TEXT INDEPENDENT SPEAKER VERIFICATION USING NEURAL NETWORKS

by

Sedat Demirbağ

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2019

ACKNOWLEDGEMENTS

First of all, I would like to thank to my thesis advisor Levent Arslan for his guidance and kind support during my study.

I am thankful to Murat Saraçlar and Engin Erzin for being in my thesis committee by devoting their time and energy.

I would like to thank to my company, Sestek, for providing the physical, computational environment and sharing the databases for this study. I thank to my colleagues there for their help in every aspect, especially Mustafa Erden for his support in forming theoretical and technical basis during this study.

For their lovely support and encouragement at all stages of this study, I would like to thank to my wife, Cansu, and my family. Their accompany has been a source of motivation.

ABSTRACT

TEXT INDEPENDENT SPEAKER VERIFICATION USING NEURAL NETWORKS

Speaker verification is the process of proving one's identity through voice. A speaker verification mechanism can be deployed in a banking system, a call center, a forensics software or a mobile application for accessing sensitive data. Text independent speaker verification deals with cases with unrestricted spoken content. The change in phonetic content and duration of the speech is a challenge in this field.

In this study, we propose a method for text independent speaker verification task. In the proposed method, phonemes of two utterances are recognized together with their boundaries. All acoustic spectral features are extracted and fused with recognized phonetic information. All phonemes are paired with the same ones from another utterance, and fused phonetic information are concatenated. This information is given to a feedforward fully connected neural network for the same/different speaker decision.

In training and tests, public English speaker verification datasets are used. Effects of various feature, data and test conditions are investigated. Equal error rates ranging from 16.7% to 23.7% are observed for recordings with changing speech duration between 15 seconds and 1 second. When the whole recording is used 15.5% error is observed.

ÖZET

SİNİR AĞLARI KULLANARAK METİN BAĞIMSIZ KONUŞMACI DOĞRULAMA

Konuşmacı doğrulama, bir kişinin kimliğinin sesi vasıtasıyla kanıtlanması işlemidir. Bir konuşmacı doğrulama mekanizması, hassas verilerin erişimi için bir banka sistemine, bir çağrı merkezine, bir adli yazılıma ya da bir mobil uygulamaya kurulabilir. Metin bağımsız konuşmacı doğrulama, konuşma içeriğinin sınırlandırılmadığı durumları kapsamaktadır. Fonetik içerik ve konuşma süresindeki değişim bu alandaki zorlu faktörlerdendir.

Bu çalışmada, metin bağımsız konuşmacı doğrulama görevi için bir yöntem önermekteyiz. Önerdiğimiz yöntemde, iki konuşmaya ait fonemler ve bu fonemlerin sınırları tanınarak belirlenir. Tüm akustik spektral öznitelikler üretilir ve tanınan fonetik bilgi ile birleştirilir. Tüm fonemler başka bir konuşmadan aynılarıyla birebir eşleştirilir ve sahip oldukları öznitelikler yine bir araya getirilir. Elde edilen bilgi ise ileri beslemeli ve tam bağlı bir yapay sinir ağına aynı/farklı konuşmacı kararı vermesi için gönderilir.

Uygulanan sistemin eğitim ve testlerinde, açık erişimli İngilizce konuşmacı doğrulama veritabanları kullanılmıştır. Çeşitli öznitelik, veri ve test koşullarının etkileri incelenmiştir. Eşit hata oranı bazında, içerdiği konuşma uzunluğu 15 saniyeden 1 saniyeye kadar değişen kayıtlar için, %16,7'den %23,7'e kadar hata gözlenmiştir. Kayıtların tamamının kullandığı durumda ise %15,5 eşit hata oranı elde edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

b_k	Bias term in k^{th} neuron in neuron model
C	Context dependency WFST in ASR decoding
$c[n]$	Cepstrum of the signal in MFCC
$\boldsymbol{D}$	Speaker variability matrix in JFA
G	Grammar WFST in ASR decoding
H	HMM WFST in ASR decoding
H_0	Hypothesis of speech being from target speaker
H_1	Hypothesis of speech being from non-target speaker
L	Lexicon WFST in ASR decoding
$\boldsymbol{M}$	Speaker and channel dependent supervector in JFA and i-vector
$\boldsymbol{m}$	Speaker and channel independent supervector in JFA and i-vector
$\boldsymbol{T}$	Total variability matrix in i-vector
t_i	i^{th} class probability in cross-entropy cost function
$\boldsymbol{U}$	Channel variability matrix in JFA
$\boldsymbol{V}$	Speaker variability matrix in JFA
w_i	Weight of i^{th} Gaussian in GMM
w_kj	Connection weight from j^{th} to k^{th} neuron in neuron model
x_j	Input value coming from j^{th} neuron in neuron model
$x[n]$	Speech signal in MFCC
$\tilde{y}_i$	Labeled value of i^{th} output neuron in mean squared error cost function
y_k	Output value of k^{th} neuron in neuron model
ϵ	Learning rate in gradient descent
θ	Likelihood ratio test threshold
$\boldsymbol{\mu}$	Mean vector
$\boldsymbol{\Sigma}$	Covariance matrix

σ	Standard deviation
ϕ	Activation function in neuron model

LIST OF ACRONYMS/ABBREVIATIONS

ANN	Artificial Neural Networks
API	Application Programming Interface
ASR	Automatic Speech Recognition
CMN	Cepstral Mean Normalization
CMS	Cepstral Mean Subtraction
CMVN	Cepstral Mean and Variance Normalization
CNN	Convolutional Neural Network
CNTK	Computational Network Toolkit
DET	Detection Error Tradeoff
DNN	Deep Neural Network
DTFT	Discrete Time Fourier Transform
EER	Equal Error Rate
EFR	Eigen Factor Radial
EM	Expectation Maximization
FAR	False Acceptance Rate
FRR	False Rejection Rate
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IDTFT	Inverse Discrete Time Fourier Transform
IVR	Interactive Voice Response
JFA	Joint Factor Analysis
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
MFCC	Mel-Frequency Cepstrum Coefficients
NAP	Nuisance Attribute Projection
NIST	National Institute of Standards and Technology
PCA	Principal Component Analysis
PLDA	Probabilistic Linear Discriminant Analysis

RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SRE	Speaker Recognition Evaluation
SVM	Support Vector Machines
T-norm	Test Normalization
TDNN	Time Delay Deep Neural Network
UBM	Universal Background Model
WCCN	Within-Class Covariance Normalization
WFST	Weighted Finite State Transducers
Z-norm	Zero Normalization

1. INTRODUCTION

Speaker verification is a means of identity authentication through voice. It is one of the popular biometric verification approaches.

First of all, why is there a need for a verification system at all? Human abilities are increasing with advancements in technology. These advancements mostly save time, effort and cost. They bring communications, transactions to a digital environment. One does not have to be physically present for some operation to take place. In such a case, because some human inspection would not be efficient and possible, there should be an automatic verification system integrated with the required mechanisms.

For many years, passwords are used for the verification purposes. They are easy to generate. One may have many passwords for many systems. However, they need to be memorized. Also, they may be easy to break by someone else. In this case, human biometry offers a good alternative to password authenticated systems. Everyone carries in his/her body some information unique to himself/herself. Some of this information can be extracted and used for verification of his/her identity. Biometric information can be extracted from retina, iris, fingerprint, palm, face, voice etc. [3]. If an adequate level of accuracy is reached, biometric verification will be a more reliable solution. There are spoofing threats for biometric systems too. However, anti-spoofing measures are being developed, which can decrease the vulnerabilities of these systems [4].

Applications using speaker verification operate mainly on call centers, banking operations, forensics or mobile devices etc. Speaker verification may be located as an additional security step for a sensitive operation such as a banking transaction. Such sensitive actions require identity verification with a high importance, and voice is used for this verification purpose. It is not used as the only means for the verification of the identity, but used as an additional verification step, or as replacement to an inefficient verification step. For the forensics cases, speaker verification stands as a helpful instrument. If evidences of a crime may exist in voice form, and require a

quantitative approval, then speaker verification comes into play. A speaker verification system, may be also deployed in an interactive voice response (IVR) environment. It may ease the verification process during a conversation, without customer even noticing it. Also, it may serve as a simple content locking tool for any mobile application. Access to the chosen content may be given for only specified, voice-enrolled users. Moreover, a user configuration may be activated on a speech-driven device. Home assistants are such devices, and they can activate preferences of a user by recognizing him/her through voice.

A generic speaker verification system consists of the following phases. In a training phase, the system is formed, system-wide models and parameters are obtained. Then, a speaker uses the system, and speaks some utterance for creating a voiceprint in first pass. Another time, when user wants to authenticate his/her identity he/she speaks again some utterance and system accepts or rejects his/her identity. This process is depicted in Fig. 1.1.

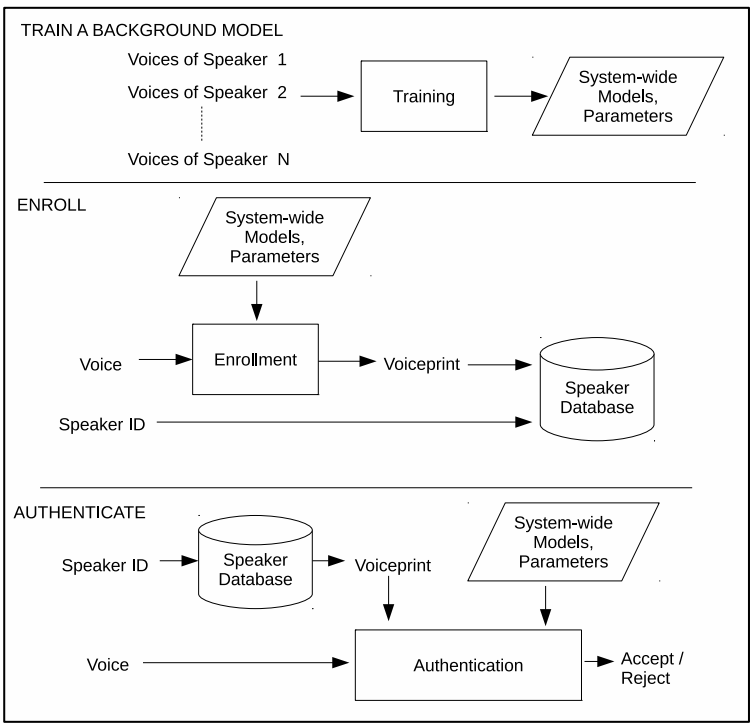

Figure 1.1. General speaker verification scheme.

There may be some challenging factors in the nature of verification process. A mismatch may occur between channels of enrollment and authentication. User may have been enrolled using a mobile phone, and may be trying to authenticate using a landline phone. Even, mismatch may occur between telephone microphone and a lapel microphone. They would have different channel and recording characteristics, thus result in a mismatch. Also, the speaker may experience changes in his/her voice due to sickness, mood change, emotional state etc. Background noises may be another challenging factor. If recording is not done in a controlled environment, many types of noises may be expected. Hence, a performance degradation may be seen in verification systems. All possible challenges are detailed under speaker based, conversation based and technology based variability sources in [5].

Some challenges may be present in application side of the verification process, such as training with limited data or targeting a different language. An additional challenge might be working with very short duration recordings.

Speaker verifications systems can be divided into two main categories: Text dependent and text independent. In text dependent case, spoken content, called passphrase, is determined beforehand. User speaks the same passphrase in both voiceprint creation (i.e., enrollment) and authentication. In text independent case, however, spoken content is not limited to any sentence. User may speak anything, on any topic. Other than content of the speech, duration of the speech is mostly another major difference of these categories in practice. In text dependent verification, passphrase is expected to have a short duration. Whereas, in text independent verification, speech may have longer durations and can be extracted even from a conversation between a customer and agent. Of course, a successful application working even with shorter durations of text independent verification recordings would be desired.

In this study, an approach is presented for text independent speaker verification task. With this approach, phonetic information is fused with spectral features in a single artificial neural network (ANN) architecture for verification phases. Details for the ANN will be given in chapters 2 and 4.

The main contributions of this thesis are i) the development of a text independent speaker verification system with phonetic information, ii) introduction of a feature fusion technique based on phonetic boundaries, iii) analysis of speaker characteristic representation among different phonemes.

Outline of the thesis is as follows. Previous works about speaker verification are given in chapter 2. A theoretical basis is formed in chapter 3. In chapter 4, the proposed methodology is described. Then, details about the dataset and experiments are presented in chapter 5. Lastly, conclusions are made about the study in chapter 6.

2. PREVIOUS WORK

In this chapter, previous studies about speaker verification are inspected with their overall working mechanisms.

For the speaker verification task, there have been various approaches in literature. The recent ones may be grouped under the following sections.

2.1. GMM-UBM

Gaussian mixture model - universal background model (GMM-UBM) systems had been quite popular for a long time. In that framework, there is a universal model representing the mixture of many speakers. Also for every known speaker, there is a model adapted from the universal one. For a hypothesis, new recording is compared against both generic model and speaker model, and a decision is made with comparison of their likelihood ratio to a tuned threshold [6].

Given a speech signal, Y, a hypothesis speaker, S, and a threshold, θ, the likelihood ratio test can be expressed as in Equation 2.1.

$$\frac{p(Y|H_0)}{p(Y|H_1)} \gtrless \theta \tag{2.1}$$

where,

$$H_0 = \text{Speech is from speaker } S.$$

$$H_1 = \text{Speech is not from speaker } S.$$

Likelihood of hypothesis H_0 is obtained from speaker specific model, whereas likelihood of hypothesis H_1 is obtained from the universal model.

The speaker model and universal model are both represented by GMMs. A GMM is formed with a collection of weighted Gaussians. The mixture density can be expressed as in Equation 2.2 and 2.3 given weight w_i and density $p_i(x)$ for i^{th} Gaussian.

$$p(x|\lambda) = \sum_{i=1}^{M} w_i p_i(x) \tag{2.2}$$

$$\sum_{i=1}^{M} w_i = 1 \tag{2.3}$$

A single Gaussian density can be further expressed as follows:

$$p_i(x) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma_i}|^{1/2}} \exp\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_i})'(\boldsymbol{\Sigma_i})^{-1}(\boldsymbol{x} - \boldsymbol{\mu_i})\} \tag{2.4}$$

where, D is the size of feature vector $\boldsymbol{x}$, $\boldsymbol{\mu_i}$ is the mean vector, and $\boldsymbol{\Sigma_i}$ is the covariance matrix of the i^{th} Gaussian.

As for generation of the models, the UBM is trained using expectation maximization (EM) algorithm [7] on a large dataset. Then, speaker models are trained via the adaptation of the UBM on speaker data using a modified EM approach. In this approach, the E step is the same as default EM algorithm, but in M step, UBM statistics are mixed with speaker statistics [6].

UBM generation may be done also on subpopulations created by gender, channel etc. groups. It may be trained, for instance, for males and females separately, then merged into a single UBM. This way, imbalances in training data amounts can be compensated [6].

Normalization is also an important factor for speaker verification. Due to mismatches of datasets, a normalization may be required. Various score and feature normalization approaches are developed and applied in literature [8–12]. Zero normal-

ization (z-norm) and test normalization (t-norm) are examples of score normalization techniques. In both approaches, the score is normalized using a mean and standard deviation value as in Equation 2.5, with S, S_{norm}, μ_I, and σ_I representing the score, normalized score, impostor mean, and impostor standard deviation, respectively. In the case of z-norm, mean and standard deviation are obtained using speaker model - impostor voice comparison scores. Z-norm statistics can be obtained in training time. In t-norm case, statistics are obtained using test voice - impostor model comparison scores. T-norm statistics can be obtained in testing time and require an impostor model set to be present aside [8].

$$S_{norm} = \frac{S - \mu_I}{\sigma_I} \tag{2.5}$$

In [9], several feature and score normalization techniques are experimented on a GMM basis. As feature normalization, they evaluated cepstral mean subtraction (CMS), variance normalization and feature warping. As for score normalization they applied t-norm, z-norm and cohort methods. Totally, they obtained best results with combination of feature warping together with t-norm.

2.2. I-vector

An advancement is reached with the development of i-vector framework. In this structure, speakers are represented with low dimensional vectors, called i-vectors.

Prior to i-vectors, joint factor analysis (JFA) method [13] represented speakers with fixed size supervectors, and described channel and speaker variabilities as given in Equation 2.6. Here, M and m represent the speaker & channel dependent and independent supervectors, respectively. D and V represent speaker variability matrices, and U represents the channel variability matrix. The vectors y, x and z are the corresponding factors [14].

$$\boldsymbol{M} = \boldsymbol{m} + \boldsymbol{V}\boldsymbol{y} + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{z} \tag{2.6}$$

I-vector approach unified the variability subspaces of JFA into a single one, called total variability space. Equation 2.7 shows the i-vector's main definition. Here, again, M and m represent the speaker & channel dependent and independent supervectors, respectively. T represents the total variability space, consisting of speaker and channel variability together. w is the corresponding factor, which is called as i-vector [14].

$$\boldsymbol{M} = \boldsymbol{m} + \boldsymbol{T}\boldsymbol{w} \tag{2.7}$$

As for speaker similarity calculation, various approaches are studied such as support vector machines (SVM) or cosine similarities of i-vectors, or using a separately trained probabilistic linear discriminant analysis (PLDA) model variants [14–18]. SVM is based on finding a linear separator on the surface. It requires a supervised training process. In case of cosine similarity, i-vectors are directly used in a cosine distance measure, and this distance is compared to a determined threshold as in Equation 2.8, where w_{target} and w_{test} represent i-vectors to be compared, and θ represents the threshold. Usage of cosine similarity reduces the complexity and processing times [14].

$$score(\boldsymbol{w_{target}}, \boldsymbol{w_{test}}) = \frac{\langle \boldsymbol{w_{target}}, \boldsymbol{w_{test}} \rangle}{\|\boldsymbol{w_{target}}\| \|\boldsymbol{w_{test}}\|} \gtrless \theta \tag{2.8}$$

For compensation of channel variabilities, several approaches are adopted. Within-Class Covariance Normalization (WCCN), Linear Discriminant Analysis (LDA), Nuisance Attribute Projection (NAP) and Eigen Factor Radial (EFR) are channel compensation techniques used [12, 14, 19, 20]. The experiments in [14] compares WCCN, LDA, NAP approaches and show the best performance in LDA and WCCN combination. In [19] and [20], EFR is compared against WCCN, LDA combination, and only WCCN cases. In the former, EFR outperformed the WCCN, LDA combination on NIST speaker recognition evaluation (SRE) database, and in the latter EFR and WCCN showed close performance on Switchboard and RSR2015 [21] databases.

In [18], i-vectors are simply normalized by their lengths. The authors showed the benefits of i-vector length normalization by improving the performance of Gaussian PLDA system up to computationally expensive Heavy-Tailed PLDA system.

2.3. ASR DNN in I-vector Backend

There has been attempts for incorporating phonetic information into verification process. Some used automatic speech recognition (ASR) deep neural networks (DNN) as i-vector backend.

In [22], UBM is replaced with an ASR DNN to generate posteriors for the i-vectors. They represented each senone with a single Gaussian. By this way, they realized comparisons over the same phonetic units. They reported 30% decrease in equal error rate (EER) with such an inclusion of phonetic information on telephone data. Later on, in [23], authors moved to microphone data and applied the training method on telephone and microphone mixed data. They observed that for a microphone test data, adding telephone speech to training data reduced error rates.

ASR DNN - i-vector approach was adopted also in [24–29]. In [24], authors emphasized the performance gain of the system under out of domain data conditions. In [26], the DNN structure is formed with a different network type called time delay deep neural network (TDNN). Besides the drop in EER, a rise in computational efficiency against the heavier load of DNNs over GMMs is showed. The work of [27] applies the method for channel mismatch conditions. In [29], the method is applied on telephone conversation data. Using this approach, they experimented increasing the DNN output components, namely the number of senones, and reached to a better performance.

2.4. Embedding

Some studies brought new speaker representations derived from DNNs, called embeddings. Embedding is obtained by the output of an intermediate level of a neural network. Figure 2.1 shows an embedding extraction representation.

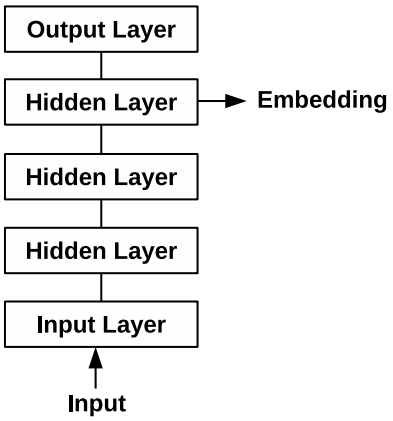

Figure 2.1. Embedding extraction.

In [30], targeting text dependent verification task, a DNN architecture is formed and trained with output labels corresponding to one-hot vectors of single speakers. The training is done at the frame level. When calculating the match score for the complete utterance, statistics of the last hidden layer are averaged, and speaker representations, called d-vectors, are obtained. In an authentication, cosine distance of two d-vectors is compared against a predetermined threshold.

Similarly, in [31], speaker representations, called x-vectors, are extracted from hidden layers of a DNN. DNN training is prepared for classification of speakers. Differently, they added a temporal pooling layer, and extracted the x-vectors just after the pooling layer, giving utterance statistics for variable length of inputs. Comparison scores are generated via PLDA models. In this study, they aimed text independent verification task, and showed that their approach outperforms i-vector approach on short duration recordings. Later, in [32], they augmented the x-vector system with additive noise and reverberation, and obtained better results. In order to better represent utterance level statistics for a text independent task, in [33], a self attention mechanism is employed. The mechanism is taken from sentence embedding literature and does not require the phonetic information for weighting frames. They observed the self attention to be effective even on language mismatched test data. Also, the work in [34] experiments the robustness of the x-vector system on various channels, noise

and reverberation conditions. The study shows the method to be robust against these condition changes.

In [35], the authors focused on short duration text independent verification. They worked on fixed 5 seconds long utterances, and created their network structures correspondingly. They mainly established a CNN structure with attention layers rather than pooling. They fed the network with 5 seconds of utterances, and extracted an embedding from a fully connected layer, then used a PLDA for the comparison of embeddings.

In [36], several types of DNNs are incorporated within both GMM-UBM and i-vector systems as feature extractors. They extracted high dimensional outputs from various hidden layers of DNNs. Then, they reduced dimensionality using principal component analysis (PCA), and obtained new features, called deep features. They ran both GMM-UBM and i-vector frameworks using deep features rather than spectral features, and obtained performance improvements on RSR2015 dataset [21] for text dependent verification task.

2.5. End-to-End Network

End-to-end speaker verification is the structure that starts from raw acoustic features and ends up with a decision for authentication. In this structure, training, enrollment and evaluation phases of the verification are integrated into a single network. Trainings and optimizations are done over this single structure. A basic view of end-to-end structure is as in Figure 2.2. When enrollment utterances of a speaker, and authentication utterance of a speaker are given, system responds with a decision of same/different speaker.

In [37], an end-to-end, text dependent framework is established on "OK Google" dataset. Millions of utterances are used in their work. Single neural network is trained for joint optimization of all components. 1 + N utterances are given to the network, where an utterance is compared with N enrollment utterances of the same speaker.

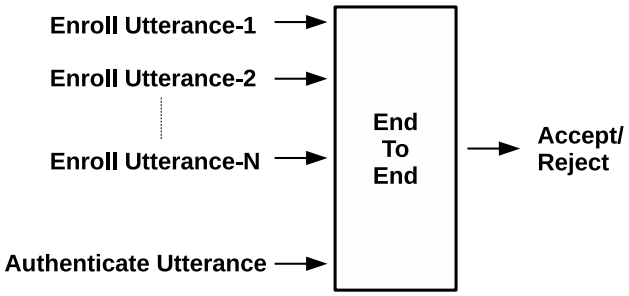

Figure 2.2. Basic end-to-end speaker verification structure.

Promising results are obtained in their work. It is also observed in this study, that using utterance level representations rather than frame level representations and modeling with recurrent neural networks (RNN) rather than feedforward neural networks lowered error rates notably.

Some approaches added attention mechanisms to their end-to-end neural network structures [38,39] for text dependent verification. In [38], experiments are conducted on "Hey Cortana" dataset. They formed the structure on a convolutional neural network (CNN) with attention mechanism to gather utterance level statistics from frame level ones. In the study, they also automatically fed the network with most similar impostors to make it learn the most challenging samples. In [39], database is constructed from "OK Google" and "Hey Google" phrases. A long short-term memory (LSTM) network is used with attention layer in their study.

In [40], an end-to-end speaker verification system is developed for the text independent task. They designed a DNN for the task. For handling variable length input, they placed a temporal pooling layer in the neural network structure. This way, statistics can be accumulated over whole utterance.

Likewise, the work done in [41] presents an end-to-end system. They run experiments on both text dependent and independent cases in Mandarin and English. In experiments, they employed CNNs and RNNs with triplet loss. The triplet loss is a

loss function adopted from image processing field [42]. In this loss, mainly 3 samples are taken as input, one is the anchor, one is positive, and the other is negative. Positive belongs to same speaker with anchor, whereas negative belongs to another speaker than anchor. Updates are done with respect to making anchor-positive cosine similarity larger than that of negative one. Namely, moving anchor towards the positive one. Their experiments give better results than a DNN i-vector system. Also, it is shown that models can be transferred across languages for a finetuning. Even, a model trained in Mandarin shows reliable EER on English data.

With a focus on short utterances, in [43], an end-to-end neural network is employed for text independent verification task. They adopted a very deep CNN framework, called Inception Net [44], with triplet loss. They generated fixed size spectrograms from inputs by cropping or padding, fed them into neural network, and for the comparison used Euclidean distance. Experiments of their study showed a better performance than i-vector baseline. In a following study [45], the authors made changes in their method for variable length inputs. They developed a system to give inputs incrementally rather than cropping, and changed the pooling layer of their CNN structure. They achieved performance gain from both approaches. Then, in [46], they also applied transfer learning in their network structure, and obtained effective results for domain mismatch.

In [47, 48], exact end-to-end structures are created. They removed any feature preparation step and fed the neural network with raw audio samples. Authors in [47] established an RNN structure. In the study, they faced with high computational load, and run tests on only two test speakers, but observed results competitive to a GMM baseline system. In [48], some neural network structures are proposed depending on CNNs and LSTMs with a pre-emphasis layer. They showed promising results on text dependent RSR2015 dataset [21]. Later on, they experimented the system on text independent VoxCeleb [49] data in [50]. In this work, they also defined speaker overfitting problem of embeddings and proposed several approaches to this problem. They showed promising results also on text independent verification task.

In [51], a loss function, called generalized end-to-end loss, is proposed. This loss function works as a training scheme, and provides embeddings of a speaker cluster together, and move away from the other ones. They observed this proposed loss function to be effective on both text dependent and independent verification tasks with respect to EER and training time.

3. THEORETICAL BACKGROUND

In this chapter, a theoretical basis is formed about our proposed verification process.

3.1. Feature Extraction

Features are the compact representations of data. They form a ground front-end for any further modeling. These representations can be shaped according to what to achieve. In the case of speaker verification, it is desired to have features reflecting speaker characteristics discriminately with least change across spoken content, session etc., namely the other changing factors. Mel-Frequency Cepstrum Coefficients (MFCC) is one such widely used feature for speaker characteristic representations.

MFCC calculation depends on generating cepstra from short duration frames. A cepstrum of a signal is generated by taking discrete time Fourier transform (DTFT) of signal, taking logarithm of its magnitude, and taking inverse discrete time Fourier transform (IDTFT) of it [1]. Equation 3.1 and 3.2 show the cepstrum calculations. $X(e^{jw})$ represents DTFT of signal $x[n]$, and $c[n]$ represents the resulting cepstrum.

$$c[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log |X(e^{jw})| e^{jwn} dw \tag{3.1}$$

$$X(e^{jw}) = \sum_{n=-\infty}^{\infty} x[n] e^{-jwn} \tag{3.2}$$

Other than transforming signal into cepstral domain, the process also includes a mel scaling step. Mel is the unit of pitch, generated from human auditory perception experiments. It shows a nonlinear relationship with frequency of the tone as given in Eq 3.3. In mel scaling, this nonlinearity is reflected with equally separated filterbanks in mel domain. Figure 3.1 shows filterbanks generated for frequencies up to 4kHz. In

cepstrum calculation, after the DTFT, frequencies are grouped under corresponding filterbank, then the process continues on them.

$$mel = 1127 \log_e(1 + f/700) \tag{3.3}$$

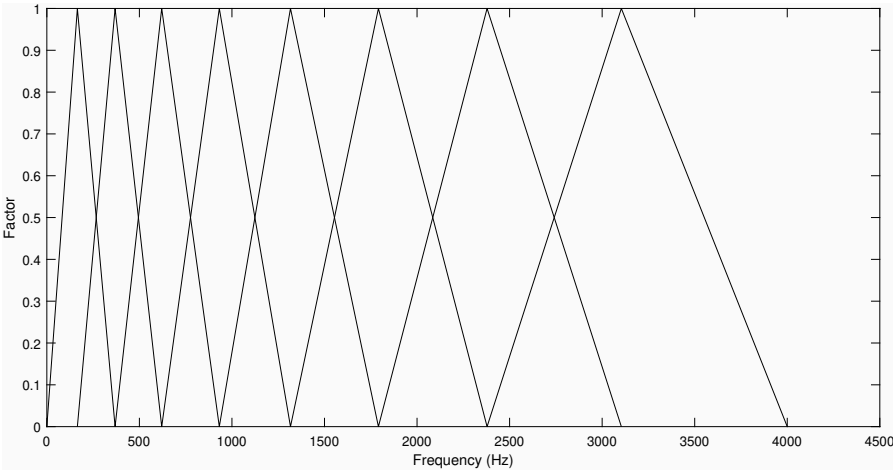

Figure 3.1. Mel filterbank.

The overall view of the MFCC blocks is given in Figure 3.2.

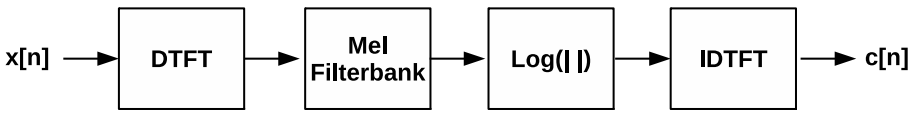

Figure 3.2. MFCC blocks.

3.2. Automatic Speech Recognition

ASR systems aim to generate text content of a given speech signal. It can be inspected in two phases as training and decoding. The training also can be further divided into two as acoustic modeling and language modeling. In acoustic modeling, the acoustic characteristic of the speech is gathered. In language modeling, the characteristic of the grammar of the target content domain is extracted.

In statistical terms, the ASR problem is given in Equation 3.4. It states that, the problem is finding the words $\hat{W}$, that maximizes the posterior probability, $P(W|X)$, of words W, given observation X. Using Bayes' rule, it can be restated as the multiplication of the likelihood, $P(X|W)$, of observation X given words W with the prior probability of words $P(W)$ occurring. Here, the $P(X)$ of the denominator, coming from Bayes' rule is discarded due to its independence from optimizations being done over W's [1].

$$\hat{W} = \arg\max_{W} P(W|X) \tag{3.4}$$

$$\hat{W} = \arg\max_{W} P(X|W)P(W)$$

Equation 3.4 can be decomposed into the main units of the ASR process. $P(X|W)$ is related to acoustic modeling, $P(W)$ is related to language modeling, and $\arg\max_W$ is related to decoding of the model.

3.2.1. Acoustic Modeling

Acoustic modeling is forming a feature pattern for given words. Acoustic modeling mainly contains a structure for keeping temporal variability, and a structure for generating posterior probabilities in a temporal context. Former is achieved via hidden Markov models (HMM), and latter is achieved via GMM or DNN structures.

HMMs are represented with mainly states, transitions between them, and emitted observations from them. Figure 3.3 represents an HMM structure with 5 states. a_{ij}'s represent transition probabilities between states, $b_i(X_t)$'s represent the posterior probabilities of observing X at time t in state i. In this sample HMM, every state has self transitions also. HMM parameter estimation can be done using Baum Welch algorithm [1]. The posteriors of the HMM can be generated with either GMMs or DNNs [52].

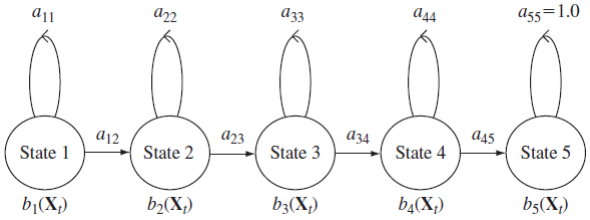

Figure 3.3. HMM structure [1].

3.2.2. Language Modeling

Language modeling is forming a word occurrence pattern using grammatical rules of target domain. It provides content coverage of the model. Grammar rules can be explicitly specified with word co-occurence probabilities or deduced by statistics of a training text corpus. In case of statistics approach, N-gram is a popular way for estimation.

In N-gram, words are taken into consideration with their previous N-1 neighboring words. Let a word sequence W have probability $P(W)$, consisting of M words as in Equation 3.5. In an N-gram language model, every word will be dependent on previous N-1 words, thus the expression will be as in Equation 3.6 [1].

$$P(W) = P(W_1, W_2, ..., W_M) \tag{3.5}$$

$$P(W) = \prod_{n=1}^{M} P(W_n | W_{n-1}, W_{n-2}, ..., W_{n-N+1}) \tag{3.6}$$

The n-gram probabilities can be estimated by word sequence frequencies in the text. For a trigram case, Equation 3.7 shows a basic approach for estimation. It calculates the probability of current word, dependent on the previous two, with the rate of trigram frequency to bigram frequency of previous two words. This basic approach may suffer

from absence of words, for them, there are smoothing solutions too.

$$P(W_n|W_{n-1}, W_{n-2}) = \frac{C(W_{n-2}, W_{n-1}, W_n)}{C(W_{n-2}, W_{n-1})} \tag{3.7}$$

3.2.3. Decoding

The decoding phase is about finding best word sequence representation for given features. A single structure is formed by merging acoustic and language models, then the decode process goes on this single structure.

Decoding can be a time consuming job for large vocabulary texts. As en efficient approach, weighted finite state transducers (WFST) provide a framework suitable for this job.

WFSTs form an efficient decoding graph with optimizations defined in framework. A WFST contains states, transitions between states, weights and input-output labels on transitions, and end state weights. Figure 3.4 shows a sample WFST structure. Each state transition carries a weight, and an input-output symbol pair. A WFST generates an output string for a given input string, namely transduces it.

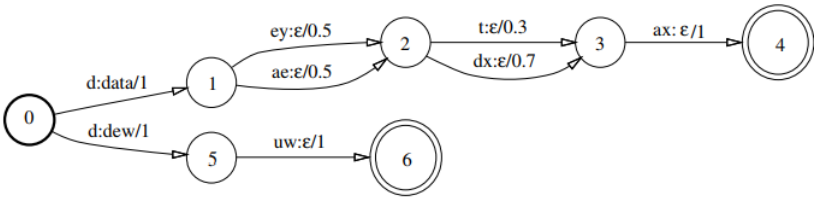

Figure 3.4. Sample WFST for pronunciation generation [2].

Some optimizations are defined in WFST framework. Two important ones are determinization and minimization. In determinization, it is ensured that there are no two transitions from a state with same input label. In minimization, the graph size is reduced [2].

For the speech recognition task, four main WFST structures can be formed. One is word level grammar (G). Second one is pronunciation lexicon (L). Third one is phoneme context dependency (C). Last one is acoustic HMM (H). In G, a grammar structure is formed by word relations. Its both input and output labels are words. The word sequence and weights have the importance. In L, the pronunciations of the words are kept, namely the phoneme sequences. Its input labels are phonemes and output labels are words. In C, context dependency of the phonemes are kept. Its input labels are context dependent phonemes, output labels are context independent phonemes. In H, HMM structure of the acoustic model is represented. Its input labels are acoustic features and output labels are context dependent phonemes. Decoding graph is basically generated from composition of these four WFSTs as in Equation 3.8 [2]

$$H \circ C \circ L \circ G \tag{3.8}$$

After applying the optimizations, the formula becomes as in Equation 3.9, where *min* and *det* represent minimization and determinization. The tilde above WFSTs mean that auxiliary symbols added to make them determinizable. The π_ϵ represents the operation of replacing the auxiliary symbols with epsilons. [2].

$$\pi_\epsilon(min(det(\tilde{H} \circ det(\tilde{C} \circ det(\tilde{L} \circ G))))) \tag{3.9}$$

3.3. Artificial Neural Networks

Neural networks are structures that learn and reflect patterns of data. They can be defined as composition of many different functions [53].

A neural network basically consists of fundamental information processing units, called neurons. A neuron can be modeled by its incoming weighted connections, a summing function and an activation function [54]. Figure 3.5 depicts basics of such a neuron modeling.

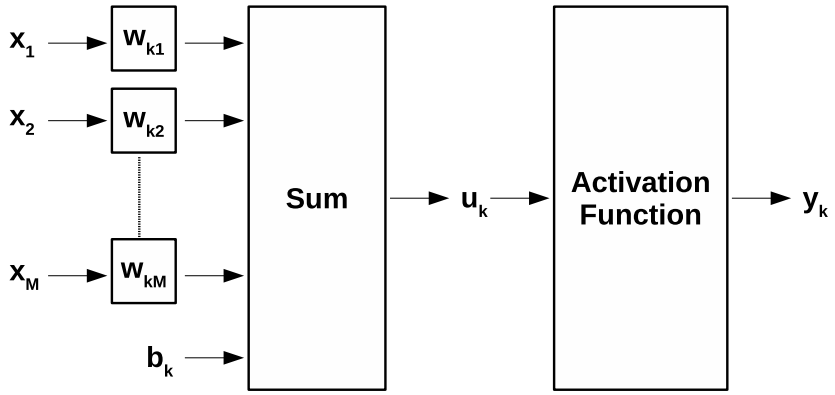

Figure 3.5. Basic modeling of neuron structure.

Equation 3.10 and 3.11 give the corresponding equations of the neuron model. In Equation 3.10, the summation part is given, where x_j is j^{th} input, w_{kj} is weight, b_k is a bias term, and u_k is the output of the summation for k^{th} neuron.

$$u_k = b_k + \sum_{j=1}^{M} w_{kj} x_j \tag{3.10}$$

In Equation 3.11, the activation part is given for neuron modeling, where ϕ is the activation function, and y_k is the output of the k^{th} neuron.

$$y_k = \phi(u_k) \tag{3.11}$$

A layer is formed from a number of neurons together. Neurons of a layer are connected to the neurons of the previous layer with weights.

Figure 3.6 shows a fully connected ANN sample with four layers. It has two neurons in input layer, three neurons in both hidden layers, and two neurons in output layer.

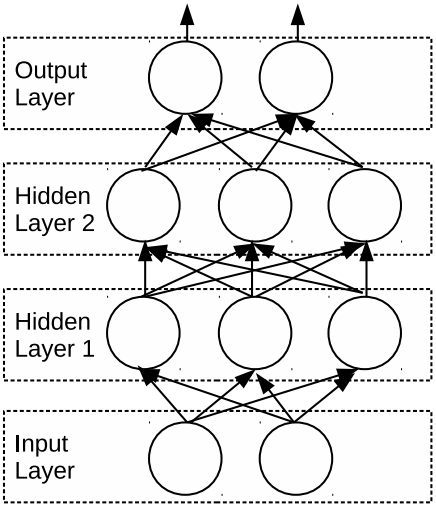

Figure 3.6. ANN structure.

3.3.1. Cost Function

Cost function shows how well the system performs with seen samples. For this, the outputs of neural network are compared with the true labels of the given samples.

Cross-entropy and mean squared error are two samples of cost functions used for classification and regression problems, respectively. Equation 3.12 shows the formula of cross-entropy. Here, t_i represents the target probability, and p_i represents the generated probability of the network for class i. For higher values of the generated true class probability, the cost will be lower.

$$C = -\sum_{i} t_i \log(p_i) \tag{3.12}$$

As for mean squared error, Equation 3.13 shows the formula for this cost function. Here, N is the output neuron size, y_i is the network generated value and $\tilde{y}_i$ is the labeled value of the neuron i. The cost becomes less, as generated value gets closer to

the labeled one.

$$C = \frac{1}{N}\sum_{i=1}^{N}(y_i - \tilde{y}_i)^2 \tag{3.13}$$

3.3.2. Activation Function

Activation functions are the units responsible for limiting outputs of neurons in neural networks [54]. They also work as nonlinearity providers. There are several widely used activation function types.

3.3.2.1. Sigmoid. The sigmoid function is given in Equation 3.14 [55].

$$f(z) = \frac{1}{1 + e^{-z}} \tag{3.14}$$

The function response graph would be as in Figure 3.7.

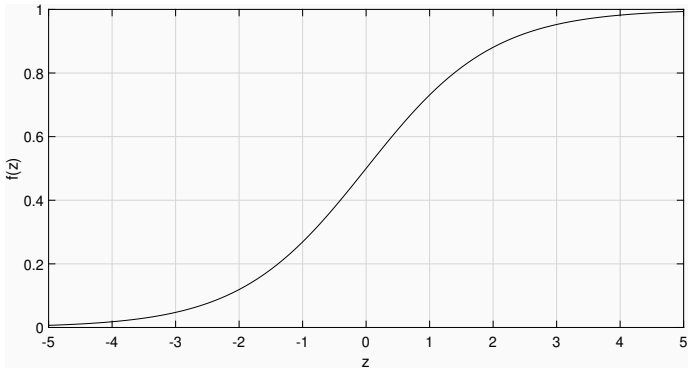

Figure 3.7. Sigmoid function response.

3.3.2.2. Tanh. The hyperbolic tangent, tanh, function is given in Equation 3.15 [55].

$$f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3.15}$$

The function response graph would be as in Figure 3.8.

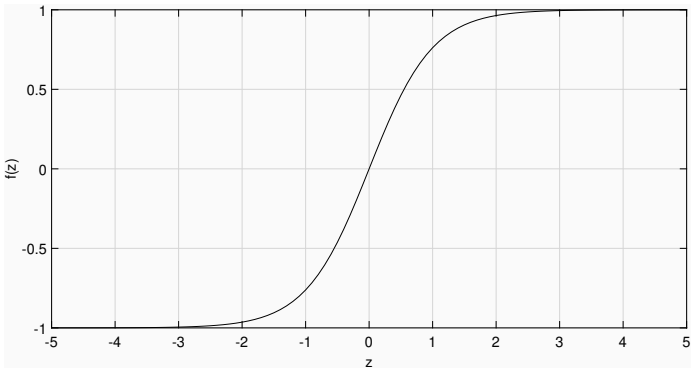

Figure 3.8. Tanh function response.

3.3.2.3. ReLU. The rectified linear unit, ReLU, function is given in Equation 3.16 [55].

$$f(z) = \max(0, z) \tag{3.16}$$

The function response graph would be as in Figure 3.9.

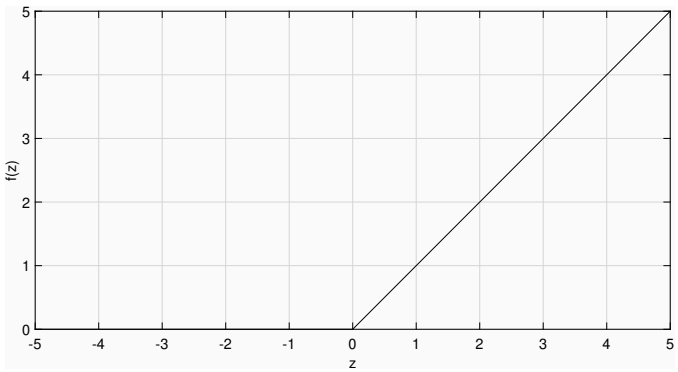

Figure 3.9. ReLU function response.

3.3.2.4. Softmax. Softmax can be used to obtain probability distribution over a collection. It is useful as the output layer of a classifier neural network. Its function is

given in Equation 3.17 for i^{th} unit z_i [53].

$$f(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3.17}$$

3.3.3. Learning

Minimization of the cost function forms the objective of neural network learning process. The main idea in learning is to find best network weights for the lowest error. The used approach is checking the output change with respect to changes in weights, and move the weights in the direction lowering the output error. This change relation between weights and error is formed via backpropagation procedure [56]. Then the optimal solution is found with gradient descent algorithm.

3.3.3.1. Backpropagation. Backpropagation is mainly a procedure to find the relationship between network weights and the cost. Derivatives of the cost function with respect to weights are taken into consideration. In a forward propagation stage, the network is run towards the end, with constant weights, and generates the output and corresponding cost. In backpropagation stage, the gradients are calculated. The calculations are derived using the chain rule of the derivatives as in Equation 3.18.

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \tag{3.18}$$

Let the activation functions and weighted sums be as follows,

$$y_k = \phi(x_k)$$

$$x_k = \sum_{h \in H} w_{hk} y_h$$

where x_k and y_k are the input output pair of the neuron $k \in layerK$, ϕ is the activation function, w_{hk} is the weight from neuron $h \in layerH$ to neuron k.

For hidden layer I and output layer J, the backpropagation works as follows in case of mean square error cost function and sigmoid activation function neglecting constant multipliers.

$$\frac{\partial E}{\partial y_j} = y_j - \tilde{y}_j$$

Then, the following can be written by chain rule and derivative of the sigmoid function.

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial x_j}$$

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} y_j(1 - y_j)$$

Then, the following again can be written using the chain rule.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial w_{ij}}$$

which is equal to the following, using derivative of the weighted sums. This gives the derivate of error with respect to last level weights.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} y_i$$

While transitioning to the next level, the following can be written for the effect of i^{th} output on j^{th} input using the summation formula.

$$\frac{\partial x_j}{\partial y_i} = w_{ij}$$

Accumulating the effect over all other inputs, the equation becomes as below. This shows that the derivative can be calculated for any layer output, and thus weights, using this chain logic [56].

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} w_{ij}$$

3.3.3.2. Gradient Descent. Gradient descent, in neural networks, is the way of finding minimum point in the error gradient surface with respect to weights. It seeks the steepest descending route to go to the minimum point faster. It may suffer from reaching a local minimum rather than a global minimum.

The main formula of gradient descent is given in Equation 3.19 [53]. The parameter vector $\boldsymbol{x}$ is updated to $\boldsymbol{x'}$, using the gradient of the cost function $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ and learning rate ϵ. Learning rate works as the step size of movement towards the minimum.

$$\boldsymbol{x'} = \boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{3.19}$$

There are many types of optimizers that can run optimizations using batches of data. Stochastic gradient descent (SGD), Adam [57], Adagrad [58], RMSprop [59] are some widely used samples of optimizers.

3.4. Error Calculation

In speaker verification task, EER is a widely used error metric. In EER, the error rate is given at the point where false acceptance rate (FAR) crosses the false rejection rate (FRR). It provides also the working threshold of the system for acceptance/rejection.

A confusion matrix is given in Table 3.1 to show the relations between actual values and predicted values. Here TP is true positive, TN is true negative, FP is false positive and FN is false negative.

Table 3.1. Confusion matrix for actual and predicted identities.

		Predicted Identity	
		Genuine	Impostor
Actual Identity	Genuine	TP	FN
	Impostor	FP	TN

In Equation 3.20 and 3.21, formulas for FAR and FRR are given.

$$FAR = \frac{FP}{FP + TN} \tag{3.20}$$

$$FRR = \frac{FN}{FN + TP} \tag{3.21}$$

4. METHODOLOGY

In this chapter, the proposed methodology is described. The building blocks are given in detail.

4.1. Overall System

The proposed system is designed as a single network structure which is optimized in training, and used in both enrollment and authentication. The overall flow of the system is given in Figure 4.1.

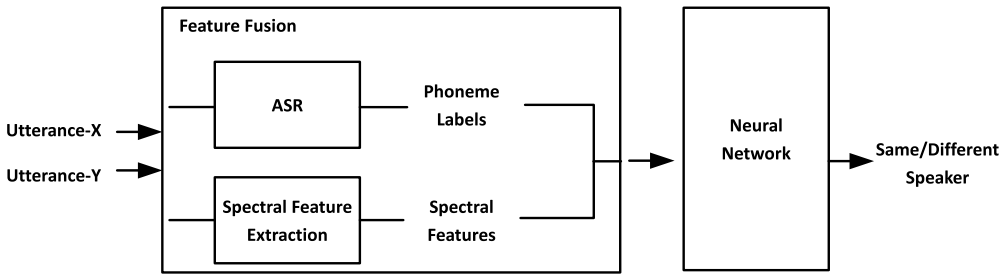

Figure 4.1. Overall flow of the system.

The system mainly works as a similarity decision mechanism on utterances. Given two utterances, firstly a feature fusion process takes place. In feature fusion, spectral features and phonetic content are extracted and merged for both utterances. The fusion process produces a collection of features for each detected phonetic unit. Then, the neural network is fed with these phoneme level features. The same/different speaker decision is taken from the output of the neural network.

4.2. Spectral Feature Extraction

As spectral features, 20 dimensional MFCCs are used. They are extracted from 25ms long frames with 10ms shifts. Sampling rate of 8000 samples/sec is used.

Over training data, mean and variance of the MFCCs are calculated. Using these statistics, cepstral mean and variance normalization (CMVN) is applied on training, development and test data globally. Equation 4.1 shows application of CMVN on an MFCC vector. For i^{th} dimension, x_i is the feature value, $\tilde{x}_i$ is the normalized version of it, μ_i and σ_i are the mean and standard deviation calculated from training data.

$$\tilde{x}_i = \frac{x_i - \mu_i}{\sigma_i} \tag{4.1}$$

4.3. Phoneme Recognition

Phoneme recognition task is done using an ASR module. On top of an acoustic model, a language model is generated using a broad range of texts from various sources in English. Pronunciations of every word in training texts are extracted and these words are replaced with their corresponding pronunciations. Every phoneme is treated like a word unit of the language model. Thus, the language model training corpus is formed via phoneme sequences.

The language model is trained with 8-gram configuration due to handling phoneme contexts. Then, it is composed with acoustic model, giving the final ASR model for phoneme recognition. Each recording is decoded using that model, and recognized phoneme type, start-end time information are kept.

4.4. Phoneme Feature Fusion

Extracted spectral features and phoneme information are fused before neural network running. Given two utterances, all phoneme boundaries are extracted. Common phonemes of the utterances are determined as in Figure 4.2. From the recognized phoneme sequences of both utterances, the common ones are paired. Iteratively, the used ones are skipped, and next pairings are searched.

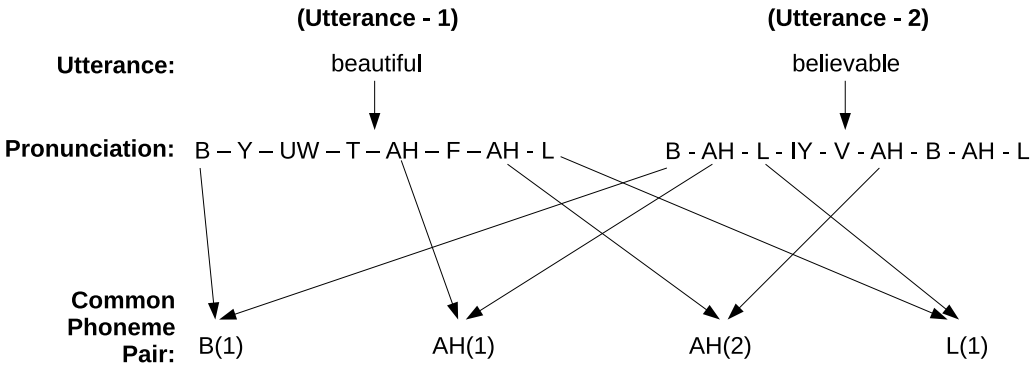

Figure 4.2. Phoneme pairing from two utterances.

For every phoneme pair, their corresponding MFCCs are found and 5 of them are selected evenly as in Figure 4.3. If frame count is not enough in a phoneme region, some of the frames are repeated. This is done for both parties and extracted vectors are concatenated. This results in a 2x5x20=200 dimensional vector ([number of utterances] x [selected feature count] x [feature dimension]).

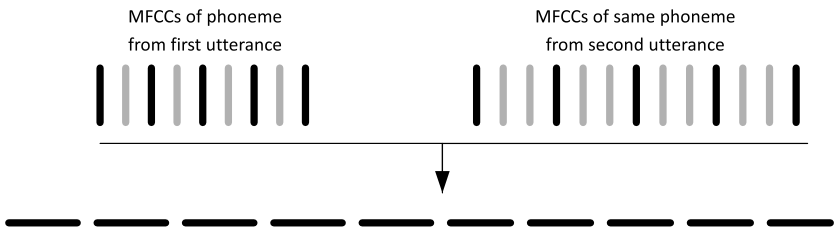

Figure 4.3. Feature fusion.

Phonemes are coded as separate one-hot vectors. The common phoneme ID is mapped to the corresponding one-hot vector and appended to the resulting vector. For the designed system there are 39 English phonemes, thus the vector dimension increases to 200+39=239. This vector is ready to be fed into neural network.

4.5. Neural Network Architecture

A feedforward neural network structure is formed with fully connected layers as in Figure 4.4. There are input, output layers and two hidden layers. Each hidden layer has 2048 neurons with ReLU activation function. Output layer has two neurons with softmax activation function. Output layer is designed to represent the binary classification problem of same/different speaker. As cost function, cross entropy is used. As the optimizer, SGD is used.

In neural network training, a separate development set is used. Before overfitting to the training data, early stopping is applied at best performing iteration on development set.

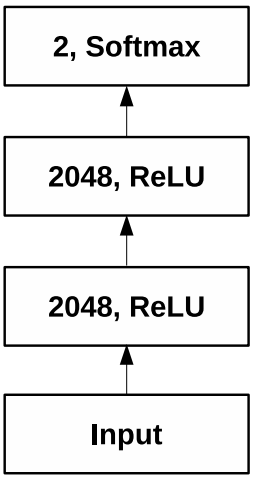

Figure 4.4. Neural network architecture.

4.6. Scoring

For every utterance pair, common phoneme features are fused and fed into neural network and soft decision scores are obtained before softmax activation. All obtained scores are averaged and a final score is given for the utterance comparison. Every comparison is done for 1 enroll and 1 authentication utterance condition. Final evaluations are done on EER metric.

5. EXPERIMENTS

In this chapter, the conducted experiments are given. Used database, test configurations and application details are presented.

5.1. Database

The experiments are conducted on English language. NIST SRE and Switchboard databases are used during the study. They contain mostly conversational telephone and microphone speech. All of Switchboard data is utilized in training. SRE data is separated into training, development and test parts with non-overlapping speakers.

5.2. Configuration

Training Dataset. The effect of training dataset change is investigated. In one case, only SRE data is used for training. In another case, Switchboard data is added to SRE. In both cases SRE test set is commonly used.

Phoneme Specific Effects. The effect of every phoneme to performance is inspected in this configuration. Error rates are calculated over only single phonemes to get a better idea for speaker discriminative representations of separate phonemes.

Phoneme to Utterance Level. The performance gain is investigated for combination of phoneme level representations into utterance level. In phoneme level calculations, neural network classification errors are taken into account. In utterance level calculation, neural network outputs are accumulated over the whole utterances, and an utterance level score is obtained from comparison of the two. Then, over the whole test set, an EER metric is calculated.

Soft Scores. Two types of phoneme level score accumulation approaches are evaluated in this configuration. One can be called as phoneme level hard scores, in which every phoneme pair is labeled as 1 or 0, according to its generated class using softmax, and average of these 1/0 decisions is calculated. The other one can be called as phoneme level soft scores, in which every phoneme pair score is generated from softmax input neuron value differences, and these scores are averaged.

Addition of Phoneme ID. The effect of adding phoneme IDs in fused features is analyzed. For every common phoneme, the recognized phoneme is mapped into a one-hot vector and appended to the feature vectors.

Addition of F0 Median. Adding fundamental frequency (F0) median value to feature fusion is experimented. F0 estimations are done within each phoneme boundaries. The median of these F0 values are appended to the feature vector for both sides of authentication for every common phoneme.

Cepstral Mean Normalization within Utterance. Cepstral mean normalization (CMN) is applied to features. The statistics are obtained for each utterance, and normalization is done across a single utterance.

Test Speech Duration and Kaldi Baseline. In this configuration, effect of net speech duration is inspected. Test recordings are clipped according to total speech durations, and new test sets are produced. The performance of the proposed system is calculated for these test sets. Also, a baseline system is formed with Kaldi x-vector recipe. The proposed system performance is compared with baseline system for various speech lengths.

Forced Alignment. In all other test configurations, phoneme boundaries are extracted using an ASR module. This module is expected to have its internal error in recognition, which may affect the performance of overall system. In order to isolate the effects of ASR module and see upper performance limits of remaining algorithm, another scenario is configured. In this case, the phoneme boundaries are not recognized, rather they are obtained by forced alignment. For that to work, a transcribed dataset is needed. A transcribed Switchboard subset is used for this configuration. The dataset consists of 231 speakers with 2 sessions. It is separated into 3 non-overlapping groups, 186 are used for training, 22 for development, and 23 for test. Two cases are experimented in this configuration. First case is the one with ASR, and the second is the one with forced alignment.

5.3. Results

Training Dataset. The experimental results are given in Table 5.1 for training dataset effect analysis. The results show that an increase in the training data from another domain results in an improvement on the phoneme level classification error.

Table 5.1. Effect of training dataset in performance.

Training Dataset	# Speakers	# ANN Train Entries	Classification Error
SRE	3200	4.9M	29.4%
SRE + Switchboard	5632	11M	27.4%

Phoneme Specific Effects. The ARPABET phonemes, their sample uses, and their specific effects on the performance are given in Table 5.2 ordered by error rates. The results show slight performance difference among different phonemes. According to the results, vowels have a tendency to have lower error rates, whereas higher error rates are observed on unvoiced consonants. It might be expected /K/, /P/, /T/, /F/ phonemes to carry less information about the speaker than vowel phonemes due to their short duration and noise-like behaviour.

Table 5.2. Phoneme specific performance changes.

Phoneme	Sample Use	Error	Phoneme	Sample Use	Error
AY	f(i)le	24.3%	L	(l)ie	27.8%
EY	f(a)ce	24.8%	D	(d)ye	28%
AW	m(ou)th	25.1%	ZH	vi(s)ion	28%
DH	fa(th)er	25.2%	CH	cat(ch)	28.1%
AE	(a)t	25.3%	OY	v(oi)d	28.4%
EH	m(e)n	25.4%	UH	g(oo)d	28.4%
NG	si(ng)er	25.4%	G	(g)uy	28.7%
AA	f(a)ther	25.6%	ER	w(or)d	28.8%
IY	s(ee)d	26%	S	ma(ss)	28.8%
AO	f(a)ll	26%	W	(w)ye	28.9%
IH	f(i)ll	26.2%	HH	(h)igh	29%
JH	(g)iant	26.3%	V	ha(ve)	29.1%
UW	f(oo)d	26.6%	TH	(th)eta	29.5%
N	(n)ight	26.6%	B	(b)uy	30%
OW	c(o)de	26.6%	SH	(sh)y	30.1%
Y	(y)es	26.6%	F	(f)an	30.3%
R	t(r)y	26.7%	T	(t)ie	31.9%
Z	(z)oo	27.4%	P	(p)ie	32%
AH	g(u)n	27.5%	K	s(k)y	32.9%
M	(m)y	27.6%			

Phoneme to Utterance Level. The effect of transitioning to utterance level scoring on EER metric is given in Table 5.3. Results show that utterance level statistics contribute to speaker specific information retrieval.

Table 5.3. Transition from phoneme to utterance level scoring.

Phoneme Level Classification Error	Utterance Level EER
27.4%	18.0%

Soft Scores. The effect of accumulating hard or soft scores is given in Table 5.4. Results show a performance gain with the use of soft decisions. Some information may be lost during conversion of soft decisions into 1/0 decisions in each phoneme.

Table 5.4. Accumulating hard or soft scores.

Score Type	EER
Hard	18.0%
Soft	17.4%

Addition of Phoneme ID. Addition of phoneme ID as one-hot vector into the fused feature affects the performance as in Table 5.5. The results show that specifying the common phoneme ID with a one-hot vector improves the performance. It may be easing the neural network to better represent different phonemes.

Table 5.5. Effect of phoneme ID addition in feature fusion.

Add Phoneme ID	EER
No	17.4%
Yes	15.5%

Addition of F0 Median. Result of F0 median addition into fused feature given in Table 5.6. Tests show that adding the F0 median in every phoneme boundary does not improve the performance.

Table 5.6. Effect of F0 median addition in feature fusion.

Add F0 Median	EER
No	15.5%
Yes	16.7%

Cepstral Mean Normalization within Utterance. The effect of applying CMN within utterance boundary is given in Table 5.7. According to the results, applying utterance CMN does not show a performance improvement.

Table 5.7. Effect of applying utterance CMN.

Apply Utterance CMN	EER
No	15.5%
Yes	17.4%

Test Speech Duration and Kaldi Baseline. The performance change with respect to test data speech duration is given in Table 5.8. It contains test results for various speech durations in both enroll and authentication utterances. The cases are full utterances, and clipped utterances at 15, 10, 5, 3, 1 seconds. Also the results of the proposed system are given comparatively with results obtained using Kaldi x-vector recipe. Results show that the x-vector method performs well on longer duration recordings. When the speech length is reduced, it degrades significantly. The performance of the proposed approach also degrades with duration limitations but does not have a severe change in performance. In longer recordings, especially the full recordings, the proposed system shows performance behind the x-vector approach. As the recordings

become shorter, the methods perform closer, and at 1 second recordings, the proposed approach shows lower EER than x-vector. Also the detection error tradeoff (DET) curve for 1 second condition in Figure 5.1 shows the false reject rate and false accept rate relations of both approaches at various thresholds. Proposed system seems to perform better at all operation points in this condition.

Table 5.8. Effect of test speech duration on proposed system and Kaldi x-vector system.

Speech Duration	Proposed System EER	X-vector EER
Full-Full	15.5%	0.8%
15sec-15sec	16.7%	3.1%
10sec-10sec	16.7%	5.0%
5sec-5sec	18.6%	9.3%
3sec-3sec	19.9%	17.4%
1sec-1sec	23.7%	32.2%

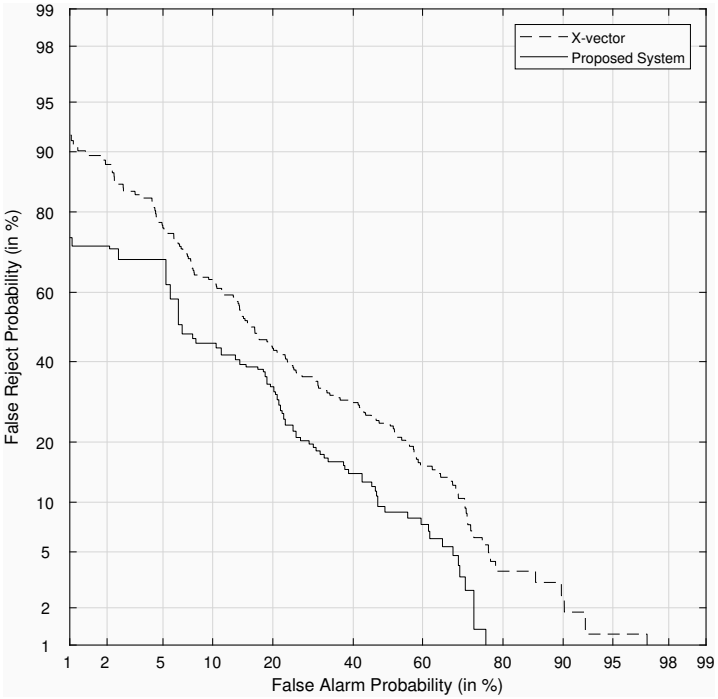

Figure 5.1. DET curves of proposed and x-vector systems for 1sec-1sec condition.

Forced Alignment. The performance comparison of ASR and forced alignment approaches for phoneme recognition is given in Table 5.9. According to results the forced alignment approach performed slightly better.

Table 5.9. Performance comparison with ASR and forced alignment.

Phoneme Recognition Approach	EER
ASR	17.8%
Forced Alignment	17.4%

5.4. Application

During the study, for x-vector calculations, Kaldi toolkit is used [60]. They have a recipe for speaker verification using x-vectors. Also they provide pre-trained models of their publications. These models are used in tests. As feature extraction and speech recognition modules, also, Kaldi toolkit is used.

The neural network structure is formed using Computational Network Toolkit (CNTK) of Microsoft [61]. It provides layers, optimizers, loss functions etc. about neural networks, and eases to establish one. It provides application programming interface (API) for C++, C# and Python. In this study, its C# and Python APIs are used.

6. CONCLUSIONS AND FUTURE WORK

In this chapter, the conclusions are made about the study, and some comments and future directions are given depending on the obtained results.

6.1. Conclusion

In this study, a system is proposed for text independent speaker verification. In this system, phonetic content information is fused with spectral acoustic features. The classification is done using a fully connected feedforward neural network structure.

Proposed system is implemented and its performance is measured on English public databases, NIST SRE and Switchboard. The experiments are conducted on various conditions.

Enriching the training data from another domain had a positive effect on the performance. The more training data is expected to increase the performance further.

As inspected phoneme by phoneme, it is seen that vowels have lower error rates, whereas unvoiced consonants tend to have higher error rates. Such a difference is expected due to the noisy behaviour of unvoiced consonants and their short durations.

Rather than phoneme level, utterance level representations resulted in higher performance. Acquiring the statistics over whole utterance recovers some of the errors made on phoneme level.

In accumulating the phoneme level scores over the utterance, using the soft scores rather than 1/0 decisions resulted in lower error rate. It shows that the floating point score of the neural network supplies more information than a binary decision, thus it is better not to lose this information.

When fusing the features, addition of phoneme ID and F0 median is also tested. Specifying the phoneme ID in fused feature vector increased the performance, but addition of F0 median did not give a performance increase.

Also, the effect of utterance level CMN is tested, and it is observed that it increases the error rate.

The proposed system is also compared with an x-vector baseline setup for various speech lengths. In these comparisons, it is observed that the x-vector setup has a better performance on long duration recordings. When speech durations become shorter, the gap closes and the proposed system performs better on 1 second recordings. The EER ranges from 16.7% to 23.7% for speech lengths between 15 seconds and 1 second. When the whole recording is used the performance was 15.5% EER.

6.2. Future Work

The proposed system shows some acceptable performance especially on short duration recordings. However, these error rates still need to be improved. In the future, the fusion of phonetic content can be accompanied with the context of the phonemes, which may cause better consistency in comparisons. Different fusion approaches may be formed to hold more information about the speaker. An alternative approach may be developed to hold all phoneme averages over the utterance. Other neural network structures may be investigated for better modeling. Also, the training data may be increased with other datasets to make the system learn comparison varieties. Finally, augmenting the data with noise and reverberation may be another key to improvements.

REFERENCES

1. Rabiner, L. R. and R. W. Schafer, *Theory and applications of digital speech processing*, Vol. 64, Pearson Upper Saddle River, NJ, 2011.

2. Mohri, M., F. Pereira and M. Riley, "Speech recognition with weighted finite-state transducers", *Springer Handbook of Speech Processing*, pp. 559–584, Springer, 2008.

3. Jain, A. K. and A. Ross, "Multibiometric systems", *Communications of the ACM*, Vol. 47, No. 1, pp. 34–40, 2004.

4. Schuckers, S. A., "Spoofing and anti-spoofing measures", *Information Security technical report*, Vol. 7, No. 4, pp. 56–62, 2002.

5. Hansen, J. H. and T. Hasan, "Speaker recognition by machines and humans: A tutorial review", *IEEE Signal processing magazine*, Vol. 32, No. 6, pp. 74–99, 2015.

6. Reynolds, D. A., T. F. Quatieri and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models", *Digital signal processing*, Vol. 10, No. 1-3, pp. 19–41, 2000.

7. Dempster, A. P., N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society: Series B (Methodological)*, Vol. 39, No. 1, pp. 1–22, 1977.

8. Auckenthaler, R., M. Carey and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems", *Digital Signal Processing*, Vol. 10, No. 1-3, pp. 42–54, 2000.

9. Barras, C. and J.-L. Gauvain, "Feature and score normalization for speaker verification of cellular data", *2003 IEEE International Conference on Acoustics, Speech,*

and Signal Processing, 2003. Proceedings.(ICASSP'03)., Vol. 2, pp. II–49, IEEE, 2003.

10. Bimbot, F., J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-García, D. Petrovska-Delacrétaz and D. A. Reynolds, "A tutorial on text-independent speaker verification", *EURASIP Journal on Advances in Signal Processing*, Vol. 2004, No. 4, p. 101962, 2004.

11. Kinnunen, T. and H. Li, "An overview of text-independent speaker recognition: From features to supervectors", *Speech communication*, Vol. 52, No. 1, pp. 12–40, 2010.

12. Fazel, A. and S. Chakrabartty, "An overview of statistical pattern recognition techniques for speaker verification", *IEEE Circuits and Systems Magazine*, Vol. 11, No. 2, pp. 62–81, 2011.

13. Kenny, P., G. Boulianne, P. Ouellet and P. Dumouchel, "Speaker and session variability in GMM-based speaker verification", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 4, pp. 1448–1460, 2007.

14. Dehak, N., P. J. Kenny, R. Dehak, P. Dumouchel and P. Ouellet, "Front-end factor analysis for speaker verification", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 19, No. 4, pp. 788–798, 2011.

15. Matějka, P., O. Glembek, F. Castaldo, M. J. Alam, O. Plchot, P. Kenny, L. Burget and J. Černocky, "Full-covariance UBM and heavy-tailed PLDA in i-vector speaker verification", *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4828–4831, IEEE, 2011.

16. Kenny, P., T. Stafylakis, P. Ouellet, M. J. Alam and P. Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration", *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7649–7653, IEEE, 2013.

17. Kanagasundaram, A., R. Vogt, D. B. Dean, S. Sridharan and M. W. Mason, "I-vector based speaker recognition on short utterances", *Proceedings of the 12th Annual Conference of the International Speech Communication Association*, pp. 2341–2344, International Speech Communication Association (ISCA), 2011.

18. Garcia-Romero, D. and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems", *Twelfth annual conference of the international speech communication association*, 2011.

19. Bousquet, P.-M., D. Matrouf and J.-F. Bonastre, "Intersession compensation and scoring methods in the i-vectors space for speaker recognition", *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

20. Larcher, A., P.-M. Bousquet, K. A. Lee, D. Matrouf, H. Li and J.-F. Bonastre, "I-vectors in the context of phonetically-constrained short utterances for speaker verification", *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4773–4776, IEEE, 2012.

21. Larcher, A., K. A. Lee, B. Ma and H. Li, "RSR2015: Database for text-dependent speaker verification using multiple pass-phrases", *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

22. Lei, Y., N. Scheffer, L. Ferrer and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network", *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1695–1699, IEEE, 2014.

23. Lei, Y., L. Ferrer, M. McLaren and N. Scheffer, "A deep neural network speaker verification system targeting microphone speech", *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

24. Garcia-Romero, D., X. Zhang, A. McCree and D. Povey, "Improving speaker recognition performance in the domain adaptation challenge using deep neural net-

works", *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 378–383, IEEE, 2014.

25. Kenny, P., V. Gupta, T. Stafylakis, P. Ouellet and J. Alam, "Deep neural networks for extracting baum-welch statistics for speaker recognition", *Proc. Odyssey*, pp. 293–298, 2014.

26. Snyder, D., D. Garcia-Romero and D. Povey, "Time delay deep neural network-based universal background models for speaker recognition", *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 92–97, IEEE, 2015.

27. McLaren, M., Y. Lei and L. Ferrer, "Advances in deep neural network approaches to speaker recognition", *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4814–4818, IEEE, 2015.

28. Richardson, F., D. Reynolds and N. Dehak, "Deep neural network approaches to speaker and language recognition", *IEEE Signal Processing Letters*, Vol. 22, No. 10, pp. 1671–1675, 2015.

29. Sadjadi, S. O., S. Ganapathy and J. W. Pelecanos, "The IBM 2016 speaker recognition system", *arXiv preprint arXiv:1602.07291*, 2016.

30. Variani, E., X. Lei, E. McDermott, I. L. Moreno and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification", *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4052–4056, IEEE, 2014.

31. Snyder, D., D. Garcia-Romero, D. Povey and S. Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification.", *Interspeech*, pp. 999–1003, 2017.

32. Snyder, D., D. Garcia-Romero, G. Sell, D. Povey and S. Khudanpur, "X-vectors:

Robust DNN embeddings for speaker recognition", *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5329–5333, IEEE, 2018.

33. Zhu, Y., T. Ko, D. Snyder, B. Mak and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification", *Proc. Interspeech*, Vol. 2018, pp. 3573–3577, 2018.

34. Novotnỳ, O., O. Plchot, P. Matejka, L. Mošner and O. Glembek, "On the use of X-vectors for Robust Speaker Recognition", *Proceedings of Odyssey*, Vol. 2018, pp. 168–175, 2018.

35. Bhattacharya, G., M. J. Alam and P. Kenny, "Deep Speaker Embeddings for Short-Duration Speaker Verification.", *Interspeech*, pp. 1517–1521, 2017.

36. Liu, Y., Y. Qian, N. Chen, T. Fu, Y. Zhang and K. Yu, "Deep feature for text-dependent speaker verification", *Speech Communication*, Vol. 73, pp. 1–13, 2015.

37. Heigold, G., I. Moreno, S. Bengio and N. Shazeer, "End-to-end text-dependent speaker verification", *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5115–5119, IEEE, 2016.

38. Zhang, S.-X., Z. Chen, Y. Zhao, J. Li and Y. Gong, "End-to-end attention based text-dependent speaker verification", *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 171–178, IEEE, 2016.

39. Chowdhury, F., Q. Wang, I. L. Moreno and L. Wan, "Attention-based models for text-dependent speaker verification", *arXiv preprint arXiv:1710.10470*, 2017.

40. Snyder, D., P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification", *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 165–170, IEEE, 2016.

41. Li, C., X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system", *arXiv preprint arXiv:1705.02304*, 2017.

42. Schroff, F., D. Kalenichenko and J. Philbin, "Facenet: A unified embedding for face recognition and clustering", *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

43. Zhang, C. and K. Koishida, "End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances.", *Interspeech*, pp. 1487–1491, 2017.

44. Szegedy, C., S. Ioffe, V. Vanhoucke and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning", *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

45. Zhang, C. and K. Koishida, "End-to-end text-independent speaker verification with flexibility in utterance duration", *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 584–590, IEEE, 2017.

46. Zhang, C., S. Ranjan and J. Hansen, "An Analysis of Transfer Learning for Domain Mismatched Text-independent Speaker Verification", *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pp. 181–186, 2018.

47. Valenti, G., A. Daniel, N. Evans, N. Semiconductors and F. Mougins, "End-to-end automatic speaker verification with evolving recurrent neural networks", *Speaker Odyssey*, 2018.

48. Jung, J.-W., H.-S. Heo, I.-H. Yang, H.-J. Shim and H.-J. Yu, "A complete end-to-end speaker verification system using deep neural networks: From raw signals to verification result", *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5349–5353, IEEE, 2018.

49. Nagrani, A., J. S. Chung and A. Zisserman, "Voxceleb: a large-scale speaker iden-

tification dataset", *arXiv preprint arXiv:1706.08612*, 2017.

50. Jung, J.-W., H.-S. Heo, I.-H. Yang, H.-J. Shim and H.-J. Yu, "Avoiding speaker overfitting in end-to-end dnns using raw waveform for text-independent speaker verification", *extraction*, Vol. 8, No. 12, pp. 23–24, 2018.

51. Wan, L., Q. Wang, A. Papir and I. L. Moreno, "Generalized end-to-end loss for speaker verification", *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4879–4883, IEEE, 2018.

52. Hinton, G., L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury *et al.*, "Deep neural networks for acoustic modeling in speech recognition", *IEEE Signal processing magazine*, Vol. 29, 2012.

53. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

54. Haykin, S. S., S. S. Haykin, S. S. Haykin, K. Elektroingenieur and S. S. Haykin, *Neural networks and learning machines*, Vol. 3, Pearson education Upper Saddle River, 2009.

55. LeCun, Y., Y. Bengio and G. Hinton, "Deep learning", *nature*, Vol. 521, No. 7553, p. 436, 2015.

56. Rumelhart, D. E., G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors", *Cognitive modeling*, Vol. 5, No. 3, p. 1, 1988.

57. Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

58. Duchi, J., E. Hazan and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, Vol. 12, No. Jul, pp. 2121–2159, 2011.

59. Tieleman, T. and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", *COURSERA: Neural networks for machine learning*, Vol. 4, No. 2, pp. 26–31, 2012.

60. Povey, D., A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer and K. Vesely, "The Kaldi Speech Recognition Toolkit", *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

61. Seide, F. and A. Agarwal, "CNTK: Microsoft's open-source deep-learning toolkit", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2135–2135, ACM, 2016.