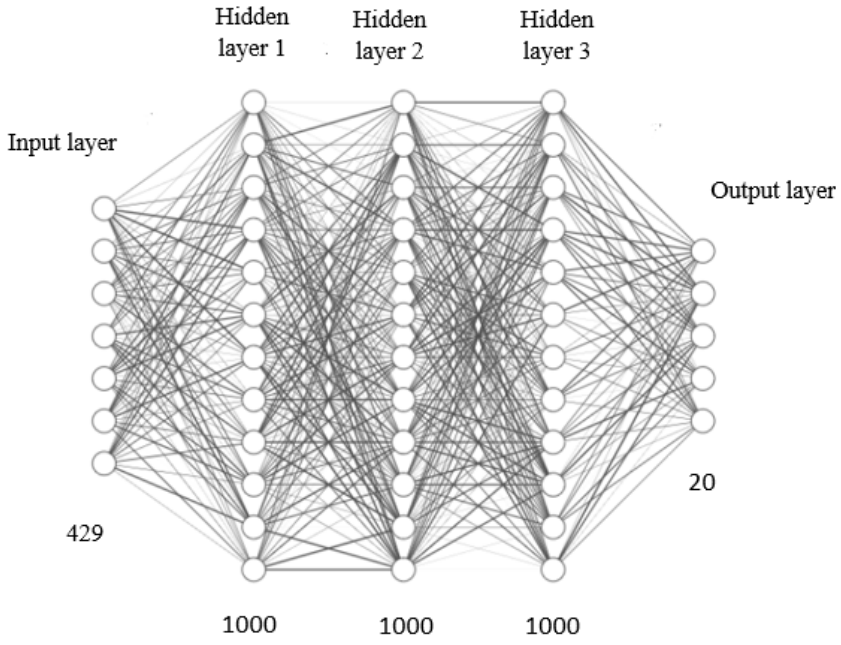


Figure 11. Baseline DNN architecture

Figure 11 shows the baseline DNN model used in our research. The baseline has three hidden layers, each with 1000 hidden units. The context-based training gave error-free performance for clean speech condition as expected and robust performance under acoustic background noise and handset variability, as described in the next section. This DNN configuration is our baseline system for the intended speaker recognition task and for subsequent complexity reduction task. This baseline DNN has 2.4M parameters.

4.2 Performance evaluation of the baseline DNN system

The performance of the baseline model was evaluated on clean speech database TIMIT, multi-handset database HTIMIT, and noise added TIMIT database.

The baseline DNN speaker recognition accuracies on TIMIT database in training and testing are both 100%. Training set here consisted of 8 utterances from each of 20 speakers, for a total of 160 utterances. Test set here consisted of 2 utterances from each of

20 speakers, for a total of 40 utterances. Each utterance is approximately 3 seconds long, which means approximately 150 frames.

Table 1. Performance of the baseline DNN model on HTIMIT Database		
Condition	Accuracy	Top-two
Training set	100%	-
Testing set	96.75%	99.5%

We used the HTIMIT database to evaluate the robustness of the baseline DNN model over handset variability. A higher error rate was found for female speakers because of the loss of high-frequency information due to the downsampling of TIMIT database and 300-3200 Hz band-limiting done to produce HTIMIT [22]. The error rate on female speakers was found to be 5.5% and the error rate on the male speakers was found to be only 1%. This is consistent with what has been reported previously where the GMM method was used [22].

In Table 1, "Top-two" refers to the performance metric where if the correct speaker identify corresponds to the top two averaged output posterior probabilities, the performance is declared as correct. The motivation for reporting top-two performance is that other context-based information, such as if the first-choice speaker is not "on duty" or the context of the ongoing dialog, can be used to correctly disambiguate between the top two choices. Top-two performance numbers are reported in the rest of this thesis whenever the standard (or "top-one") performance is less than 100%. The performance of the baseline DNN model on test data from individual handsets is given in Table 2. Training in these cases was performed over data from all ten handsets.

Table 2. Performance of the baseline DNN model on individual handset data

Handset	Accuracy	Top-Two
cb1	95%	97.5%
cb2	100%	-
cb3	95%	100%
cb4	90%	100%
el1	100%	-
el2	97.5%	100%
el3	97.5%	100%
el4	97.5%	100%
pt1	95%	97.5%
senh	100%	-

We achieved 100% performance for cb2, el1, and senh handset conditions. Recall that senh is a high-quality microphone. The handsets cb3, cb4, and pt1 produced lower performance as expected. Recall that pt1 is a cordless phone.

Table 3. Performance of the baseline model on noise added TIMIT

Condition	Accuracy		
	20 dB	10 dB	5 dB
Training set	100%	100%	100%
Testing set	100%	98.75%	98.125%
(Top-Two)	-	(100%)	(100%)

To study the robustness of the baseline DNN model in acoustic background noise, we evaluated the baseline system using the noise-added TIMIT database, which includes 4 noise types and 3 SNRs for each noise type. The performance of the baseline DNN model in noise at different SNRs is shown in Table 3. Figure 12 shows speaker recognition error rate at 3 different SNR conditions.

Training was performed over all 4 noise types at a given SNR. The baseline DNN model performs at 100% for 20 dB SNR condition and above 98% in 10 dB and 5 dB conditions. As mentioned above, the top-two performance means that the correct speaker is always in the top two choices in the DNN posterior probabilities. Under the top-two performance metric, the baseline DNN is flawless for all three SNRs, which is impressive.

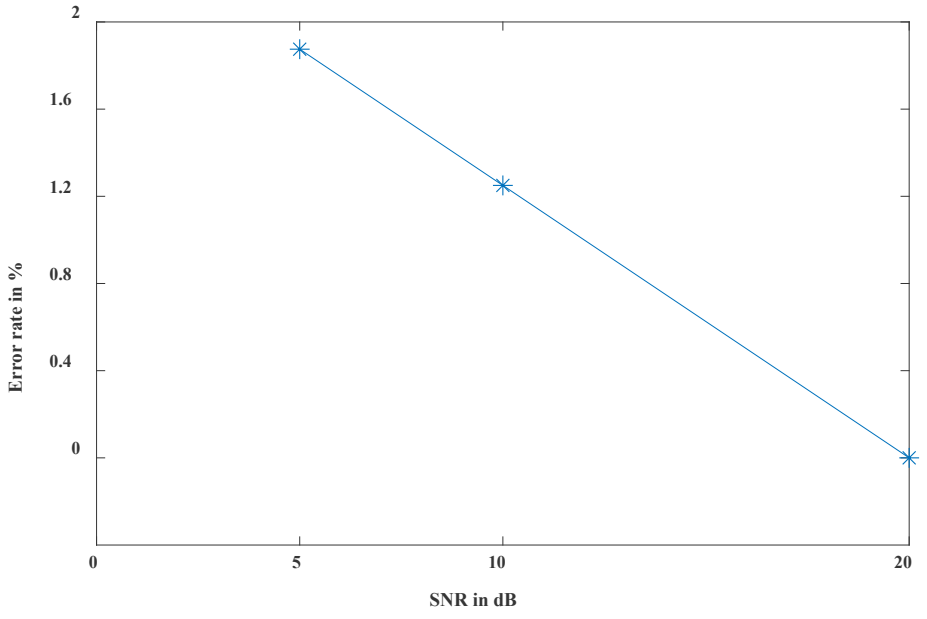


Figure 12. Performance of the baseline DNN on noise added TIMIT at different SNR conditions

V. INVESTIGATION OF COMPLEXITY REDUCTION TECHNIQUES AND THEIR PERFORMANCE

Reducing the complexity of the model is critical in small footprint DNN based speaker recognition system. When the available resources are limited it is possible to design a simplified DNN speaker recognition system using a suitable complexity reduction technique. Complexity reduction of the model can be achieved in several ways. This research is focused on reducing the number of parameters in the DNN model without significant loss in the accuracy. Pruning is a technique of removing the parameters from the network using a suitable algorithm. Pruning technique converts the fully connected dense network into a sparse network [34].

As noted earlier, this research uses the CNTK deep learning library to design the DNN. However, CNTK parameters can only be dense corresponding to fully connected hidden layers This is one of the drawbacks of using CNTK for pruning purpose. Therefore, we had to come up with a different approach to implement pruning using CNTK. Instead of removing the parameters from the network we set the less important parameters to zero and explicitly developed a python-based function in CNTK to avoid the zero weights from being updated while training the network.

5.1 Adaptive pruning

Pruning is a technique of removing parameters, namely weights, from the neural network without significant loss in accuracy. In a pruning technique, after training is completed, any weight that is less than a certain threshold is considered less important and is thus discarded or zeroed out. In the adaptive pruning technique we investigated, the standard deviation of each layer's weights multiplied by a quality factor is used as a

threshold, which makes the threshold data-adaptive rather than fixed. The weights below this adaptive threshold are removed from the network (or zeroed out as mentioned above) and the network is retrained. Failing to retrain will significantly impact the network's performance [34].

As noted above, the baseline model has 2.4M parameters. The parameters, namely weights, that fall below the adaptive threshold in each layer are set to zero and the final model with sustained (that is, non-zero) parameters is obtained. It is important to retrain the model starting with the sustained parameters rather than re-initializing the model weights. This makes the network train relatively quickly because of good starting conditions, thereby requiring less computation for retraining as well.

Table 4. Performance of adaptive pruning on TIMIT Data

Condition	Performance	Parameters left (Complexity reduction)
Training set	98.73%	254K (9.5X)
Testing set	95.23%	

We initially experimented on TIMIT database. The performance of the pruned model on training and test sets is as shown Table 4. We obtained 1.27% error on training set and 4.77% error on test set. The adaptive greedy pruning technique exhibited a high variation in the performance of the model. It is, therefore, important to closely watch the retraining process. We were able to reduce the number of parameters nearly 9.5X compared to the original baseline but with a significant drop in accuracy.

Table 5. Performance of adaptive pruning on HTIMIT		
Condition	Performance	Parameters left (Complexity reduction)
Training set	98.3%	215K (11X)
Testing set	85.5%	

To study the robustness of the pruned model in various handset conditions, we experimented on the HTIMIT database; the performance results are given in Table 5. The adaptive pruning produced a 1.7% error rate on the training set and 14.5% error rate on the test set, although with over 11X complexity reduction. With the adaptive pruning technique, it is harder to prune the network weights without sizeable performance loss, when the dataset has various handset conditions. In other words, when the database is not homogeneous, the effectiveness of the pruning approach becomes less.

We then experimented the pruning technique on noise added TIMIT database in different noise levels and different background noise types. The performance of the system in different noise conditions is shown in Figure 13.

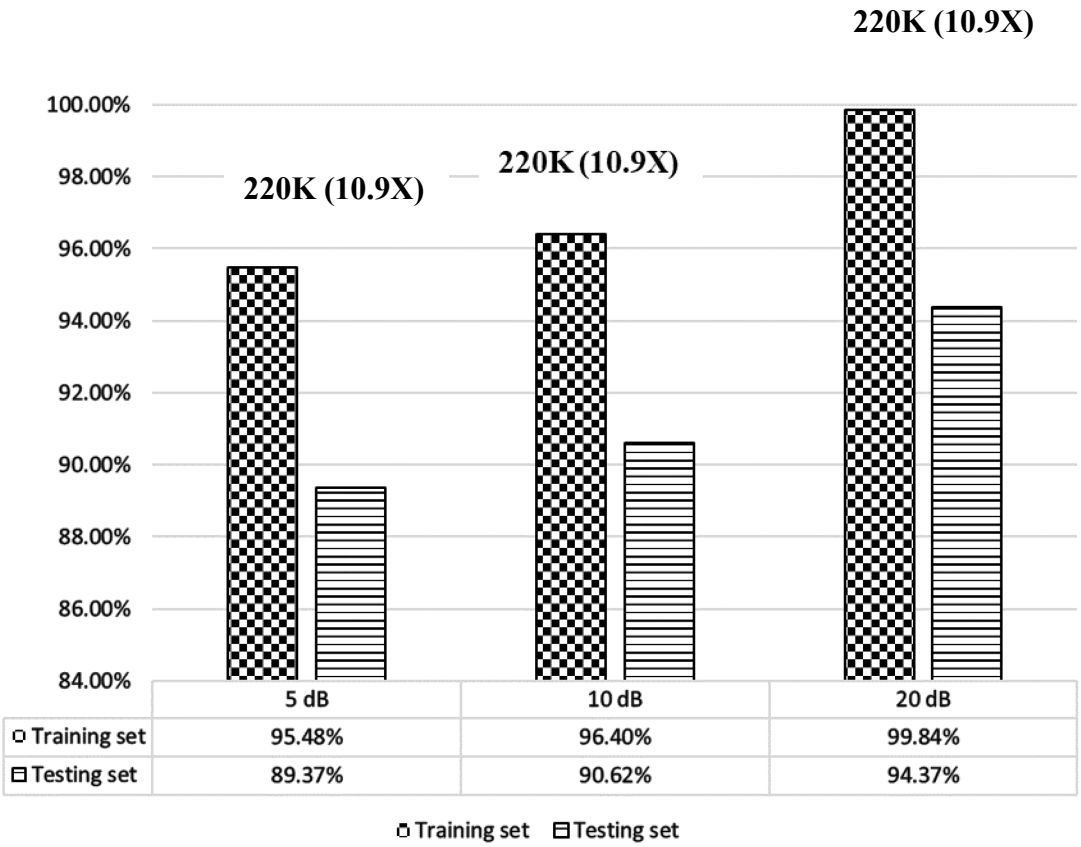


Figure 13. Performance of adaptive pruning on noise added TIMIT at different SNR conditions

This greedy pruning technique makes pruning less effective and exhibits high variation in the performance of the system under noise. We were able to achieve 10.9X complexity reduction but with a large performance loss.

We tried using L2 regularization to scale the updating of parameters while retraining to minimize the variations in the network but didn't get any significant improvement in the performance. The lack of effectiveness of pruning may, in part, be due to the fact that each noise condition involves 4 different noise types making the database non-homogeneous.

5.2 Sequential Layer Specific pruning

As noted earlier, in DNN the lower layers learn simple features and the higher layers learn more complex features. As stated in the literature review section, in indirect DNN based speaker recognition, DNN is used as a feature extractor. Usually, the hidden layers near the output layer are used for feature extraction because these layers contain speaker-specific information. The extracted features are then used with a secondary classifier to predict the speaker identity. We used this working principle of DNN to find the sequence or order in which the layers need to be pruned. This approach is based on the inter-dependency [46] of DNN layers. It is important to note that pruning here refers to zeroing out the parameters that fall below the threshold, as we are limited by the functionality of CNTK for pruning purpose. However, in cases of pruning supported software, one can use the sequential layer specific (SLS) pruning technique to remove the weights instead of making them zero. Figure 14 shows the baseline DNN architecture, with number of nodes of each layer shown.

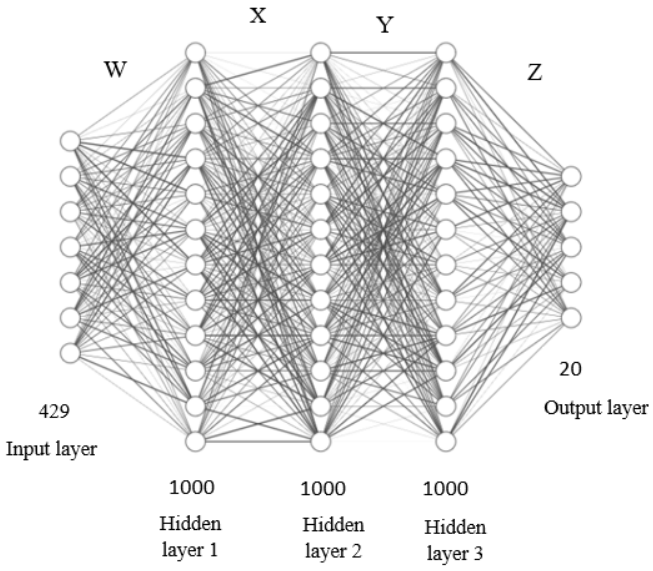


Figure 14. Our baseline Deep Neural Network architecture

The four sets of connections W, X, Y, and Z are sequentially dependent on each other: Y as a collection depends on X's output, X on W's output, and Z on Y's outputs. As before, we initially trained the fully connected DNN model. The standard deviation of each layer's weights is calculated. The threshold for each layer is calculated by multiplying the respective standard deviation with the quality factor, as before. We started pruning the final hidden layer weights Y; we chose this layer to prune first because this layer contains speaker-specific information and parameters that fall below the threshold are considered as not speaker-specific and removing them should not impact the performance of the system. However, as before, it is important to retrain the model after pruning the weights Y, as otherwise the performance of the model will be significantly impacted. The pruning technique is then applied to X followed by retraining; this procedure is then repeated for W; at the end, pruning is applied to Z. This entire pruning process is called Sequential Layer-Specific (SLS) pruning. The SLS pruning technique proposed here is not greedy; it is dependency-based pruning. As described, SLS pruning takes place in multiple stages in the specific sequence mentioned, with different number of parameters removed in each stage immediately followed by retraining.

To describe the SLS process further, while pruning Y weights, other layer weights are kept untouched; hence the retraining of parameters of weights Y is dependent on other layers and doesn't result in a significant drop in accuracy as compared to the earlier greedy pruning technique. As we are pruning one layer at a time in the right sequence as described above, the network is resistant to significant performance losses. We have found the SLS pruning techniques to outperform the earlier-described adaptive pruning

technique. The sequence of pruning we implemented is: start pruning from the last hidden layer and move towards the input layer and then finally prune the output layer if necessary. From experiments, we have found that aggressive pruning in the output layer can result in a large drop in accuracy. Hence, it is important to be cautious in pruning the output layer. The sequence of pruning a three hidden layer DNN is given as follows:

Y | Keeping X, W, Z constant

X | Keeping W, Y, Z constant

W| Keeping X, W, Z constant

Z | Keeping X, W, Y constant

The flow chart in Figure 15 illustrates the SLS pruning algorithm.

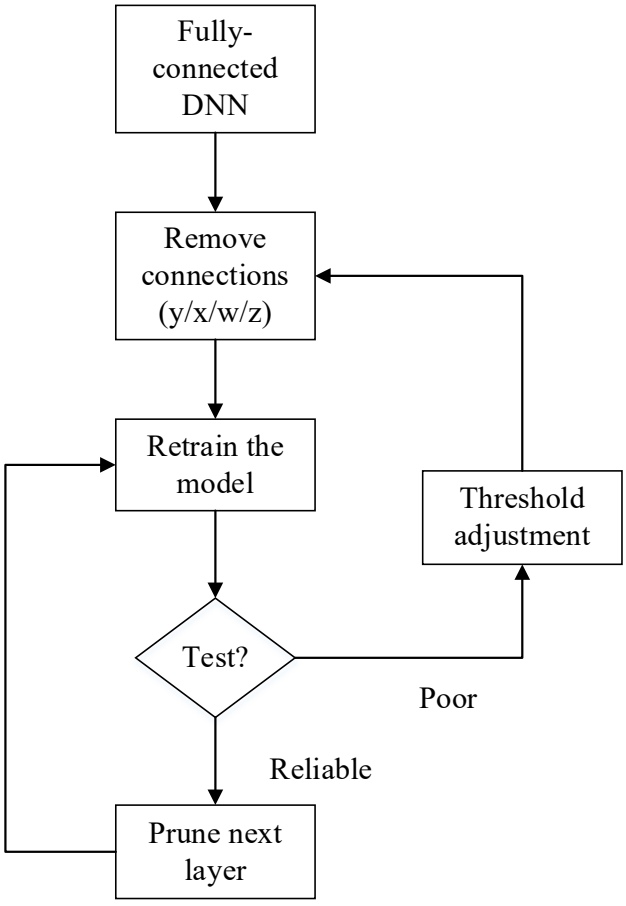


Figure 15. Flowchart of the SLS algorithm

The sequential layer specific pruning technique is a stable and efficient way of pruning the parameters in the neural network. This pruning technique makes the network resistant to sensitivity and performance loss. The proposed SLS technique involves zeroing out the less important parameters one layer at a time in a selective, sequential manner, followed by retaining the DNN model with sustained parameters at each stage.

The SLS pruning technique requires less computation compared to other pruning techniques because here the number of pruning and retraining processes is equal to the number of layers of the neural network. In contrast, in greedy pruning technique the network is made to learn the connections by greedily removing parameters in large numbers all at once.

We initially experimented on TIMIT database to examine the effects of SLS pruning on clean speech data. Table 6 shows the layer-wise performance of SLS pruning on TIMIT.

Table 6. Layer-wise performance of SLS pruning technique on TIMIT database

Connections	Train set	Test set	Parameters left in respective layer
Y	100%	100%	7.1K
X	100%	100%	11K
W	100%	100%	38.2K
Z	100%	100%	20K

The SLS pruned system has 100% accuracy for TIMIT database. The experimental result shows that we can prune relatively aggressively in the hidden layers

but not in the input and output layer. After multiple stages of layer-wise pruning, we were able to zero out approximately 2.373M parameters and the final pruned model has 79K parameters without any loss in accuracy. The SLS pruned system on TIMIT database achieved 100% accuracy. (The 79K count in the final pruned model includes 3K bias values and 20K output layer connections.) We didn't prune bias values as they help the model to fit the data better. We also didn't prune the final layer because we already zeroed out a maximum number of parameters in other layers and the final layer parameters were just 20K and pruning this layer wouldn't have contributed significantly to the overall complexity reduction rate. When we tried pruning the final layer, it yielded an error rate of 2%; this confirms our earlier position for not pruning the output layer connections.

To study the robustness of the SLS pruned DNN model we conducted experiments on multi-handset database HTIMIT. Let us recall that the HTIMIT database has 10 different handset conditions. The SLS pruning technique leads to a small loss in the performance due to the heterogeneous nature of the database. It is possible to carry out the layer-wise pruning if each layer pruning in a heterogeneous database exhibits a small error rate because every subsequent stage retraining will improve the overall performance of the network and yields a low complexity pruned model without significant loss in accuracy. The performance of the DNN model after pruning each layer and retraining the network is shown on Figure 16 , which also includes parameters left in respective layer (shown at the top of the bar graph). The performance the SLS pruned DNN is shown in Table 7, which also includes the top-two performance metric. The final

SLS pruned model has 538K parameters for a 4.5X reduction relative to the fully connected model and exhibits reliable performance in the speaker recognition task.

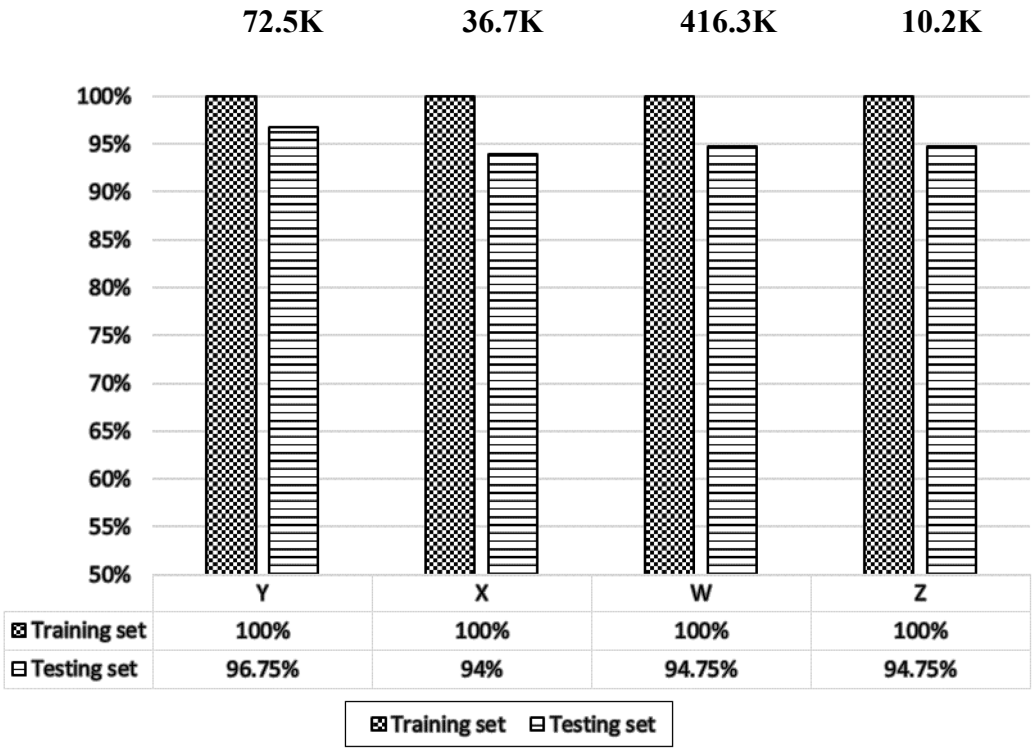


Figure 16. Layer-wise performance of SLS pruning technique on HTIMIT Database

Table 7. Overall performance of the final pruned model on HTIMIT database

Data	Accuracy	Top-two
Training set	100%	-
Testing set	94.75%	98.25%

The robustness of the SLS pruned model in noisy conditions is studied using the noise added TIMIT database. As discussed before, this database consists of 4 different noise types: car noise, white noise, wind noise, and city traffic noise. As shown earlier, it

is difficult to prune the model effectively in noisy conditions, using the earlier adaptive pruning technique. The SLS pruning technique is, however, more efficient to prune the parameters. The performance of the system for different noise conditions is shown in the following figures and tables.

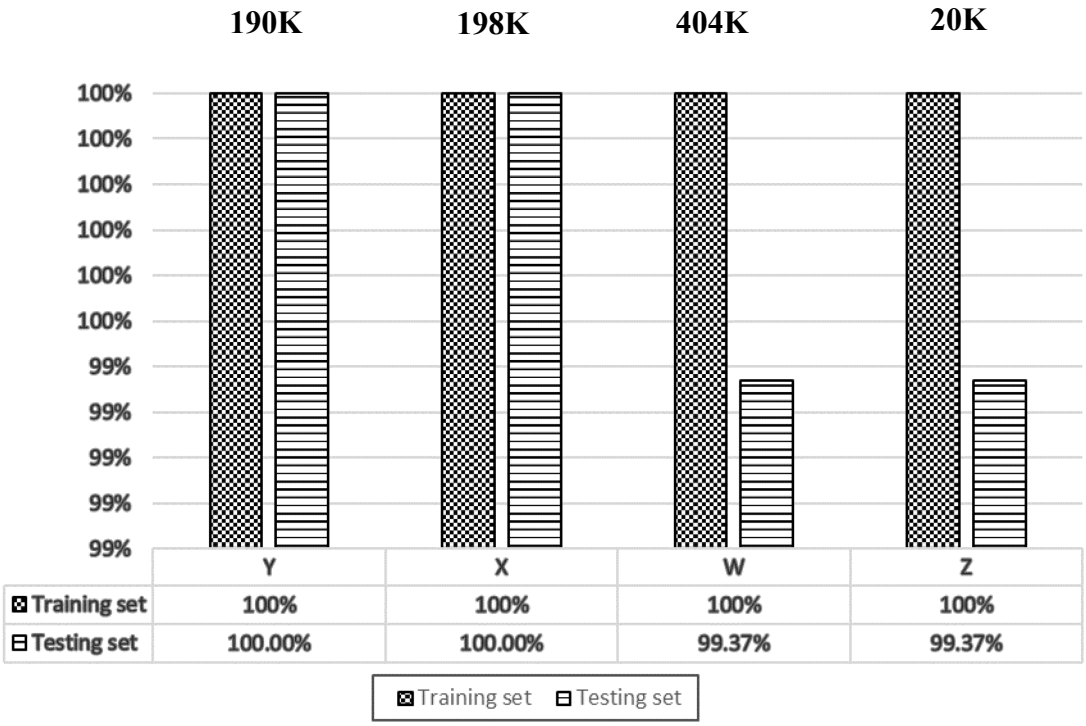


Figure 17. Layer-wise performance on noise added TIMIT database at 20 dB SNR

Table 8. Overall performance of the final SLS pruned model on noise added TIMIT at 20 dB SNR		
Condition	Accuracy	Top-two
Training set	100%	-
Testing set	99.37%	100%

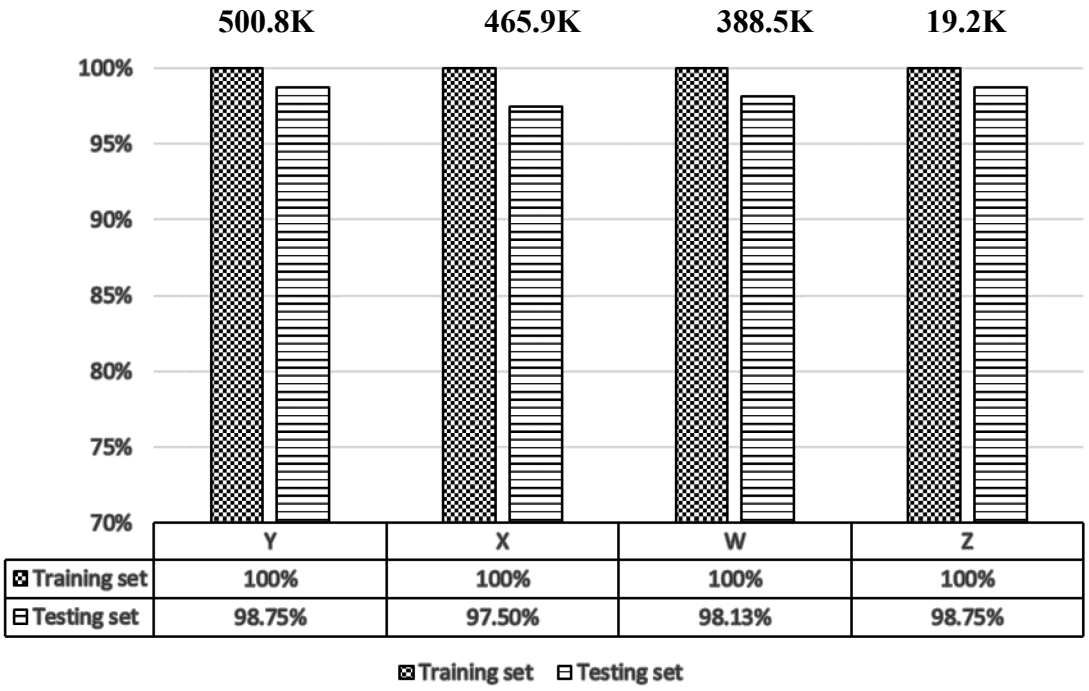


Figure 18. Layer-wise performance on noise added TIMIT database at 10 dB SNR

Table 9. Overall performance of the final SLS pruned model on noise added TIMIT at 10 dB SNR

Condition	Accuracy	Top-two
Training set	100%	-
Testing set	98.75%	100%

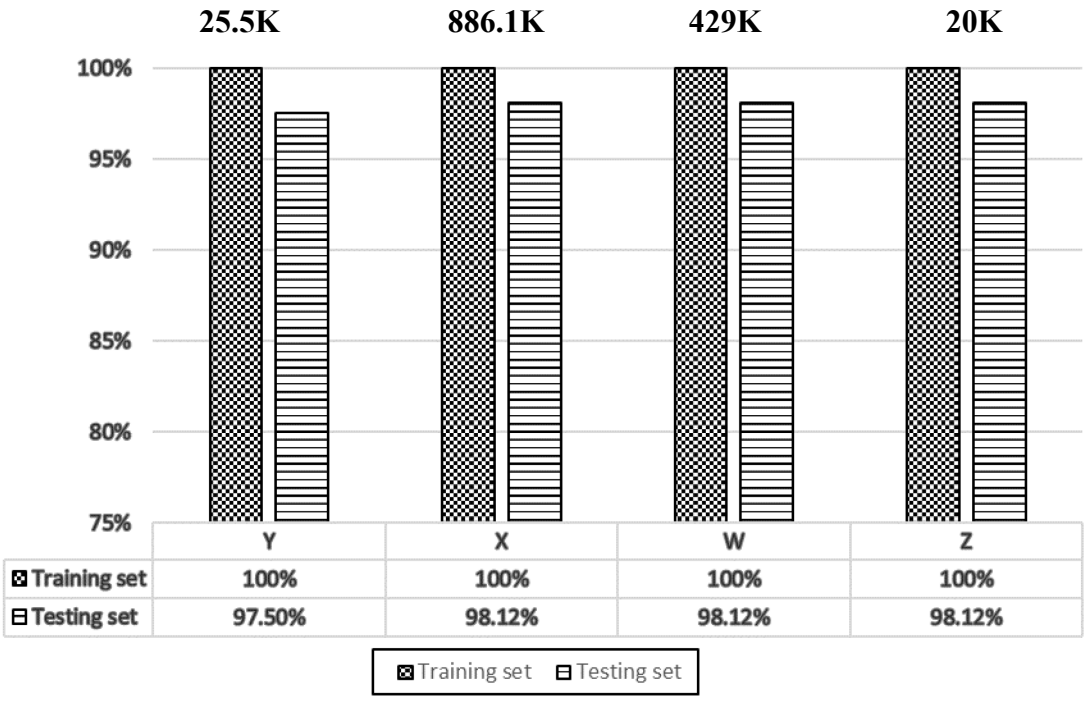


Figure 19. Layer-wise performance on noise added TIMIT database at 5 dB SNR

Table 10. Overall performance of the final SLS pruned model on noise added TIMIT at 5 dB SNR		
Condition	Accuracy	Top-two
Training set	100%	-
Testing set	98.12%	98.75%

The SLS pruning technique results in complexity reduction and speaker recognition accuracy for the 3 SNR conditions as follows: 3X, 99.37% for 20 dB SNR; 1.7X, 98.75% for 10 dB SNR; and 1.7X, 98.125% for 5 dB noise. It is interesting to note that in both 5 dB and 10 dB SNR conditions, the performance of the SLS pruned model is the *same* as that of the unpruned baseline, although at only a modest 1.7X complexity

reduction. For 20 dB SNR condition, a higher reduction of 3X is achieved but at a 0.625% performance loss relative to the baseline. The SLS technique can therefore be used to prune the model for noisy conditions without significantly compromising the performance of the system. Figure 20 shows the error rate of the SLS pruned model at different SNR conditions, complexity reduction rate for each case is also shown.

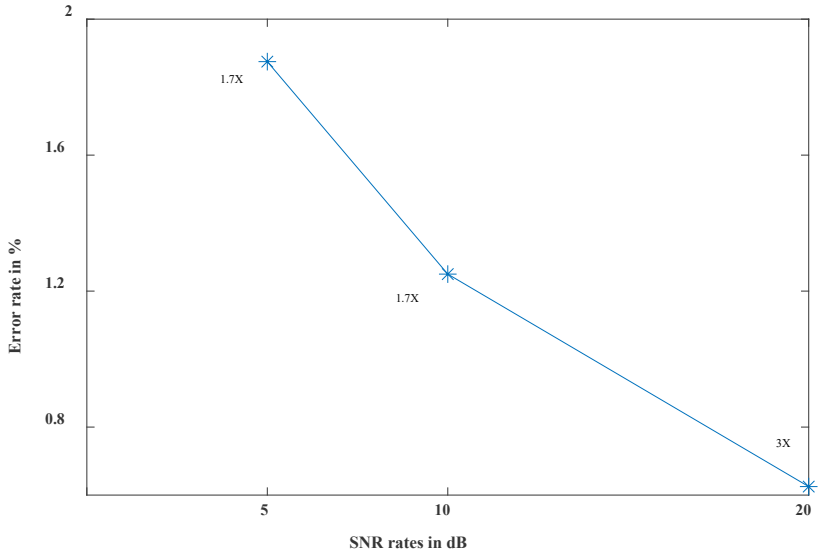


Figure 20. Performance of the SLS pruned model at different SNR conditions

VI. FINAL DNN BASED SMALL FOOTPRINT SPEAKER RECOGNITION SYSTEM

6.1 Details of the DNN baseline Model

We used direct DNN based speaker recognition. A 39-dimensional MFCC feature vector is extracted for each frame of the speech signal using the HTK toolkit. With context-window being 11 frames wide, the dimension of the input layer is 429 (=11x39). The output layer dimension is 20 as we are developing a speaker recognition system for 20 speakers. We used 3 hidden layers with 1000 hidden units per layer. In addition to this, the DNN baseline model has 3K bias parameters; hence the total number of parameters in our fully connected baseline model is about 2.4M (=429K+1M+1M+20K+3K). Figure 11 in Chapter IV shows the breakdown of parameters across layers. We used the ReLu activation function to the hidden layer units and softmax activation function to the output layer to get the output as a probability distribution over the 20 classes. Adam optimization algorithm is used with a learning rate of 0.001 for updating the network weights. A dropout rate of 30% and L2 regularization are used to effectively handle overfitting and hence to decrease the generalization error on unseen test data. This DNN baseline model is used for investigation of complexity reduction techniques.

6.2 Performance of the DNN baseline system

The direct DNN based baseline system is trained using the context-based training approach. The frame level DNN predictions from the output layer are averaged over the test utterance using a python script developed to predict the speaker identity. The performance of the baseline DNN system is flawless at 100% on training set data for

TIMIT, HTIMIT, and the three noised added TIMIT databases. The performance on test set is as follows: 100% accuracy on clean speech TIMIT database, 96.75% on multi-handset database HTIMIT, and 100%, 98.75%, and 98.125% on noise added TIMIT database at 20 dB, 10 dB, and 5 dB SNR, respectively. The top-two performance on multi-handset HTIMIT database is 99.5%. Under the top-two performance metric, the baseline DNN achieved 100% for both 10 dB and 5 dB SNR conditions, which is impressive; for 20 dB SNR, the performance (top-1) is already 100%.

6.3 Performance of the final pruned DNN system

The layer-specific standard deviations multiplied by layer-specific quality factors are used as thresholds to respective layer's connections for pruning purpose. We developed a custom python script to use CNTK for pruning purpose so as to allow the implementation of sparse connections. We used the Sequential Layer-Specific (SLS) pruning technique to prune the layers of the DNN baseline model in multiple stages in a selective, sequential manner. The pruned model in each stage is retrained to improve the performance of the model. We achieved a complexity reduction rate of 31X with no loss in accuracy for clean speech TIMIT database. For non-homogeneous database conditions like the multi-handset HTIMIT database, we achieved a complexity reduction rate of 4.5X with 2% drop in accuracy. For the non-homogeneous database of noise added TIMIT at each of 3 SNRs, each condition involving four different noise types, we achieved a complexity reduction rate of 1.7X with no loss in accuracy relative to the baseline DNN model for 10 dB and 5 dB SNR conditions. We were able to prune more parameters in 20 dB SNR condition and achieved a complexity reduction rate of 3X but with 0.625% accuracy drop. Under the top-two performance metric, the final pruned

DNN model achieved 98.25% accuracy for multi-handset HTIMIT database, and 100%, 100%, and 98.75% accuracy for noise added TIMIT database at 20 dB, 10 dB, and 5 dB SNR, respectively.

VII. CONCLUSIONS AND FUTURE WORK

In this research we have developed a reduced complexity direct DNN based small footprint text-independent closed-set speaker recognition system for 20 speakers. Consistent with the small footprint design goal, a baseline DNN model was developed with just enough layers and enough hidden units per layer, thereby reducing the total number of parameters, and by careful design to avoid the common problem of overfitting and to optimize algorithmic aspects including context-based training, activation functions, regularization, and learning rate. This baseline model was evaluated on two commercially available databases, clean speech TIMIT and multi-handset speech database HTIMIT, and on noise added TIMIT database that we created using four types of noises at three different signal-to-noise ratios (SNRs). The speaker recognition accuracy of the baseline is 100% for TIMIT, 96.75% for HTIMIT, and 100%, 98.75% and 98.125% for noise added TIMIT database at 20 dB, 10 dB and 5 dB SNR, respectively. This demonstrates that the baseline system has an error-free performance in relatively clean speech and a robust performance under telephone handset variability and in acoustic background noise. The baseline model has a total of 2.4M parameters.

We then developed a novel and enhanced pruning technique called Sequential Layer Specific (SLS) pruning. The SLS pruning technique performs pruning sequentially in multiple stages and in a layer-specific manner, followed by retraining after each pruning stage, while ensuring no or only minor performance loss in each pruning stage. For the SLS pruned model, the speaker recognition accuracy is 100% for TIMIT database with 31X complexity reduction; 94.75% for multi-handset database HTIMIT with 4.5X complexity reduction. For noise added TIMIT database, with 1.7X complexity reduction

there is no additional drop in speaker recognition accuracy relative to the baseline DNN in both 5 dB and 10 dB SNR and a 99.37% accuracy is achieved with 3X complexity reduction in 20 dB SNR. For cases where the speaker recognition accuracy is less than 100%, a higher "accuracy" is obtained using the "Top-two" performance metric in which recognition success is declared if the correct speaker lies in the top two choices predicted by the DNN model. The complexity reduced DNN gave comparatively similar results as the complex baseline DNN; hence, we have achieved the goal of developing a DNN with reduced complexity for small footprint applications.

The main contributions of this research are listed below:

- Optimized baseline DNN configuration is developed for a small footprint system.
- Baseline DNN provides error-free performance for clean speech and a robust performance under handset variability and acoustic background noise.
- Complexity of DNN is reduced using complexity reduction techniques.
- Methodology is developed to customize CNTK for implementing pruning of weights.
- Adaptive pruning is not as effective in non-homogeneous database conditions.
- A novel and effective pruning technique called Sequential Layer Specific pruning is developed exploiting layer-specific properties of DNN.
- SLS-pruned DNN system provides error-free performance for clean speech and a robust performance under handset variability and acoustic background noise.

Areas of future work are described next. First, a speech detector preprocessor may be used to identify and discard silence frames and use only speech frames for training and testing the DNN model, as silence frames do not carry speaker-specific information, and

therefore not using them may increase the effectiveness of the DNN model. Second, the acoustic background noise is one of the important operating conditions to be considered when using DNN for speaker recognition in real-world applications. A future research goal is to improve the performance of the developed DNN model in acoustic background noise by using a speech enhancement preprocessor. Third, techniques like cepstral mean subtraction preprocessor may be used to improve performance under handset variability. Fourth, for embedded applications including the one that may be used in the International Space Station, the total number of parameters used in the DNN model is an important factor as it determines required storage, memory bandwidth, and computational resources. The associated energy cost can be estimated for a given processor architecture using the approach discussed in [34].

REFERENCES

[1] G. Salazar, A. Romero and D. Juge, Personal communication, 2016, Human Computer Interface, Avionics Systems Division, NASA Johnson Space Center,Houston,TX.

[2] B. Lei Jimmy and R. Caruana, "Do Deep Nets really need to be Deep?," in *NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing systems*, 2014.

[3] G. Fant, "Phonetics and Phonlogy in the last 50 years," in *Dept. of Speech, Music and Hearing*, KTH,Sweden, 2004.

[4] D. Povey, A. Ghoshal, G. Boulianne, N. Goel, M. Hannemann, Y. Qian, P. Schwarz and G. Stemmer, "The Kaldi Speech Recognition Toolkit," in *IEEE Signal Processing Society, Hilton Waikoloa Village, Big Island*, Hawaii,US, 2011.

[5] D. Yu, A. Eversole, M. L. Seltzser, K. Yao and B. G. Huang, An Introduction to Computational Networks and the Computational Network Toolkit, Microsoft Research,, 2015.

[6] E. J. Bradley, K. Panagiotis, A. Zeynettin, K. Timothy and K. Philbrick, "Toolkits and Libraries for Deep Learning," in *Digit Imaging*, 2017.

[7] D. A. Reynolds and R. C. Rose, "Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models," in *IEEE Transactions on speech and Audio Processing*, 1995.

[8] D. Garcia-Romero, X. Zhang and A. McCree, "Improving Speaker Recognition Performance in the Domain Adaptation using Deep Neural Networks," in *IEEE Spoken Language Technology Workshop*, South Lake Tahoe, NV, USA, 2014.

[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohammed, N. Jaitly, A. Senior and V. Vanhoucke, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," in *IEEE Signal Processing Magazine*, 2012.

[10] J. P. Campbell, "Speaker Recognition: A Tutorial," in *Proceedings of the IEEE*, 1997.

[11] D. Reynolds, "Speaker Identification and Verification using Gaussian mixture speaker models," in *Speech Communications*, 1995.

[12] F. Richardson, D. Reynolds and N. Dehak, "Deep Neural Network Approaches to Speaker and Language Recognition," in *IEEE SIGNAL PROCESSING LETTERS*, 2015.

[13] E. Variani, X. Lei, E. McDermott, I. L. Moreno and J. Gonzalez-Dominguez, "Deep Neural Network For Small Footprint Text-Dependent Speaker Verification," in *IEEE International Conference on Acoustic, Speech and Signal Processing*, 2014.

[14] D. Kriesel, A Brief Introduction to Neural Networks, https://www.dkriesel.com, 2007.

[15] G. Jiuxiang, Z. Wang, J. Kuen and L. Ma, "Recent Advances in Convolutional Neural Networks," in *arXiv.org*, 2015.

[16] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic Optimization," in *International Conference for Learning Representations*, San Diego, 2015.

[17] M. A. Nielsen, Neural Network and Deep Learning, Determination Press, 2015.

[18] A. V. Sharma, Understanding Activation Functions in Neural Networks, https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

[19] W. Brett R, K. K. Paliwal and B. Wildermoth, "GMM Based Speaker Recognition on Readily Available Databases," in *Microelectronics Engineering Research Conference*, 2003.

[20] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus and D. S. Pallett, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," Linguistic Data Consortium, 1993.

[21] D. Reynolds, "HTIMIT," Linguistic Data Consortium, 1998.

[22] D. A. Reynolds, "HTIMIT and LLHDB: Speech Corpora For The Study Of Handset Transducer Effects," in *International Conference on Acoustics, Speech, and Signal Processing*, 1997.

[23] S. Han, H. Mao and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in *ICLR*, 2016.

[24] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: from features to supervectors," *Elsevier Science Publishers,* vol. 52, no. 1, pp. 12-40, 2010.

[25] D. A. Reynolds, T. F. Quatieri and R. B. Dunn, "Speaker Verification using Adapted Gaussian Mixture Models," in *Digital Signal Processing 10*, 2000.

[26] M. K. Omar and J. Pelecanos, "Training Universal Background Models for Speaker Recognition," in *The Speaker and Language Recognition Workshop*, Czech Republic, 2010.

[27] G. Nijhawan and D. M. K. Soni, "Speech Recognition using MFCC and Vector Quantisation," *International Journal on Recent Trends in Engineering and Technology,* vol. 11, no. 1, 2014.

[28] Ondrej Novotny, O. Plchot, P. Matejka, L. Mosner and O. Glembek, "On the use of X-vectors for Robust Speaker Recognition," in *The Speaker and Language Recognition Workshop*, France, 2018.

[29] A. Zulfiqar, A. Muhammad, A. M. Martinez-Enriquez and G. Escalada-Imaz, "Text-Independent Speaker Identification Using VQ-HMM Model Based Multiple Classifier System," in *Mexican International Conference on Artificial Intelligence*, 2010.

[30] H. Lee, P. Pham, Y. Largman and A. Y. Ng, "Unsupervised feature learning for audio classification using convolution deep belief networks," in *Advances in Neural Information Processing Systems 22*, 2009.

[31] M. Denll, B. Shaklbl, L. Dlnh, M. A. Ranzato and N. d. Freitas, "Predicting Parameters in Deep Learning," in *ARXIV*, 2013.

[32] H. Wang, Q. Zhang, Y. Wang and R. Hu, "Structured Deep Neural Network Pruning by Varying Regularization Parameters," in *ARXIV*, 2018.

[33] L. Theis, I. Korshunova, A. Tejani and F. Huszar, "Faster Gaze Prediction With Dense Networks and Fisher Pruning," in *ARXIV*, 2018.

[34] S. Han, J. Pool, J. Tran and W. J. Dally, "Learning both Weights and Connections for Efficient Neural Networks," in *ARXIV*, 2015.

[35] Y. LeCun, J. S. Denker and S. A. Solla, "Optimal Brain Damage," in *Advances in Neural Information Processing Systems 2*, 1989.

[36] L. Prechelt, “Early stopping- But When?,” in Neural Networks:Tricks of the Trade, Springer,Berlin,Heidelberg, 2012.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Network from Overfitting," *Journal of Machine Learning Research 15,* pp. 1929-1958, 2014.

[38] D. H. park, C. M. H0 and Y. Chang, "Achieving Strong Regularization for Deep Neural Networks," in *ICLR*, 2018.

[39] K. Simonyan and A. Zisserman, "Very Deep Convolution Networks For Large-Scale Image Recognition," in *ICLR*, 2015.

[40] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM,* vol. 60, no. 6, pp. 84-90, 2017.

[41] G. Cheng, P. Vijayaditya, D. Povey, V. Manohar, S. Khudanpur and Y. Yan, "An exploration of dropout with LSTMs," in *Interspeech*, 2017.

[42] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell and D. Ollason, The HTK, 2001.

[43] https://www.microsoft.com/en-us/research/blog/microsoft-computational-network-toolkit-offers-most-efficient-distributed-deep-learning-computational-performance/.

[44] C. Bagwell, "Sound eXchange," SoX Contributors.

[45] https://www.freesoundeffects.com/free-sounds/ambience-10005/.

[46] https://ml4a.github.io/ml4a/how_neural_networks_are_trained/.

[47] S. Rydin, "Text dependent and text independent speaker verification systems. Technology and applications," in *Centre for Speech Technology*, KTH,Stockholm, 2001.

[48] X. Zhao, Y. Wang and D. Wang, "Deep Neural Networks For Cochannel Speaker Identification," in *IEEE International Conference on Acoustic and Signal Processing*, 2015.

[49] https://pythonmachinelearning.pro/a-guide-to-improving-deep-learnings-performance/.

[50] T. Kaddoura, K. Vadlamudi, S. Kumar, P. Bobhate, L. Guo, S. Jain and M. Elgendi, "Acoustic diagnosis of pulmonary hypertension:automated speech-recognition-inspired classification algorithm outperforms physicians," *scientific reports,* 2016.