SPEAKER ADAPTED SPEECH SYNTHESIS WITH DEEP NEURAL NETWORKS

by

Miraç Göksu Öztürk

B.S., Computer 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 Computer Engineering

Boğaziçi University

2018

SPEAKER ADAPTED SPEECH SYNTHESIS WITH DEEP NEURAL NETWORKS

APPROVED BY:

Assoc. Prof. Arzucan Özgür .....
(Thesis Supervisor)

Assoc. Prof. Cenk Demiroğlu.....
(Thesis Co-supervisor)

Prof. Tunga Güngör .....

Assoc. Prof. Erhan Öztop .....

DATE OF APPROVAL: 19.10.2018

ACKNOWLEDGEMENTS

During the 3-year period of my MS study, I took countless tips and advises from my professors, friends, family and colleagues. Above all, I must thank firstly and deeply to my two advisers Arzucan Özgür and Cenk Demiroğlu although I cannot be thankful enough.

I thank Arzucan Özgür from the bottom of my heart for her incredible patience, her infectious and assuring optimistic attitude and surely her valuable lessons both on academic and non-academic topics. She is one of the most, if not the most, genuine, decent and caring people I have ever met and she is a true scientist in every sense of the word. She has been very sympathetic and solution oriented every time I showed up with a problem, which has been a huge support that I could not have replaced. After all, she introduced me to Cenk Demiroğlu when I wanted to quit my job, which was literally the beginning of this research.

I also and most sincerely thank Cenk Demiroğlu to whom I will be thankful also for the rest of my life for his unbelievable guide and support. Aside from our regular meetings, thousands of mails, text messages and phone calls, he has been my landlord, he registered me to this TÜBİTAK project, made it possible for me to have a place at Özyeğin University's dormitory, he introduced me to several research scientists in USA or more briefly, he provided me with an extensive support in every possible way. He is a remarkably versatile and smart person with great cognitive skills and I think that is why I have had no problem whatsoever explaining and discussing problems with him. He has also his unique vision that is way beyond the academy which has been as important and inspiring to me as this study itself.

I thank my dear parents and sisters that have always been appreciative of the importance of education and always helped me both financially and psychologically.

I thank my jury members, Tunga Güngör and Erhan Öztop for their time and feedbacks.

After I quit my job after the first year of my MS study, I have been lucky enough to be funded by TÜBİTAK since the beginning of my research which gave me the flexibility and freedom to explore new ideas without worrying about my finance.

As a friend and a flatmate, I am very grateful for the effort and valuable advises of Yasin Özkanca who is exceptionally intelligent, positive and constantly joyful.

I am also in great debt to the past and current people of speech lab in Özyeğin University, Saeed Sarfjoo, Berkant Kepez, Okan Ulusoy, Eray Eren, Aydın Emre Güzel, Merve Nur Ekmekci, Ahmad El Sayed, Ahad Aghapour and the people of AI lab in Boğaziçi University, Mert Tiftikci, Gönül Aycı, Arda Çelebi, Çağıl Uluşahin, Hakime Öztürk.

I would also like to thank my dear friends, Gökhan Esassolak, Ersel Gider, Burak Yılmaz, Hamdi Erkut, Tonguç Çataklı, Onur Torna, Uğur Kalkan, Gülcihan Büyükdemircioğlu, Hesna Doğanel Kalkan, Anıl Yolbüken, Doğa Bayram, İrfan Söğüt, Yılmaz Görgülü, Serhat Okan, Talha Ongun, Meliha Çiftçi and Turna Demirci.

Lastly, I thank my previous colleague Oytun Türk who has been a great team player and also very kind and supportive to me. Additionally, I am very grateful for the opportunity given by the nicest employers in the world, Mustafa Elbir and Fatih Çakmak.

ABSTRACT

SPEAKER ADAPTED SPEECH SYNTHESIS WITH DEEP NEURAL NETWORKS

Text-to-speech (TTS) systems have been an assisting technology since the 1970s. Although commercial use has begun decades ago, synthetic speech quality is still not as good as recorded speech. One particular subject of this field focused by this study is the speaker adaptation in TTS systems. Speaker adaptation is the task of modifying a given TTS model such that the modified model synthesizes speech samples with the voice characteristic of a desired speaker. In this study, deep neural network (DNN) based novel speaker adaptation techniques incorporating transfer learning methods are presented. We replaced the high dimensional speaker embeddings with few dimensional vectors using clustering methods. Objective results indicate significant improvement to the adaptation performance compared to baseline techniques in addition to a significant drop in the number of parameters. The second aspect of this study is the speaker adaptation performed on DNN-based postfiltering methods. The subjective results show that the adaptation of postfiltering increases the similarity of synthetic speech to the desired speaker's voice although no significant improvement in quality is observed. The techniques proposed in this study are independent of the choice of the DNN architecture and speaker embedding, thus, can be extended and used for experiments of relevant fields such as speech recognition in the future.

ÖZET

DERİN YAPAY SİNİR AĞLARI KULLANAN KONUŞMA SENTEZİ SİSTEMLERİNDE KONUŞMACIYA UYARLAMA

Metinden-konuşma (TTS) sistemleri, 1970'lerden beri yardımcı bir teknoloji olmuştur. Ticari kullanım on yıllar önce başlamış olmasına rağmen, sentetik konuşma kalitesi hala kayıtlı konuşma kadar iyi değildir. Bu çalışmanın odaklandığı konulardan biri, TTS sistemlerinde konuşmacı uyarlamasıdır. Konuşmacı uyarlaması, belirli bir TTS modelini, sesi arzu edilen bir konuşmacının ses karakteristiği ile sentezleyecek şekilde değiştirmektir. Bu çalışmada, transfer öğrenme yöntemlerini içeren derin sinir ağı (DNN) tabanlı yeni konuşmacı uyarlama teknikleri sunulmuştur. Kümelenme yöntemlerini kullanarak çok boyutlu konuşmacı temsil vektörlerini birkaç boyutlu vektörlerle değiştirdik. Nesnel sonuçlar, parametrelerin sayısında önemli bir düşüşe ek olarak, başlangıç performansına göre uyarlamada belirgin iyileşme olduğunu göstermektedir. Bu çalışmanın ikinci yönü, DNN tabanlı post filtreleme yöntemleri üzerinde gerçekleştirilen konuşmacı uyarlamasıdır. Öznel sonuçlar, postfiltre uyarlanmasının, sentetik konuşmanın istenen konuşmacının sesine benzerliğini arttırdığını, ancak kalitede önemli bir iyileşmenin gözlenmediğini göstermektedir. Bu çalışmada önerilen teknikler, DNN mimarisinin ve konuşmacı temsil vektörlerinin seçiminden bağımsızdır, bu nedenle, ileride konuşma tanıma gibi ilgili alanların deneyleri için genişletilebilir ve kullanılabilir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbf{0}$	Vector of zeros
$\mathbf{0}_{DxD}^{(t)}$	t^{th} D-by-D zero matrix
$\mathbf{1}$	Vector of ones
ap	Output of discriminator
A^{-1}	inverse of A
A^T	Transpose of A
cpf	Postfiltered cepstral features
c_t	Cepstral feature vector at time step t
$c_t(i)$	i^{th} dimension of cepstral features
$c(d,t)$	d^{th} mel-cepstral coefficients of reference signals in the time frame t
$\hat{c}(d,t)$	d^{th} mel-cepstral coefficients of estimated signals in the time frame t
d_i	Input dimension
d_o	Output dimension
e_i	Error for the time step i
E_L	Expected value of L
f	Arbitrary linear function
g	Activation function
$\mathbf{I}_{DxD}$	D-by-D identity matrix
$L_+^{(1)}$	Upper limit of delta context weights
$L_-^{(1)}$	Lower limit of delta context weights
$L_+^{(2)}$	Upper limit of delta-delta context weights
$L_-^{(2)}$	Lower limit of delta-delta context weights
L_{BCE}	Binary cross entropy loss
L_{MSE}	Mean squared error loss
$\mathcal{L}_{ADV}$	Adversarial loss
$\mathcal{L}_{PF}$	Loss function of postfilter
o_t	Generated output vector at time step t

p	probability density function
pn	Phoneme information
st	State information
$w^{(m)}(0)$	m^{th} scalar weight
$w^{(m)}(1)$	m^{th} scalar delta context weight
$w^{(m)}(2)$	m^{th} scalar delta-delta context weight
$x[t]$	x signal in time domain
$X[t]$	x signal in time domain
Δc_t	Delta feature vector of cepstral feature vector at time step t
$\Delta^2 c_t$	Delta-delta feature vector of cepstral feature vector at time step t
μ	Mean vector
$\hat{\mu}$	Estimated mean vector
$\sigma(x)$	Sigmoid of x
$\hat{\sigma}$	Estimated variance vector
Σ	Covariance matrix
$\|A\|$	Determinant of A
$cos(x)$	Cosine of x
$\frac{d}{dx}$	Derivative with respect to x
$\frac{\partial}{\partial x}$	Partial derivative with respect to x
$\lim_{x \to \infty}$	Limit when x goes to infinity
log	Logarithm base 10
$log(L\|X)$	Log-likelihood with respect to X
ln	Natural logarithm
$\sum$	Summation
$tanh(x)$	Hyperbolic tangent of x
$\|v\|_2$	L2 norm of vector v

LIST OF ACRONYMS/ABBREVIATIONS

ADV	Adversarial Training
ANN	Artificial Neural Network
BA	Baseline Adaptation
BAP	Band Aperiodicity
BBAM	Bernoulli Bidirectional Associative Memory
BCE	Binary Cross Entropy
BPTT	Backpropagation Through Time
CAT	Cluster Adaptive Training
CLSI	Cluster-based Initialization
DBN	Deep Belief Networks
DFT	Discrete Fourier Transform
DL	Deep Learning
DNN	Deep Neural Network
DSP	Digital Signal Processing
EM	Expectation Maximization
FF	Feedforward
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
HTS	HMM-based Speech Synthesis System
LF0	Log of Fundamental Frequency
LHUC	Learning Hidden Unit Contributions
LSTM	Long Short Term Memory
MCD	Mel-Cepstral Distortion
MGE	Minimum Generation Error
MLLR	Maximum Likelihood Linear Regression

MLPG	Maximum Likelihood Parameter Generation
MOS	Mean Opinion Score
MSE	Mean Squared Error
NLP	Natural Language Processing
NN	Nearest Neigbor
OCR	Optical Character Recognition
PESQ	Perceptual Evaluation of Speech Quality
PF	Postfiltering
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SI	Speaker Independent
SPSS	Statistical Parametric Speech Synthesis
SVM	Support Vector Machine
TANH	Hyperbolic Tangent Function
TTS	Text To Speech
TVS	Total Variability Space
UBM	Universal Background Model
Vocoder	Voice Coder
VODER	Voice Operating Demonstrator
VUV	Voiced/Unvoiced Information
WSJ	Wall Street Journal

1. INTRODUCTION

1.1. What Are Text-To-Speech Systems

A Text-To-Speech (TTS) system reads any text as input, recognizes the words in the text and predicts the appropriate pronunciations of the words and the prosodic features according to a predetermined set of linguistic rules [1] in order to eventually generate an utterance corresponding to the text [2]. The model used for relation between linguistic features and phonetic information to prosodic and acoustic features is crucial for high quality sound output and building the components of the system is only possible by a proper integration of methods from different fields. On one end of a TTS system, a given text needs to be analyzed and the correct pronunciations of units in the text should be determined. This part of the system is called front-end and these front-end processes are handled by the methods of Natural Language Processing (NLP) and Linguistics. In addition to that, the system must also include Digital Speech Processing (DSP) techniques that are utilized for processing speech data. Finally, the methods that model the relation between text and speech must be built. In this work, these methods are composed of Deep Learning (DL) techniques or more specifically Deep Neural Networks (DNN).

The most important aspect of a TTS system to focus is the synthetic speech's quality. In the context of speech processing, a speech's quality is defined by its naturalness and intelligibility. In other words, the generated voice must be clear and sound human-like. According to Taylor (2009), there is a strong correlation between user satisfaction and naturalness of speech [3]. Robotic speech is frowned upon and considered annoying as shown by user experience [3].

1.2. A Short History of Synthetic Speech

The idea of producing sounds to finally synthesizing speech arose in a letter written by Leonhard Euler in 1773, questioning how the human speech is created by

vocal organs. Thereafter, a prize for this challenge was declared in St. Petersburg which was claimed by a German physicist Christian Gottlieb Kratzenstein, who was also a doctor and an engineer, explaining the emergence of vowels in a vocal tract [4]. A more complex "speaking machine" was built by a Hungarian inventor Wolfgang von Kempelen whose work paved the way for a better theory of speech articulation [5]. Von Kempelen was mainly famous for his fake chess-playing machine that includes an actual man hidden inside, which was later exposed. His fake invention discredited his other works causing his "speaking machine" to stay underappreciated. Kempelen's machine was later improved in 19th century by a British scientist Charles Wheatstone [5].

An electronic analogue of speech synthesizer modeling vocal organs was first proposed by an American astrophysicist John Quincy Stewart in 1922 [6]. The first synthetic speaker, named Voice Operating Demonstrator (VODER), was presented in 1939 at New York's World Fair by Dudley *et al.* (1939) [7]. The first TTS system was introduced in Japan by [8], [9]. Although it was invented by the Japanese, the system was designed for the English language and the generated speech was fairly intelligible but poor in naturalness [5]. The earliest reading machine for the visually impaired, named The Kurzweil Reading Machine, was introduced in 1976, by Raymond Kurzweil [10].

1.3. Challenges

Text form of natural languages are in general composed of symbols or letters. Correct forms and pronunciations of words and phrases are determined based on preferences, thus, cannot be boiled down to simple or consistent patterns. The correct pronunciation of letters and syllables may vary depending on their position in a word, accent of speaker, syntactic and semantic characteristics of the word etc. There can also be more than one correct pronunciations of a word. For example, "7:15pm" can be read as "seven fifteen PM", "quarter past seven PM", "quarter past nineteen", "fifteen past nineteen" and so on. All of these are correct choices but expectations of users must be taken into consideration [3]. An announcement system such as those in stations must be precise and explicitly state whether it is AM or PM.

One other challenge is the disambiguation of homographic (i.e. words that are spelled the same but pronounced differently) words, namely, "bow", "project", "record" etc. The disambiguation process may require part of speech tagging as well as the correct sense of the word. Even so, there might be derivative words such as "gif" whose pronunciation has been subject to a famous controversy of internet over whether "g" at the beginning should be read as "g" as in "giraffe" or "g" as in "graphics".

Intelligibility is not the major problem of TTS systems since 1970s. Most TTS systems generate widely intelligible voices by keeping the variance low in the output [3]. However, contemporary system's main concern is the naturalness with no compromise on the intelligibility.

A relatively less crucial concern regarding the quality of the output is the proper use of intonation and rhythm (i.e. prosody) associated with the given text. Nonetheless, it is not an easy task considering that these are hardly encoded in text.

1.4. Synthesis Techniques

There are mainly two TTS approaches: concatenative and statistical approach. There are pros and cons to both sides. Concatenative methods produce natural-sounding speech and can handle smoothing joins between speech units whereas statistical approaches are less resource-consuming, cover a broader range of speech characteristics and are able to adapt a base model to an arbitrary speaker.

A widely common concatenative technique is unit selection which is still one of the most dominant speech synthesis method [11] in practice. Unit selection is the concatenation of waveform units from large and single-speaker databases [12]. Since the units go through very little or no modification during synthesis, this method generates the most natural-sounding speech. On the other hand, it heavily relies on the database coverage leading to high resource consumption as well as space requirement which is called the footprint in the context of TTS. Moreover, the generated speech is limited to the combination of speech units in the database and cannot include a new unit.

While the best examples of unit selection are better than the best examples of statistical parametric synthesis [11], HMM-based synthesis was able to provide intelligible speech [13], [14] in the corpus-based speech synthesis challenge of Blizzard.

Acoustic units represent linguistic features and are used to construct voices in concatenative speech synthesis. However, a large amount of acoustic units is hard to collect and store. As a consequence, new methods that can generate speech by using a smaller database and without compromising the quality of the speech were searched. HMM-based speech synthesis became one of the main subjects and is commonly used in both academic and commercial applications. In the 2012 Interspeech conference, which is one of the biggest international conferences on speech information processing, around 76% of papers were focused on statistical speech synthesis approaches [15]. Unlike the conventional concatenative approach, HMM-based speech synthesis models the spectrum, pitch and state duration of speech simultaneously in a unified framework [16] and provides a large range of variance in the output voice characteristic. HMM produces speech parameters which are then converted to an actual sound by a parametric vocoder. A vocoder (voice encoder) is a crucial component of a TTS system. It is used to extract a feature representation from which the original speech can be reconstructed. The quality of the vocoder is one of the constraining factors to the quality of the speech and it may even be the bottleneck of the system. However, there are promising vocoders that can parametrize and reconstruct speech without significant deterioration such as STRAIGHT [17].

The major drawback of a Statistical Parametric Speech Synthesis (SPSS) system is the quality of the speech. Aside from vocoding and oversmoothing, accuracy of the acoustic model, which handles the complex mapping between the linguistic and acoustic features, is the culprit of this drawback [11]. In HMM-based speech synthesis systems, as its name implies, the hidden Markov techniques are utilized for the acoustic model which has been recently replaced by neural networks. Even though neural networks have been known and used since the 90's, [18] modern applications of neural networks can have more hidden layers and can be trained on more data with the help of hardware improvements and algorithms, which led to more complex and sophisticated network

structures to be effectively implemented (e.g. computation on graphic processors [19]).

In the study of Zen *et al.* (2013), it is aimed to address the HMM's incapability in modeling context dependencies and it is shown that the DNN approach as an acoustic model outperformed the HMM-based approach when the number of parameters are similar [20]. Ling *et al.* (2013) attempted to alleviate the over-smoothing problem of HMMs with deep belief networks (DBN) as a replacement for Gaussian distribution in representing the low-level spectral envelopes and improved the quality of the synthesized speech [21]. The spectrum and F0 features are modeled simultaneously with DBN in the work of Kang *et al.* (2013) and it is demonstrated that there is a decrease in distortion for the spectrum generated from DBN [22]. In the study of Fernandez *et al.* (2013), a hybrid model integrating parametric and non-parametric approach was used where a neural network structure was employed for feature extraction from input in order to be used as an input to a non-parametric model [23]. Lu *et al.* (2013) applied DNN as a model between continuous vector-space representations of linguistic context and probability distributions over acoustic features despite the fact that in this case, it didn't outperform HMM systems [24]. In the work of Qian *et al.* (2014), a DNN is utilized for converting text features to acoustic features (V/U, F0 and LSP) and superiority of the DNN to HMM baseline was shown by both objective and subjective measurements [25]. Consequently, these encouraging results paved the way for the future neural network research on speech synthesis and proved the great potential for yielding natural sounding speech outperforming conventional HMMs.

1.5. Speaker Adaptation In Speech Synthesis

Speaker adaptation in the context of TTS is the process of adjusting a model for an arbitrary target speaker. The resulting model is then expected to produce utterances according to the voice characteristic of the target speaker. The adaptation is generally based on a small amount of speech data taken from the target speaker. In spite of the fact that concatenative approaches generate high-quality speech, they are not tailored to perform this task since they do not offer a flexible model. A huge database is required to be able to produce different speaking styles or expressions and

storing units for each distinct speaker or emotion is impractical and costly. Hence, statistical approaches in speech synthesis are the primary choice for this task due to its highly flexible nature compared to conventional approaches such as unit selection speech synthesis. Statistical approaches are theoretically able to adapt to different scales of speaking styles and emotions, since they use the adaptation data at training to tune their models.

A speech synthesis system that has the ability to adapt to an arbitrary speaker with small amount of adaptation data is a quite appealing feature for applications that use speech as input and/or output, especially for embedded systems with modest storage and computation capabilities. Fast adaptation also eases the burden of recording long duration of speeches, thereby reducing cost of adaptation by demanding a very small sized sample (a couple of minutes).

Fast speaker adaptation systems can provide cheaper solutions in fields where hiring a voice artist is not affordable, such as talking toys, cartoon characters, dubbing of movies and audio theaters to a limited extent or audio stories. Area of application of speaker adaptation can be expanded as the adaptation quality increases in time, since it offers a wide range of speaking styles and emotions with a small footprint.

There are many studies on speaker adaptation for speech synthesis using statistical approaches. However, very few of them used deep neural networks (DNN). As a result, very little is known about the possible contributions of DNN to this field and will be examined in detail in the rest of this research.

In this study, shortly, we proposed new alternatives to the conventional speaker codes used in speaker adaptation and also showed the advantage of using post-filters as an adaptation approach as well as a post-filter mechanism trained with an adversarial network.

2. MOTIVATION

2.1. Applications of TTS

There are a number of areas where a TTS system is employed. Although a TTS system itself might hardly seem to address a common vital issue, it becomes one of the essential components of an assistive technology when deployed with other complementary components. It can be fundamental for particular groups of people such as those who have learning disabilities (e.g. dsylexia), literacy problems, or visual impairment. For instance, for people whose voice-related disabilities are due to motor or sensory neurons rather than mental diseases, TTS systems, which can convert textual information to speech, can be very helpful. The famous physicist Stephen Hawking had been one example of these people who is a continuous user of such a system even for presenting lectures until his death. Aside from people with hearing/speaking, or visual disabilities, people for whom reading can be an exhausting experience, can also make use of TTS applications together with an optical character recognition (OCR) systems to read a random text. Some of these applications are casually used in trains, train stations, airports, screen readers in computers/mobile phones, automotive navigation systems, or other fields where recording of a human speech is costly. Additionally, TTS is also applied in telecommunication services, language education, talking books and toys, and man-machine communication [2]. For instance, an educational robot with a TTS system embedded would be available anytime on demand for tasks such as teaching spelling or pronunciation which is impossible for a human teacher.

2.2. Applications of Speaker Adaptation in TTS

Although a TTS system is expected to read any text input, the performance of the system is likely to be better when the input comes from a specific and limited domain. For example, a system that generates only the pronunciation of numbers and dates is relatively easier to build than a general-purpose system. For example, a TTS module deployed in a train station's audio announcement system is expected to produce only

station names, route numbers, emergency messages etc. For these scenarios, it may be challenging to collect enough data to build a high quality model. This issue can be addressed by using a speaker-adapted model trained with limited relevant data. In cases where the utterances to be produced are known in advance, a trivial look-up system will be sufficient. However, even the combination of dates and numbers alone can easily inflate the number of possible utterances.

A TTS system's one aspect leading to prohibitive costs is hiring a voice artist. Adaptation with small dataset allows low-budget start-ups or enthusiasts to create their own TTS module with a desired speaker's voice instead of a few openly available voices. Thus, it becomes more convenient and practical to build diverse voices especially for recreational purposes such as voices in video games or cartoons.

Adaptation of a TTS model to an arbitrary speaker can be considered as a special case of transfer learning and an analog of style transfer in the domain of speech processing. Hence, techniques applied here can be transferred to other domains as well. Additionally, instead of from one speaker to another, it can be an adaptation from neutral speech to emotional (e.g. happy, sad, excited etc.) speech provided that the voice still sounds as if it is uttered by the same speaker. This type of model transformation might also provide us with a better understanding and analysis of speech.

3. RELATED WORK

Deep learning applications has become mainstream in a short period of time, not only in the area of speech processing but also in many different fields such as Natural Language Processing, Machine Translation, Image Classification etc. However, that has not been always the case. Although the important concepts in Deep Learning such as backpropagation and recurrent structures have been known for over a decade, they were not the popular choices until the advances in hardwares. Especially general purpose GPU computing sped up the backpropagation by parallelizing the matrix multiplications and the gradient calculations.

Adaptation of a TTS system for an arbitrary speaker with limited data is a difficult task. Especially when the number of parameters in the model is high, good generalization performance based on a few utterance is hard to achieve. There were several common techniques before deep learning methods had prevailed among others. In the work of Mohammadi *et al.* (2014), Eigenvoice technique was used. Even though this framework was successful in speech recognition systems and is also mathematically meaningful, the synthesized speech has perceptual artifacts after adaptation [26]. The proposed system by Mohammadi and Demiroglu (2013) focused on the input model rather than the acoustic model itself for each target speaker [27]. The approach provides the closest reference speaker to the target speaker to be used in the speaker adaptation process. Mohammadi and Demiroglu (2013) integrated both hybrid and nearest-neighbor/cluster approach and made an improvement while it is still outperformed by speaker-dependent methods [27].

In the study of Tamura *et al.* (1998), phoneme HMMs were employed as synthesis units without taking prosodic features into consideration such as pitch contour and duration [28]. Maximum likelihood linear regression (MLLR) method is used to adapt the voice characteristics to the target speaker. HMM-based speech synthesis suggested by Tamura *et al.* (2001) continued the task and adapted other prosodic features to the target speaker by training on a few utterances [29]. Multi-space probability distribution

is used for modeling spectrum and pitch and again MLLR is applied for adapting to an arbitrary speaker. In addition, another algorithm on HMM-based speech synthesis was proposed by Yamagishi *et al.* (2009) [30]. It compared different configuration choices and algorithms in training and model setup in speaker adaptation for HMM-based speech synthesis. Additionally, a hidden semi-markov model was proposed in Yamagishi *et al.* (2007) [31]. They worked on another prosodic feature, phone duration, by trying to normalize its distributions together with other prosodic features such as spectrum and F0. However, one difficulty is that the mean of the duration distribution can be negative for the reason that the phone duration distributions are assumed to be Gaussian.

Cluster adaptive training (CAT) in the study of Gales (1999) is one of the effective models which is also used by Zen *et al.* (2012), Latorre *et al.* (2012) and Wan *et al.* (2012) [32–35]. CAT algorithm is defined to be a simple extension of speaker clustering where a linear interpolation of all the cluster means is utilized for the target speaker instead of choosing one cluster [32]. Zen *et al.* (2012) proposed a framework in which language characteristics are factorized as well as speaker characteristics in order to make language adaptation possible too [33]. In the work of Latorre *et al.* (2012), a set of clusters was used for representing each of the speaker-specific features and expressive features such as emotions [34]. By changing the weights associated with each set independently, obtaining intermediate degrees of both emotions and speaker identities in synthetic speech is possible. However, it is claimed that expressiveness of the model still needs improvement. In the study of Wan *et al.* (2012), speaker adaptive and speaker dependent approaches are compared in terms of quality of speech [35]. It is noted that the CAT approach produced more preferable results than speaker dependent models.

As with the speech synthesis with deep learning models, the area is still open to exploitation. Studies on statistical speech synthesis with recurrent neural networks are not uncommon [36, 37]. In the work of van den Oord *et al.* (2016), the training is directly performed on 44 hours of raw audio waveform from 109 different speakers [38]. Although the model is capable of producing natural-sounding audio, the proposed

model is not practical since the generation process is computationally expensive. A faster method is proposed by Wang *et al.* (2017) [39]. The suggested network is composed of various types of layers such as attention-based encoder-decoder, convolutions, highway networks, bi-directional RNNs and residual connections. The resulting system is claimed to be robust to spelling errors and sensitive to punctuations. In the work of Arik *et al.* (2017), the TTS system is unconventionally divided into components where each component is modeled by a different neural network and the components together form an end-to-end system [40]. With the proposed system, Arik *et al.* (2017) aims to remove the feature engineering process which requires domain specific knowledge [40]. Moreover, errors in this process are likely to propagate and affect the overall accuracy of the system.

Aside from speech synthesis, there are very few studies on speaker adaptive speech synthesis using recurrent neural networks or any other neural network structure except for the papers that worked on speech recognition. DNN-based TTS for speaker adaptation is applied for the first time in 2015 [41]. State-of-the-art speaker adaptation techniques that are used for speech recognition are investigated in the context of speaker adaptive speech synthesis. As a DNN framework, feedforward structure is employed. As in HMM-based techniques, average voice models are constructed. Adaptation is later performed by retraining the networks for the target speaker. It is shown that the DNN approach outperformed HMM according to objective closeness of the generated speech to the original speech. However, in this work input labels are not fully utilized and the default quinphone context is simplified to triphone context. Additionally, generated parameters are fed directly to vocoder for synthesis. Skipping the post-filtering process which ignores the first and second time derivatives can cause unnatural flat areas in the generated parameters. Silent parts are also not removed from the data before training which incorrectly affects the model. Fan *et al.* (2015) proposed a network structure in which there are shared hidden layers trained by every speaker's samples and there are also exclusive layers that are only updated by its speaker's samples but note that there is no recurrent nodes [42]. The structure is in Figure 3.1. In addition, an adaptation method is proposed where shared hidden layers are borrowed from this multi-speaker model. Then, a new layer is added on

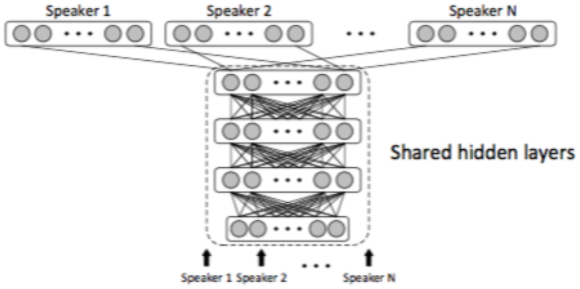

Figure 3.1. Figure is taken from Fan *et al.* (2015) [42]

top of the shared layers and the new layer is trained according to a target speaker while shared layer weights are fixed. This adaptation method outperformed speaker-dependent DNN-based baseline approach in subjective results although the baseline approach produced better objective results. This approach is called multi-task learning. It is based on the assumption that the samples have to share some information and also somewhat differentiate from each other, which are, respectively, represented by the lower shared layers and higher individual layers in the structure. The same task was later carried out with the RNN extension [43]. Another multi-task learning approach was adopted by Chen *et al.* (2014) where a bottleneck layer is placed in the network assuming that it encodes binary cluster information and the multi-task model is applied later on an another independent network with the information coming from the first network's bottleneck layer encoding [44].

In the work of Wu *et al.* (2015), DNN-based framework is employed together with combination of different adaptation techniques such as i-vector, learning hidden unit contribution (LHUC) and feature space transformation [45]. It is demonstrated that combination of the three methods with DNN-based speaker adaptation performed significantly better than HMM baseline. An unsupervised multi-speaker adaptation technique is proposed by Fan *et al.* (2016) with the framework used in the previous work by Fan *et al.* (2015) [42, 46]. However, the performance is not as good as its baseline supervised counterpart even though the gap is not significant. Different recurrent network structures are compared in modeling speech and the contribution of inner gates of cells are investigated by Wu *et al.* (2016) [47]. Although there are many works

on the analysis of RNNs from different aspects, the work of Wu *et al.* (2016) is in the context of TTS [47].

Another aspect of this study is regarding the improvement and adaptation in postfiltering the generated speech parameters. According to Jokinen *et al.* (2012), postfiltering is the post-processing of speech signals in order to increase the intelligibility and the quality of speech not only in speech synthesis but also in other speech-related technologies [48]. Older postfiltering techniques were applied on generated parameters of Hidden Markov Model (HMM)-based TTS methods. In the study of Tokuda *et al.* (2000), the postfiltering task is done by a forward-backward algorithm assuming that the observations of the HMM is the generated speech parameters and the state sequence of the HMM is only partly known [49]. Toda and Tokuda (2007) give both a conventional parameter generation method given a sequence of parameters and their dynamic features (first and second derivatives) and also an iterative method that additionally includes the global variance [50]. Although Toda and Tokuda (2007) worked on HMM-based methods, the given conventional method can be applied to the predictions of the DNN-based methods [50]. In the work of Chen *et al.* (2014), a DNN-based postfiltering model was proposed while the DNN does not consist of contemporary feedforward layers but of Restricted Boltzman Machine (RBM) and Bernoulli Bidirectional Associative Memory (BBAM) and it was also designed for HMM-based speech synthesis [51]. In a more recent work by Muthukumar *et al.* (2016), a plain RNN architecture was introduced and jointly trained with the main acoustic model [52]. A more complex Generative Adversarial Network (GAN)-based postfilter architecture was shown by Kaneko *et al.* (2017) in which the synthetic parameters were fed to the generator network of the GAN together with random noise vectors as input [53]. The synthetic parameters were also taken by the intermediate layers as input and the generator was trained based on the discriminator's loss function.

4. DATASET AND EVALUATION

4.1. Dataset

All experiments were conducted with the Wall Street Journal (WSJ) speech database. 154 features were extracted at a sampling rate 16KHz and 5 ms frame rate 25 Mel-Generalized Cepstrum Coefficients (MGC), 1 log of fundamental frequency (LF0) and 25 Band Aperiodicity (BAP) together with their delta and delta-delta features. Additionally, an unvoice/voice binary feature was appended to represent voicing information.

There are 135 speakers, each of them having different number of utterances with between 60-1200. Aside from those 135, there are also 21 speakers, each of them having more than 100 utterances with different transcriptions in the dataset. The utterances from the 135 speakers will be used for the construction of the speaker-independent (average-voice) model and the rest will be used during the adaptation process.

Utterances are in raw audio format and around 6-8 seconds whose some parts are silent. For each utterance, there is a label file, which contains the phonetic and linguistic context features of the corresponding text. Label files are extracted from raw text with the help of Festival speech synthesis tool and prepared according to HTK label format [54]. A label file can include text features such as phoneme information, position of the phone, syllable position, part-of-speech tagging etc. Label files are the unprocessed input data of the system, whereas raw audio files are the unprocessed output data.

A 5-state HMM model was applied to model phoneme durations by using HTS 2.3 speech synthesis tool [55]. Among the total of 156 speakers, 135 of them -50 utterances for each- were assigned as reference speakers and used for training, whereas the test and adaptation processes were performed on the remaining 21 target speakers' data. We extracted 456 features from the input text, both binary and numerical, and the

features then were normalized into the range of [0.01 0.99] before training. However, the number of text features depends on the data set. The higher the number of features, the more extensive the model is. Yet, higher dimensional input features require more parameters to be learned. The predicted acoustic features are later fed to a vocoder [17] to produce a waveform.

Each label file has as many frames or time steps as its corresponding expected output and each frame contains both phonetic and linguistic information. Each frame is converted to a numerical representation, a vector, according to a language- and context-dependent label format. The HTK (The Hidden Markov Model Toolkit) template and a single example in HTK format is as below:

The HTK label template:

```
p1^p2-p3+p4=p5@p6_p7
/A:a1_a2_a3
/B:b1-b2-b3@b4-b5&b6-b7#b8-b9$b10-b11!b12-b13;b14-b15|b16
/C:c1+c2+c3
/D:d1_d2
/E:e1+e2@e3+e4&e5+e6#e7+e8
/F:f1_f2
/G:g1_g2
/H:h1=h2@h3=h4|h5
/I:i1=i2
/J:j1+j2-j3
```

A label example for the sentence 'I want to be a great engineer': where the implied information is defined by Zen (2006) and documented as below [56].

```
p1: the phoneme identity before the previous phoneme
p2: the previous phoneme identity
```

Figure 4.1. Figure of the label of the sentence 'I want to be a great engineer'

p3: the current phoneme identity

p4: the next phoneme identity

p5: the phoneme after the next phoneme identity

p6: position of the current phoneme identity
in the current syllable (forward)

p7: position of the current phoneme identity
in the current syllable (backward)

a1: whether the previous syllable stressed or not
(0: not stressed, 1: stressed)

a2: whether the previous syllable accented or not
(0: not accented, 1: accented)

a3: the number of phonemes in the previous syllable

b1: whether the current syllable stressed or not
(0: not stressed, 1: stressed)

b2: whether the current syllable accented or not
(0: not accented, 1: accented)

b3: the number of phonemes in the current syllable

b4: position of the current syllable in the current word (forward)

b5: position of the current syllable in the current word (backward)

b6: position of the current syllable in the current phrase (forward)

b7: position of the current syllable in the current phrase (backward)

b8: the number of stressed syllables before the current syllable

in the current phrase

b9: the number of stressed syllables after the current syllable
in the current phrase

b10: the number of accented syllables before the current syllable
in the current phrase

b11: the number of accented syllables after the current syllable
in the current phrase

b12: the number of syllables from the previous stressed syllable
to the current syllable

b13: the number of syllables from the current syllable to
the next stressed syllable

b14: the number of syllables from the previous accented syllable to
the current syllable

b15: the number of syllables from the current syllable to
the next accented syllable

b16: name of the vowel of the current syllable

c1: whether the next syllable stressed or not
(0: not stressed, 1: stressed)

c2: whether the next syllable accented or not
(0: not accented, 1: accented)

c3: the number of phonemes in the next syllable

d1: gpos (guess part-of-speech) of the previous word

d2: the number of syllables in the previous word

e1: gpos (guess part-of-speech) of the current word

e2: the number of syllables in the current word

e3: position of the current word in the current phrase (forward)

e4: position of the current word in the current phrase (backward)

e5: the number of content words before the current word
in the current phrase

e6: the number of content words after the current word
in the current phrase

```
e7: the number of words from the previous content word
    to the current word
e8: the number of words from the current word to the next content word
f1: gpos (guess part-of-speech) of the next word
f2: the number of syllables in the next word
g1: the number of syllables in the previous phrase
g2: the number of words in the previous phrase
h1: the number of syllables in the current phrase
h2: the number of words in the current phrase
h3: position of the current phrase in utterance (forward)
h4: position of the current phrase in utterance (backward)
h5: TOBI (Tones and break indices) end tone of the current phrase
i1: the number of syllables in the next phrase
i2: the number of words in the next phrase
j1: the number of syllables in this utterance
j2: the number of words in this utterance
j3: the number of phrases in this utterance
```

The list can be expanded or shrunk as desired. The full set or a subset of these linguistic features are converted into numerical values to be used as features for the desired model. Notice that extraction of such features is not language-independent and requires linguistic knowledge.

4.2. Pre-processing and Post-processing

Both input and output features are normalized before the training. This normalization process is required to avoid weight explosions as well as saturation and bias at activation functions. The selected activation function in all the three feed forward layers is hyperbolic tangent (TANH), which is zero-origined as seen in Fig 4.2. As the function goes to ∞ or $-\infty$, the value of the gradient function (Equation 4.1) tends to vanish and the maximum value is obtained when the input is around zero. This can

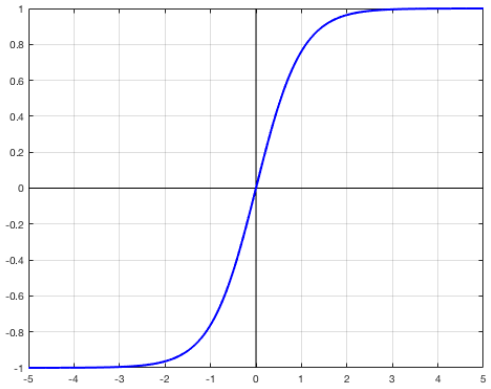

Figure 4.2. Hyperbolic Tangent graph

be seen both visually in Fig 4.2 and equation 4.2.

$$\frac{d}{dx}tanh(x) = 1 - tanh^2(x) \tag{4.1}$$

$$\lim_{x\to\infty} \frac{d}{dx}tanh(x) = \lim_{x\to\infty} 1 - tanh^2(x) = 0 \tag{4.2}$$

Therefore, it is necessary to z-normalize the data to avoid saturation by keeping it around 0 so that the gradients do not quickly converge to 0. Otherwise, the small gradients will dramatically slow down the training process since the updates are directly dependent on the gradients.

Input features have either binary values or numerical values in the range of $[0, 1]$ whereas output features have a larger variance. Label files and output files are normalized with 0 mean and 1 variance using the total mean and standard deviation vectors calculated according the training set before being fed to the network. Due to the fair assumption that input and output features come from a different distribution, global mean and global variance vector of input and output data are calculated independently of each other.

Z-normalization is the subtraction of the mean followed by the division by the standard deviation as in equation 4.3.

$$z = \frac{(x - \hat{\mu})}{\hat{\sigma}} \tag{4.3}$$

where

$$\hat{\mu} = \frac{\sum x}{n} \tag{4.4}$$

and

$$\hat{\sigma} = \sqrt{\frac{\sum (x - \hat{\mu})^2}{n}} \tag{4.5}$$

where x is a feature vector, n is the total number of samples (vectors), $\hat{\mu}$ is the mean vector and $\hat{\sigma}$ is the standard deviation vector.

After the training is finished and the model outputs a prediction, the raw output is de-normalized according to the saved global mean and variance in the pre-processing step. Then, maximum-likelihood parameter generation (MLPG) as given by Toda and Tokuda (2007) was used for generating the eventual output features from these predicted features [50]. Finally, the output features are sent to the vocoder in order to synthesize speech waveform. For the training case, single global mean and variance were used for normalizing all speakers' data while these were re-calculated for each speaker during adaptation.

4.2.1. Maximum-likelihood parameter generation (MLPG)

MLPG process is derived as follows: Let the output or prediction for a time step t be $o_t = [{c_t}^T, \Delta {c_t}^T, \Delta^2 {c_t}^T]^T$ where $c_t = [c_t(1), c_t(2), ...c_t(D)]^T$ is the static feature vector

and Δc_t and $\Delta^2 c_t$ are the dynamic feature vectors such that:

$$\Delta c_t = \sum_{m=L_-^{(1)}}^{L_+^{(1)}} w^{(m)}(1) c_{t+m} \tag{4.6}$$

$$\Delta^2 c_t = \sum_{m=L_-^{(2)}}^{L_+^{(2)}} w^{(m)}(2) c_{t+m} \tag{4.7}$$

We know that our acoustic features consist of time step vectors, hence, are represented as matrices. However, they are considered as one vector therefore they are flattened such as:

$$O = [o_1^T, o_2^T, ..., o_\tau^T]^T \tag{4.8}$$

and

$$C = [c_1^T, c_2^T, ..., c_\tau^T]^T \tag{4.9}$$

where τ is the number of time steps. Notice that C and O are not matrices but vectors of DT-by-1 and $3DT$-by-1. The linear relation between these two is:

$$O = WC \tag{4.10}$$

and

$$W = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_\tau]^T \tag{4.11}$$

$$\mathbf{w}_t = [\mathbf{w}_t^{(0)}, \mathbf{w}_t^{(1)}, \mathbf{w}_t^{(2)}]^T \tag{4.12}$$

$$\mathbf{w}_t^{(n)} = [\mathbf{0}_{DxD}^{(1)}, ..., \mathbf{0}_{DxD}^{(t-L_-^{(n)}-1)}, (-L_-^{(n)})\mathbf{I}_{DxD}, ..., w^{(n)}(0)\mathbf{I}_{DxD}, ..., (L_+^{(n)})\mathbf{I}_{DxD}, ..., \mathbf{0}_{DxD}^{(\tau)}]^T \tag{4.13}$$

where $\mathbf{0}_{DxD}^{(t)}$ is the tth D-by-D zero matrix and $\mathbf{I}_{DxD}$ is the identity matrix. $L_-^{(0)}$, $L_+^{(0)}$ and $w^{(0)}(0)$ were given as 0 by Toda and Tokuda (2007) [50]. In our experiments, we set the remaining parameters as $L_-^{(1)} = L_+^{(1)} = L_-^{(2)} = L_+^{(2)} = 1$ and $w^{(-1)}(1) = -0.5$, $w^{(0)}(1) = 0$, $w^{(1)}(1) = 0.5$ and $w^{(-1)}(2) = 1$, $w^{(0)}(2) = -2$, $w^{(1)}(2) = 1$.

We know that the log-likelihood of a given multivariate sample vector x with k random variables is

$$log(L) = -\frac{1}{2}(log(\Sigma) + (x-\mu)^T\Sigma^{-1}(x-\mu) + klog(2\pi)) \tag{4.14}$$

therefore, if plug O into 4.14

$$log(L|O) = -\frac{1}{2}(log(\Sigma) + (O-\mu)^T\Sigma^{-1}(O-\mu) + Dlog(2\pi)) \tag{4.15}$$

which becomes

$$log(L|O) = -\frac{1}{2}(log(\Sigma) + O^T\Sigma^{-1}O + O^T\Sigma^{-1}\mu + \mu^T\Sigma^{-1}O + \mu^T\Sigma^{-1}\mu + Dlog(2\pi)) \tag{4.16}$$

since Σ^{-1} is diagonal, $O^T\Sigma^{-1}\mu$ and $\mu^T\Sigma^{-1}O$ must be equal, thus, we can write

$$log(L|O) = -\frac{1}{2}O^T\Sigma^{-1}O + O^T\Sigma^{-1}\mu + K \tag{4.17}$$

where K can be considered as a constant since our objective is to find C that maximizes $log(L|O)$ and K is not a function of C. We take the partial derivative of $log(L|O)$ with respect to C and set it to 0.

$$\frac{\partial log(L|WC)}{\partial C} = 0 \tag{4.18}$$

then

$$-W^T\Sigma^{-1}WC + W^T\Sigma^{-1}\mu = 0 \tag{4.19}$$

and

$$C = (W^T\Sigma^{-1}W)^{-1}W^T\Sigma^{-1}\mu \tag{4.20}$$

where μ, Σ and W are of the shape $3DT$-by-1, $3DT$-by-$3DT$ and $3DT$-by-DT, respectively

4.3. Analysis of Speech Waveform

Vocoders (Voice Coder) initially has been designed for the transmission of speech via low bandwidth channels by encoding or compressing at one end and decoding or reconstructing at the other end. Typical speech signal contains highly redundant information considering the human perception of speech such as human ear's insensitivity to phase information of frequency components. The regular approach in modeling the production of speech is the source-filter model inspired by the vocal system of human where the energy or source is provided by the flow of air from lungs through the vocal cords and acoustically filtered by the vocal tract (lips, teeth and tongue) and the nose. The position of the vocal tract determines which frequencies to be amplified or weakened, hence, producing phonemes. Similarly, vocoders break down a given speech signal into two parts namely the spectral features (filter) and the excitation (source).

If the vocal cords are slightly open while speaking, the flowing air causes them to vibrate which creates a periodic sound wave and the sounds created as such are called the voiced sounds whose fundamental frequency is equal to the vibration rate of the vocal cords e.g. all the vowels. On the other hand, when the vocal cords are open, no periodic sound is produced and the sound is articulated by the turbulent air flowing through the vocal tract as well as the nose and the phonemes articulated in this fashion

are called the unvoiced sounds e.g. [s] as in September or any whispered sound. This phenomenon is also modeled by the excitation part of the vocoder by using a switching mechanism that chooses between a noise or a periodic sound as the source.

4.3.1. Spectral Analysis of Speech

The speech signals are known to be dynamic over time due to the continuous change in the position of vocal organs. Thus, the speech signals are analyzed within short intervals where the spectral features are assumed to be stationary. The entire utterance is divided into small intervals which is called windowing and extracted by the following equation.

$$y[n] = w[n]s[n] \tag{4.21}$$

s is the source signal and w is the windowing function. Each processed window is called a frame, the width of the window in milliseconds is the frame size and the distance between the left end of consecutive windows is the frame shift [57] as shown in Figure 4.3.

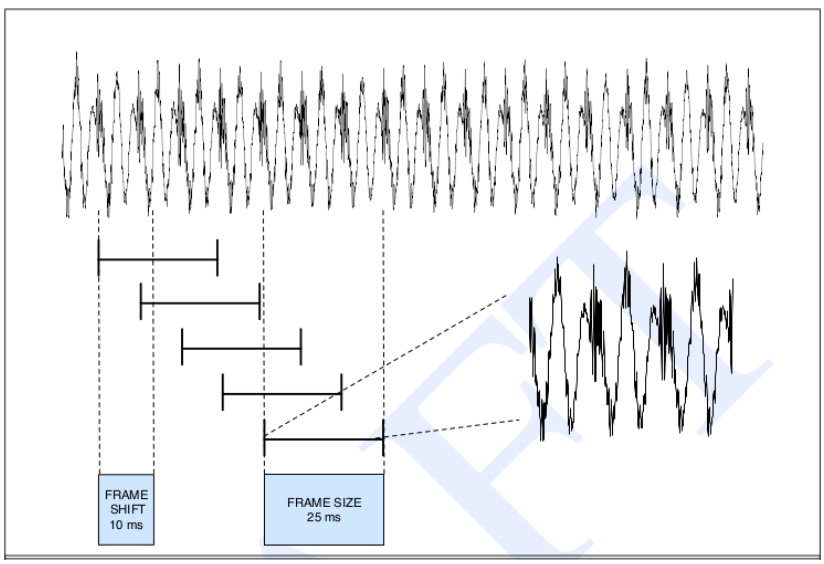

Figure 4.3. The figure shows the frame size and shift when they are 25ms and 10ms, respectively. (Taken from Martin and Jurafsky (2009) [57])

The most basic windowing function is the rectangular window which takes a region with no additional modification as below.

$$w[n] = \begin{cases} 1 & 0 \leq \text{n} \leq \text{L-1} \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

where L is the length of the window. However, the rectangular window might be problematic for the Fourier transformation since the signal has discontinuities at the edges of the window. This is mitigated by shrinking the magnitude on both sides and one common function to do that is hamming window whose equation is as below.

$$w[n] = \begin{cases} 0.54 - 0.46cos(\frac{2\pi n}{L}) & 0 \leq \text{n} \leq \text{L-1} \\ 0 & \text{otherwise} \end{cases} \tag{4.23}$$

The output signal after being applied both hamming and rectangular window to a cosine signal are shown in Figure 4.4.

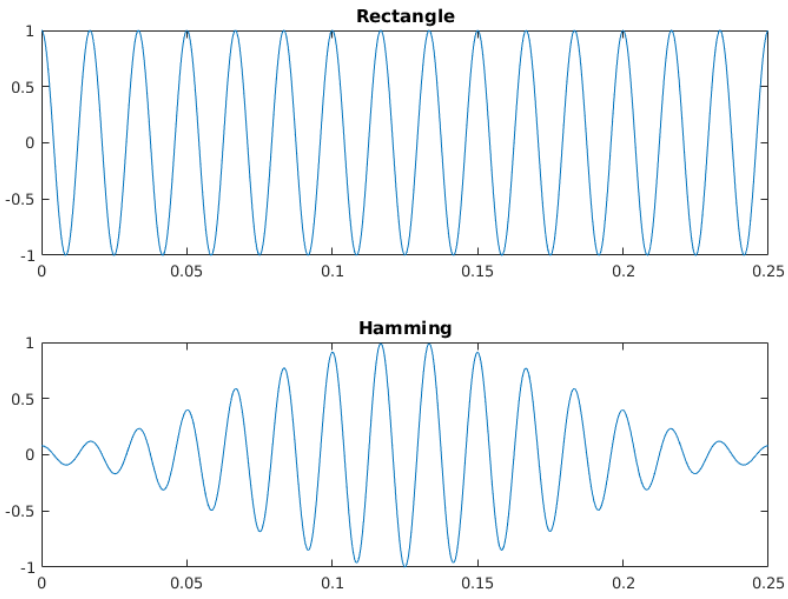

Figure 4.4. The difference between rectangular and hamming window is demonstrated on a simple cosine wave.

The following step is the extraction of the component frequencies' magnitudes. This is obtained by the discrete Fourier transform (DFT) whose input is the windowed signal. The DFT algorithm yields a complex number for each frequency component whose absolute value is the corresponding magnitude. We know that human auditory perception is insensitive to phase changes therefore the magnitude information will suffice for further analysis. The DFT equation for an input signal x of length N is given as below.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \tag{4.24}$$

One way to decompose a given signal into source and filter is the cepstrum (first four letters of spectrum are reversed) analysis which is simply the inverse Fourier transform of the log of the Fourier transform of the signal and described by the following equation:

$$c[n] = \sum_{n=0}^{N-1} log\left(\left| \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \right| \right) e^{-j\frac{2\pi}{N}kn} \tag{4.25}$$

Notice that we switched back to the time domain and the coefficients of the cepstrum are actually the samples belonging to the analyzed frame, which are called quefrency (reorganizing the letters of frequency). Initial coefficients (less than 30) carry the information about the articulated phoneme whereas higher coefficients can be examined for the pitch information [3]. In Figure 4.5, it is clearly seen that there is a peak around 120 which is the periodicity of the analyzed signal.

Aside from the cepstrum analysis, there are other spectral estimation methods. Tokuda *et al.* (1994) proposed a general framework that includes the cepstrum as a special case, which is called mel-generalized cepstral (MGC) analysis [58].

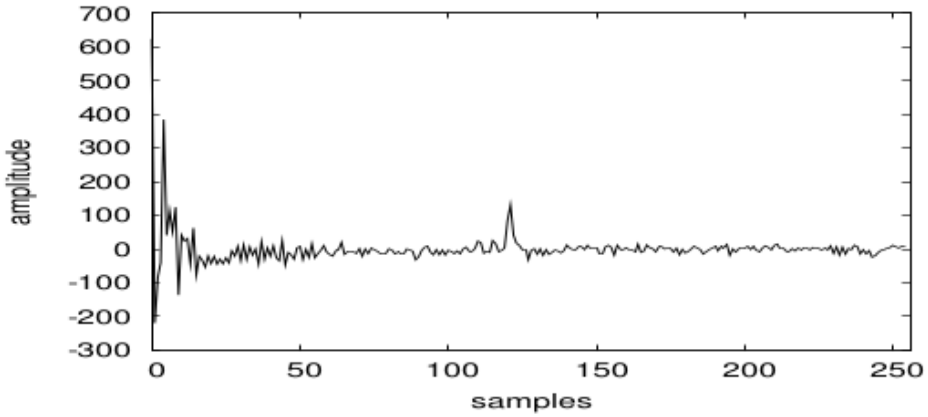

Figure 4.5. Cepstrum figure of a signal whose fundamental frequency is 120 Hz (Taken from Taylor (2009) [3]).

4.3.2. Fundamental Frequency (F0) of Speech

The difference between the fundamental frequency of the perceived signal and the frequency at which the vocal cords vibrate is very tiny and can be assumed to be equal [3]. The fundamental frequency (F0) of a signal is the frequency rate of the lowest frequency component of the signal. There are several techniques to extract the F0 information from a given signal.

One way to find the F0 is the examination of the cepstrum analysis since its higher values possess the information regarding the pitch and a spike is visible at a point corresponding to the period of the fundamental frequency as shown in Figure 4.5 where the the period is 120. The F0 is found by dividing the sampling rate by this number since the frequency is the inverse of the period.

$$f_0 = \frac{1}{T_0} \tag{4.26}$$

In cases where there is no apparent peak in the higher quefrencies (right side of the cepstrum), the signal is said to be unvoiced and sourced by a noise rather than a

periodic wave.

The calculated F0 values are converted into the log scale (LF0) before the training process.

4.3.3. Band Aperiodicity of Speech

Vocal cords do not move in a perfectly periodic manner even when voiced phonemes such as vowels are articulated. Uttered sounds also include the energy of components coming from turbulent air which prevents the produced waveform from being strictly periodic. Aperiodicity is described as the ratio between the power of periodic component and the power of noise [59]. This ratio varies depending on the frequency band. Thus, it needs to be specified for each frequency band, hence the name, band aperiodicity (BAP).

4.4. Objective Evaluation

Subjective tests are in general hard to replicate and costly. They also require more time than objective tests [60]. One of the most successful objective tests in generating correlated results with Mean Opinion Score (MOS) is the Perceptual Evaluation of Speech Quality (PESQ) [61].

The parameters derived from the original speech can be a reference for comparison with generated parameters. One of the acoustic features, that is used for re-constructing speech, is Mel-Generalized Cepstral Coefficient (MGC). As a similarity measure, the distance between MGC of the target parameters and the estimated parameters are calculated, which is called the Mel-Cepstral Distortion (MCD) [62].

$$MCD = \frac{10\sqrt{2}}{ln10}\frac{1}{M}\sum_{m}\sqrt{\sum_{d}(c(d,m)-\hat{c}(d,m))^2} \tag{4.27}$$

where $\hat{c}(d,m)$ and $c(d,m)$ are dth mel-cepstral coefficients of estimated and reference signals in the time frame m, respectively [62]. m is the frame number and M is the total number of frames.

Another objective evaluation measure is the root mean square error (RMSE) which is an error measure between the observed and the predicted values.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i}^{n} e_i} \tag{4.28}$$

where e_i is the error and n is the sequence length. RMSE score is not utilized in this study since it fails to capture the logarithmic nature of the human perception.

4.5. Subjective Evaluation

4.5.1. Mean Opinion Score

In terms of quality and intelligibility, subjective tests provide more accurate measures than objective tests. One of the most famous subjective measure is the mean opinion score (MOS) [63]. In MOS, listeners are expected to score the quality of a given sound sample in a room that meets the conditions recommended by P800 of Rec (2006) [63], "The listener should be seated in a quiet room with volume between 30 and 120m^3 and a reverberation time less than 500 ms (preferably in the range 200-300 ms)." Additionally, it is also recommended that the environmental noise should be configured to the appropriate level according to noise spectrum provided by P800 of Rec (2006) [63].

Opinion score is determined according to the Table 4.1. The MOS is calculated by taking the mean of all the scores of every listener.

Table 4.1. Mean Opinion Score (MOS) table.

Score	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

4.5.2. ABX Test

The standardization attempts of auditory tests and the proposition of the ABX test were made by Munson and Gardner (1950) to evaluate whether the difference between two set of samples in terms of similarity to reference samples is significantly perceivable [64]. During the test, 3 audio samples labeled as A, B and X are presented to the subjects where samples with A and B labels are generated by the 2 models being compared whereas the audio sample with the X label is the reference sample and all 3 of them belong to the same text. The subjects are asked the question that which one of A or B is more similar to X (the reference sample) in terms of naturalness, intelligibility, voice characteristics etc. The subjects are free to stay neutral and not choose A or B. Samples from the two sets are randomly placed in A or B option in each trial in order to prevent the occurrence of bias by the subject towards one set over the other during the test. If one of the two sets is not significantly preferred by the subjects, then it is said that there is no perceivable difference between the two sets.

4.5.3. AB Test (Preference Test)

Akin to the ABX test, the subjects are asked to choose either A, B or neutral option in the same manner. However, in the AB test, the reference samples are not shown to the subject and the investigated aspect is that which one of A or B has a more preferable sound quality instead of similarity to a third sample.

Both in AB and ABX tests carried out in this study, we modified the web-based interface provided by Kraft and Zölzer (2014) and used it as our test environment [65]. AB tests were conducted prior to the ABX tests since the target samples are revealed to subjects in ABX tests. Otherwise the subjects' preferences might have been affected in AB tests where they are not supposed to see the target sample.

5. PROPOSED APPROACH

The neural network framework is commonly used by many for training acoustic models. However, there are challenges of training sequential data, which might have arbitrarily long contextual dependencies. Aside from exploding gradient problem [66], the major one among the challenges is the vanishing gradient problem, which causes inability for the model to memorize long dependencies [67, 68]. The problem emerges as the depth of the neural network structure increases, which is the case for deep neural networks and recurrent neural networks. Nonetheless, recurrent units in the neural network structure such as Long-Short Term Memory (LSTM) are proved to be successful for acoustic modeling and handling the vanishing gradient problem [69–72].

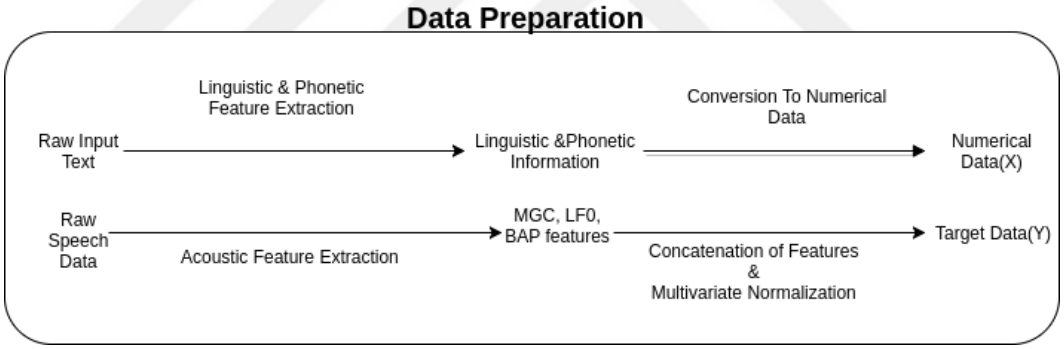

Figure 5.1. Workflow of data preparation

As a baseline approach, an average model or speaker independent (SI) model is built using utterances from 135 different speakers. Then as a baseline adaptation technique, weights in all the layers of the network except the LSTM layer and the output layer, which are relatively more decisive on the output, will be frozen during adaptation and the weights of the LSTM layer and the output layer will be retrained with the adaptation data. The workflow of the whole process is as shown in Figures 5.1, 5.2 and 5.3.

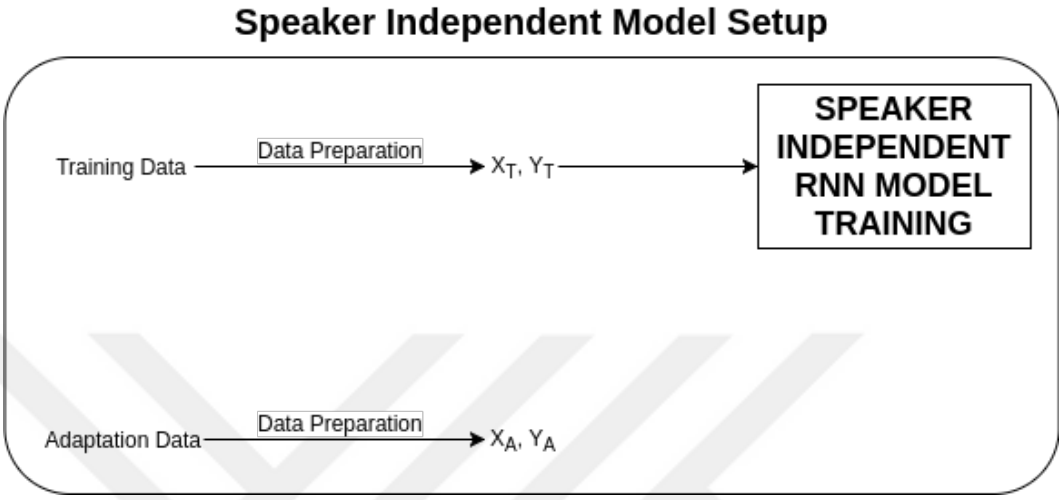

Figure 5.2. Speaker Independent Model

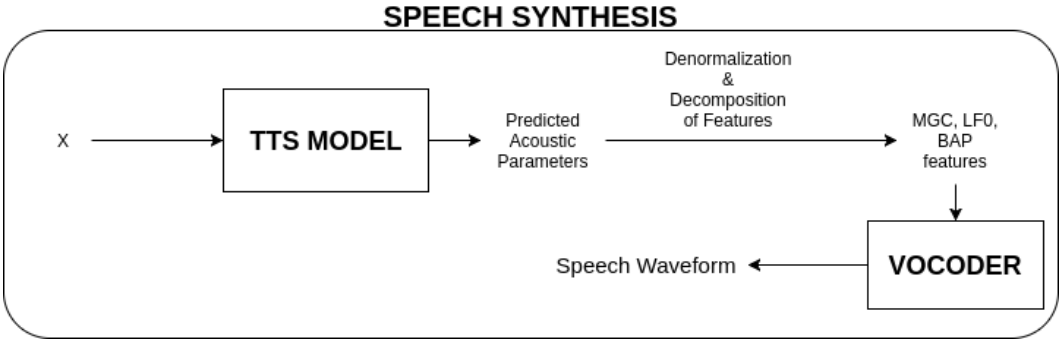

Figure 5.3. Workflow of speech synthesis

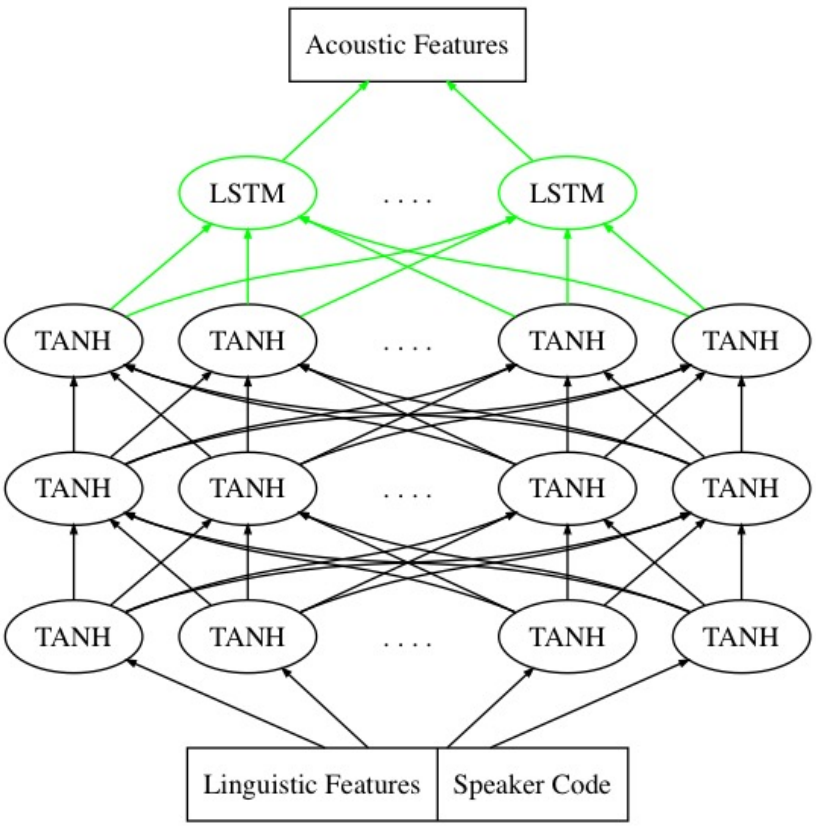

Figure 5.4. Baseline network architecture. This figure shows the structure that is trained to form the speaker independent (SI) model. The model predicts the acoustic features conditioned on the linguistic features and a speaker code.

This technique is known as transfer learning. Many of the pioneering studies in TTS field included multi-task learning as mentioned in chapter 3 which is a similar concept to transfer learning. However, there is an important distinction between the two. In multi-task learning, the learning agent simultaneously collects information on diverse tasks and as many models as the number of tasks are formed as a result of a collective training. However, in transfer learning there is a one-way dependency from the source task model to the target task model [73]. The source model is included in the target model according to a design of transferring information.

As another part of the baseline adaptation (BA), the network is fed by speaker codes in addition to the text features as input. A speaker code is any vector that can be considered as an embedding of a speaker's voice identity.

The transfer learning approach and using speaker codes for adaptation are already known techniques. The main contribution of this work both includes and extends these two. Our novel adaptation techniques are two-fold. One aspect is a better replacement of a given speaker code (identity vector or i-vector in our case) using nearest neighbor and clustering methods. It is shown that the proposed method significantly boosted the adaptation performance regardless of whether the baseline adaptation is applied or not. The other aspect of this work is the adaptation of postfilters, to the best of our knowledge, which are used in the TTS context but not examined in terms of their ability to adapt to new speakers. We also extended the postfilter adaptation process and incorporated adversarial training into it. Moreover, a cluster-based weight initialization technique is applied to better optimize the postfilter network. We finally demonstrated that the postfilter adaptation significantly increases the similarity of the generated samples to the reference samples.

6. METHODOLOGY

6.1. Recurrent Neural Networks and LSTM

RNNs have recently attracted the attention of different communities modeling and analyzing temporal data especially due to the advancements of hardwares leading to significantly faster training and their unprecedented success. Unlike conventional Artificial Neural Networks (ANN), RNNs also have feedback connections which, theoretically, enable them to use previous states' information. Despite its cyclic structure, no neural network structure including RNNs has backward connection when unfolded. This might be counter-intuitive at a first glance but an RNN's feedforward structure could be seen once it is unfolded in time as shown in Fig 6.1 and Fig 6.2 where the temporal length of data becomes the depth of the unfolded network.

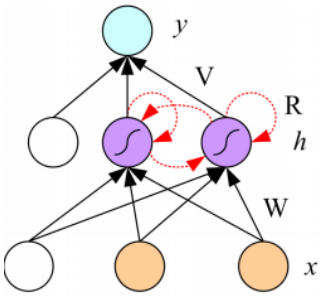

Figure 6.1. A simple RNN. Figure is taken from Alpaydin (2010) [74]

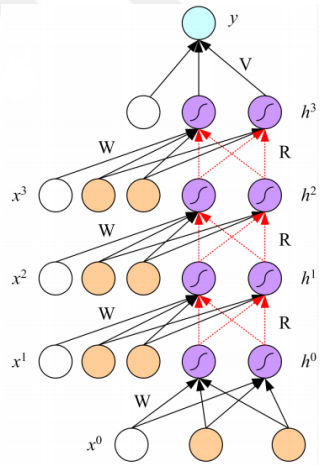

Figure 6.2. Unfolded version of the RNN. Figure is taken from Alpaydin (2010) [74]

A simple, conventional RNN cell is a function of previous layer's output and the previous output of itself. Feedback mechanism is generally local where recurrent cells use their own previous outputs.

$$h_i^t = g(f(x_i^t, h_i^{t-1})) \tag{6.1}$$

f is mostly an arbitrary linear function of x_i^t and h_i^{t-1}, g is an activation function, x_i^t and h_i^t are the i^{th} layer's input and output at time $t-1$, respectively [74]. Activation functions squash their input into a small interval (other non-linear transformations are possible too, such as Rectified Linear Unit where $ReLU(x) = max(0, x)$) which brings nonlinearity to the model and prevents the structure from being simple consecutive matrix multiplication of layer weights.

One of the most advantageous attribute of RNNs is the parameter sharing. The same parameters are shared with all the time steps but this is possible because of the strong assumption that the transition between time steps t and $t+1$ is static and can be modeled independently of t [75]. This assumption may be an oversimplification in cases where dependencies are long such as NLP but in the context of TTS, it is a plausible assumption since a generated output at a time step barely depends on long-past information i.e. a given result at time t^m, h^{t^m} can be assumed to be independent of h^{t^n} or any state information s^{t^n} where $m >> n$.

One particular and commonly preferred type of RNNs is the Long Short-Term Memory (LSTM). Due to its ability of capturing sequential dependencies in a temporal data and its popularity among scholars, it is appropriate to make an exclusive introduction to LSTM under RNN section. LSTMs possess the recurrent characteristic as their output is dependent on previous outputs. Unlike conventional RNNs, an LSTM cell also have 3 gating mechanisms (input, forget and output gate): The input gate scales the effect of current input and the previous output on LSTM's cell state, the forget gate scales the effect of cell state on the cell's generated output and the output gate scales how much of the generated output will be released. All the equations in an LSTM cell are as follows:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{t-1}\right) \tag{6.2}$$

$$g_i^t = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^t + \sum_j W_{i,j}^g h_j^{(t-1)}\right) \tag{6.3}$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right) \tag{6.4}$$

$$s_i^t = f_i^{(t)} + g_i^{(t)}\sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right) \tag{6.5}$$

$$h_i^{(t)} = tanh\left(s_i^{(t)}\right) q_i^{(t)} \tag{6.6}$$

where σ is the sigmoid function and applied to all 3 gates' output to obtain a result between $[0, 1]$.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6.7}$$

and

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{6.8}$$

f, g, q, s, h are respectively the scaling of the forget gate, the input gate and the output gate, the state information and the eventual output.

By adjusting the weights of its gates, this complex inner structure provides flexibility to choose between storing and discarding information seen in a time step. To this day, LSTM is still one of the most popular choice and it is hard to say that it is being replaced by its contemporary counterparts such as Gated Recurrent Unit (GRU) which has a similar structure with less parameters [76].

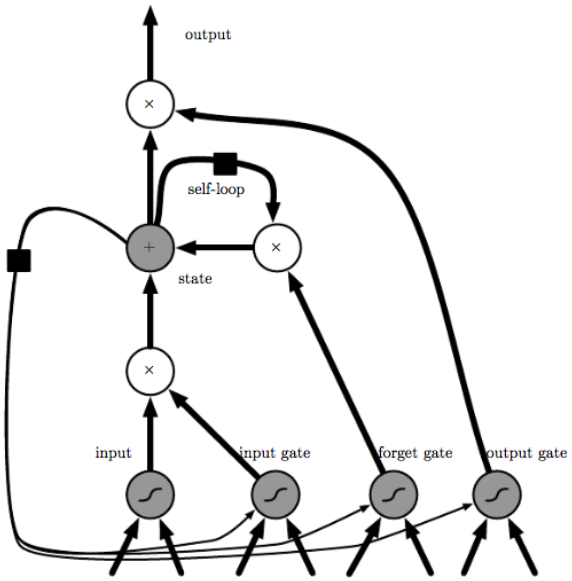

Figure 6.3. LSTM Cell Structure. Figure is taken from Goodfellow *et al.* (2016) [75].

6.2. Backpropagation Through Time (BPTT)

Classical backpropagation algorithm is a repetition of forward and backward passes where a forward pass is simply the calculation of prediction conditioned on the input with the current weights and a backward pass is the update of weights. This repetition is applied as many time as specified before training or until the weights are converged. In a forward pass, output is predicted based on the current weights. Then, weights are fine-tuned in a backward pass by updating each weight in proportion to a predefined learning rate times the weight's individual gradient with respect to a loss function as in Equation 6.9.

$$w \Leftarrow w - \alpha \frac{\partial F}{\partial w} \tag{6.9}$$

where α is the learning rate, and F is the loss function. F might change depending on the problem's definition but the prediction of acoustic features is a linear regression problem, hence, it is the mean square error (MSE) between the natural features and the predictions.

The feedback mechanism is visualized as a backward dependency in many representations but if the recurrent model is unfolded, it becomes clear that the underlying structure is a feedforward neural network that takes in the entire temporal data as previously shown in Fig 6.1 and Fig 6.2.

BPTT is very similar to a classical backpropagation algorithm except that it is done on RNN's unfolded version so that the error in loss is actually backpropagated through the time steps of data. This process may have to go through many layers if the length of a sequence is long. However, this overhead can be relieved by truncating the BPTT to a fixed number of steps, which is also called Truncated BPTT.

6.3. Environment

6.3.1. CUDA

CUDA is a platform and a programming interface created by NVIDIA enabling general-purpose GPU computing. It makes use of GPUs' multi-core structure to substantially increase computing performance by parallelizing the workload of algorithms. Thanks to CUDA, C, C++ and Fortran code can be directly received by GPU [77]. However, there are also libraries that can translate Python code to lower level instructions which can run on hardware. Unfortunately, the use of CUDA is limited to certain graphic cards and GPU-computing is done via CUDA Toolkit.

6.3.2. Theano

Theano is an open source Python library in which mathematical expressions can be defined symbolically [78]. These expressions are compiled as a graph on GPU with CUDA - compilation on CPU is also possible, nonetheless slower. Differentiation of symbolic expressions and parallel processing are the significant features for the implementation of backpropagation algorithm. RNN and feedforward DNN implementations for all the experiments are done via Theano framework.

6.4. Speaker Codes

The source task and also the first stage of the experiment is the training of an average model with different speakers. The input features that are fed to the network includes only some general linguistic and phoneme information but no attribute of the speaker. Information regarding the speaker's voice needs to be somehow given to the network in order to make the distinction between speakers. Two types of speaker codes are employed in the experiments. Selected speaker code (i-vector or cluster ID) is concatenated to every frame of input features.

6.4.1. Identity Vector and Its Extraction Process

Identity vectors (i-vector) are low-dimensional vectors carrying speaker characteristics extracted from high-dimensional speaker-dependent vector space via factor analysis [79]. It has been mainly used in the area of speaker identification. An i-vector can be extracted from any utterance but if there are more than one utterances for a speaker, which is the case in our data set, i-vectors are averaged so that a speaker is represented by a single i-vector.

Using i-vectors as a representation of voice characteristic of speakers is one of the state-of-the-art methods in text-independent speaker verification. The extraction process consists of several steps some of which require introduction to better understand the overall concept.

Before i-vector or any other speaker modeling techniques, feature vectors of voices are modeled as high dimensional vectors using Gaussian Mixture Models (GMM). GMM has the essential underlying assumption that d_S dimensional feature vectors of speech come from a probability density function that is a weighted summation of Gaussian densities. Let S be the feature vectors that are extracted from a speech sample such that $S = \{s_1, s_2, s_3, ..., s_T\}$ where s_t is a d_S dimensional feature vector representing the features in the time step t. Let our GMM consist of M Gaussian densities such that $G = \{w_j, \mu_j, \sum_j\}$ and $1 \leq j \leq M$ where μ_j is a d_S dimensional

mean vector and $\sum_j$ is a d_S by d_S diagonal covariance matrix and $\sum_{j=1}^{M} w_j = 1$. Diagonal covariances are more computationally convenient and they still perform comparably to full covariances [80]. Given a GMM G, probability density function of s_t for G_j is

$$p_j(s_t) = \frac{1}{(2\pi)^{d_s/2} |\sum_j|^{1/2}} e^{-(1/2)(s_t-\mu_j)^T \sum_j^{-1}(s_t-\mu_j)} \tag{6.10}$$

The density with respect to the model G is the weighted summation of Equation 6.10:

$$p(s_t) = \sum_{j}^{M} w_j p_j(s_t) \tag{6.11}$$

Therefore, the log-likelihood of the speech sample S becomes

$$logp(S|G) = \frac{1}{T} \sum_{t=1}^{T} logp(s_t) \tag{6.12}$$

Given the feature vectors, the mixture weights and the parameters of the GMM model's mean and covariance matrices are iteratively estimated using the expectation-maximization (EM) algorithm [81]. Once a GMM model is trained with given speech features, all the mean values belonging to M Gaussian densities are concatenated to build one vector as a representation of the speech, which is called the supervector of the speech.

In speaker verification tasks, given a test sample speech, likelihood of two GMM models that are the model of the speaker to be verified (speaker-dependent model) and a model that is trained with a dataset of speech samples from many different speakers (a universal background model (UBM) or speaker-independent model), are compared to verify whether the given speech comes from the speaker's GMM or not. Since the speaker's GMM can be trained with a few samples, the EM training of the model is not initialized with random parameters as done in the UBM training but with the parameters of the UBM [80].

Supervector of mean vectors of UBM is typically a high dimensional vector and a dimensionality reduction technique can be used to reduce the number of parameters to adapt. A total variability space (TVS) approach is proposed by Dehak *et al.* (2011) that models the speaker variability in a single total variability space [79]. In the TVS approach,

$$\mathbf{m}_s = \mathbf{m}_0 + \mathbf{T}\mathbf{w}_s \tag{6.13}$$

where $\mathbf{m}_s$ is the speaker-dependent supervector, $\mathbf{m}_0$ is the speaker-independent mean supervector, $\mathbf{T}$ is a low rank rectangular matrix, and $\mathbf{w}_s$ is called an identity vector (i-vector) that has a lower dimension than the supervectors. The matrix $\mathbf{T}$ and the i-vector $\mathbf{w}_s$ are estimated using the EM algorithm once the supervectors of UBM and the speaker-dependent are estimated as mentioned above. This dimensionality reduction process enables some methods to be more practical that are otherwise computationally expensive such as a support vector machine (SVM) training on supervectors [82].

6.4.2. Cluster-based Features

I-vectors are high dimensional vectors and during the Speaker Independent (SI) model training the network is fed only a limited number of i-vectors many times. Thus, how well the network parameters can generalize to a new i-vector when adaptation is performed with limited data is an important concern. In the cluster-approach, the training speakers are clustered into K groups using the k-means algorithm with their i-vectors and the cosine similarity as in equation 6.14

$$Sim(v_1, v_2) = \frac{v_1.v_2}{\|v_1\|_2\|v_2\|_2} \tag{6.14}$$

where v_1 and v_2 are vectors. The 1-of-K approach is then used to set the speaker code. Since the number of clusters, K, can be way smaller than the i-vector dimensions and all cluster codes are observed by the network during training, the issue of generalization is addressed. However, some of the speaker-specific information is also lost during

clustering. Thus, a balance between generalizability and retaining the speaker information is needed when setting K. The parameter K is set as 5 since it has the best performance in preliminary experiments.

Since a nearest-neighbor approach is used here, a distance measure is needed to measure acoustic distances between the speakers. The cosine distance is employed as a metric between the i-vectors of speakers in order to choose the nearest neighbors.

6.5. Model Training

6.5.1. Architecture Choice

Many distinct training techniques and their combinations were utilized and compared during the experimentation. For each technique, the same network structure is used in order to preserve consistency while comparing results. The structure includes both conventional feed-forward and recurrent cells. In a deep network structure, cells in lower layers and higher layers might have different behaviors. Lower layers could be seen as a feature extractor which means that the raw input is converted into a more abstract information to be used in higher layers. Since the RNNs are powerful in sequence modeling, one may suggest that it would be best to use RNN cells in every layer. However, an RNN cell has more parameters than a feed-forward cell and it is generally harder to train parameters in lower layers than higher layers. Besides, forget gates of RNN cells in lower layers tend to saturate towards 0 suggesting that those cells block the previous state information and use only the current one, which is the behavior of a feed-forward cell. In contrast to that, RNN cells in high layers are more likely to incorporate previous state information [83]. Therefore, it is convenient to put RNN cells on top of feed-forward layers. As for our task a neural network structure with 3 feed-forward hidden layers at the bottom and 1 recurrent layer at the top is chosen as well as a linear output layer. This DNN structure is shown in Figure 5.4. The model consists of 3 fully connected feedforward (FF) layers with 512 units for each and hyperbolic tangent as activation, 1 long-short term memory (LSTM) layer with 256 units and a linear FF layer with the number of units as many as the num-

ber of acoustic features. The connections shown in green are the connections that are re-trained as part of the adaptation process whereas the black connections are trained only in SI model training. This choice is based on the study of Wu *et al.* (2015) where performance of recurrent cells are compared and significance of their gate structures are examined [45]. As for the recurrent layer, LSTM was chosen due to its well-known success compared to other recurrent cells.

6.5.2. Other Parameters

In order to fine-tune the model, Stochastic Gradient Descent (SGD) algorithm is employed. SGD has a faster convergence rate than full-batch learning because weight updates are done after one or few samples are seen rather than calculating the gradients for the whole data before update. However, updating the model frequently causes the training error to fluctuate. For that reason, momentum is added to the update equation. Since momentum takes the previous update into consideration, a smoother decrease trend in error graph is observed. For the first 20 epochs, 0.3 is selected for momentum. After 20 epochs, it is increased to 0.9. Feed-forward layers in the structure have hyperbolic tangent (TANH) as an activation function. Learning rate is $5x10^{-4}$ until 20th epoch. Then, learning rate is halved after each epoch because reducing the step size lowers the probability of overshooting the local minimum while the model is near it. Number of epochs for training is set to 200 but if the validation error does not improve for 10 epoch in a row, training stops automatically.

6.5.3. Input-Output

Acoustic features (MGC, LF0, BAP and VUV) are extracted from raw speech files and normalized. This output format, O, is chosen for all the training and adaptation processes. As for the input I, each label file is converted into binary form in addition to a few dimensions that are numeric in the range of $[0, 1]$. In addition to label files, an identity vector (i-vector) v^i for each speaker is taken into consideration for training. Preprocessing stages of input and output are explained in detail in section 4.1.

Both I and O are temporal data and consist of frames where a frame represents a time interval. Corresponding input and output samples must have the same number of frames even though the dimension of an input and an output frame might differ. E.g for an utterance, input and output matrices are of size (t^f, d_i) and (t^f, d_o) where d_i, d_o and t^f are input, output dimensions of a frame and the number of frames, respectively.

When i-vectors are incorporated into text features, for each frame in I, the speaker's i-vector v^i is concatenated to the frame so an input sample in the new input set I_v has the size $(t^f, (d_i + d_v))$ such that d_o, d_i and d_v are 154, 456 and 400, respectively.

6.5.4. Speaker Independent Training

A few utterance of a speaker might carry significant information regarding the speaker's voice characteristic. However, such a small set of data alone fails to provide enough information for building a new TTS model from scratch. Having an auxiliary SI model improves the synthesis intelligibility and quality by offering extra information extracted from a speech database with a great deal of variety.

Speaker independent (SI) model can be seen as an average of different speakers' voice characteristics. SI model accounts for the base of the adaptation process and is required as a complementary to target data. Since the ultimate quality of adaptation relies on the SI model's quality, SI model -thus, the training data for SI- must be comprehensive and well-trained. WSJ data set is divided into two parts, namely reference speakers and target speakers. SI model is trained with the speech of reference speakers whereas data of target speakers are used for both execution and performance evaluation of adaptation task. There are 135 reference speakers each of which has 50 utterances for training. In order to avoid overfitting, 1 utterance from each speaker is excluded from training and treated as a validation set. In other words, $135x49$ utterances are employed for fine-tuning and validation loss is calculated with respect to the remaining $135x1$ utterances at each epoch. The weights of the model that yield the best validation loss are saved.

The training process involves 2 distinct models which vary based on the speaker code that is used. A single SI model was trained using cluster IDs and similarly, one other model using i-vectors. However, for the i-vector case, the SI model was re-trained for each one of the 135 speakers which gives out 135 new models. Hence, at the end of the training, one SI model trained with cluster IDs, one trained with i-vectors and 135 i-vector-trained models that are later re-trained for each speaker were obtained. First two of these three are averaged models where the last 135 are specialized for each of 135 reference speakers.

The primary objective evaluation is to investigate the adaptation performance of the DNNs for the MGC features. Thus, in the experiments, we used DNNs that predict only MGC and DNNs that predict all MGC, LF0 and BAP features. Comparisons are done for both cases.

6.5.5. Adaptation of Higher Layers

Adaptation process is basically the re-training of a previously trained model (the average SI model in our case) on a target speaker's data. However, in adaptation, we chose to update only the weights in specified layers instead of all weights. Rest of the weights are frozen and not modified. The top 2 layers, linear layer at the top of the neural network and the LSTM layer are chosen and their weights are updated exclusively during adaptation.

Both types of SI models, with i-vectors and with cluster IDs, are re-trained for each target and reference speakers. 5, 15 and 30 utterances are taken from each target speaker as adaptation data.

6.5.6. Nearest Neighbor Approach

Nearest Neighbor (NN) search is a special case of k-NN search where k is equal to 1. NN algorithm is simply searching for the nearest point to a given point with respect to a predefined distance metric. In our context, every speaker is considered as a point

and represented by their i-vector. The distance is calculated as

$$D = 1 - \frac{v_j^i v_k^i}{\|v_j^i\| \|v_k^i\|} \tag{6.15}$$

where v_j^i is the i-vector of speaker j.

NN approach is included in the adaptation process in several ways. Firstly, closest reference speaker (nearest neighbor) to each target speaker is found according to equation 6.15. Adapted models of the nearest reference speakers (NN model) are used and no adaptation is carried out during synthesis from NN models.

Layer adaptation is also combined with NN. Rather than re-training on SI models, adaptation is started with the NN model's weights. This gives rise to a better weight initialization for training which can be valuable because models with different initial points in state space may converge to different local minima.

6.6. Postfiltering Cepstral Features

6.6.1. Speaker-Independent model

Before any adaptation process, a Speaker-Independent (SI) Deep Neural Network (DNN) model with three feed-forward (FF) layers followed by one Long-Short Term Memory (LSTM) layer and one output layer is trained where an FF layer, the LSTM layer and the output layer have 512, 256 and 154 (the dimension of the output) units, respectively. The model is fed by approximately 5 hours of balanced speech data taken from 135 different male speakers. Every speaker has 50 utterances each of which has the duration of 3-4 seconds. The DNN model is trained for 50 epochs with the batch size of 4 where each utterance is zero-padded to the longest utterance's length for the training convenience. During the training, the speakers' i-vectors are given to the network together with the text features in order to represent the speakers' identity. The i-vectors extracted from every utterance of a speaker are averaged to one vector and this

one i-vector is concatenated to each frame of the text features. Prior to the training, the text features are normalized to the range of [0, 1] by min-max normalization whereas the acoustic features are standardized by subtracting the mean and dividing by the standard deviation.

6.6.2. Baseline Adaptation (BA)

As a baseline approach for the adaptation process, we retrain the prepared SI model with 1, 2 and 3 utterances of the target speaker. However, only the LSTM layer and the output layer parameters are re-updated since these layers are more decisive (weights in higher layers have greater gradients on the loss function) for the network's output. The parameters of the three lower FF layers are frozen and not updated to keep the number of parameters small and to avoid overfitting, considering the fact that the target speaker's data is very limited. As another part of the baseline adaptation process, the i-vector extracted from the target speaker's utterance (s) is given to the network in the same way as in the SI model training together with the text features.

6.6.3. Postfiltering (PF)

The proposed Post-Filtering (PF) method is applied only to Mel-Generalized Cepstral (MGC) features once the baseline adaptation process is completed. Differently from the baseline model, another fully-connected FF model with one hidden layer having 64 units and no recurrent structure is implemented and trained using stochastic gradient descent to map the generated MGC features to the target MGC features. Let $o_i = [\hat{c}_i^T, \Delta\hat{c}_i^T, \Delta^2\hat{c}_i^T]^T$ be the output vector of the baseline network model for the i^{th} time step and $\hat{c}_i^T, \Delta\hat{c}_i^T, \Delta^2\hat{c}_i^T$ be the cepstral, delta-cepstral and delta-delta-cepstral coefficients, respectively. The ultimate prediction c_i of the baseline, is generated when the output o_i goes through the Maximum Likelihood Parameter Generation (MLPG) [49]. Since the PF structure does not include any recurrent layer, the context information is provided to the PF model by giving the previous c_{i-1} and the next step's c_{i+1} features. Additionally, the PF model also takes state and phoneme information as 1-of-k vectors. Let pn_i and st_i be the 1-of-k vectors representing, respectively, the

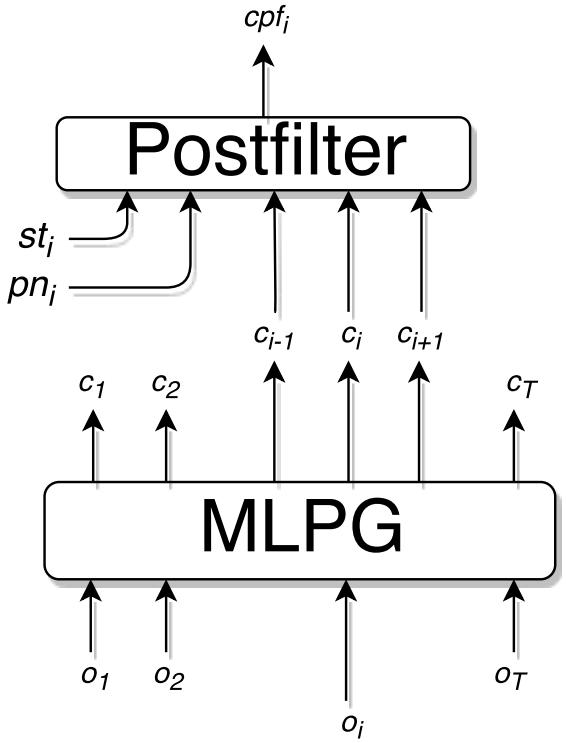

Figure 6.4. An overview of the postfiltering algorithm applied to a vector representing the cepstral features at a time step i in a context. Phoneme and state ids belonging to the ith time step are also given to the postfilter system as input.

phoneme and the state information of the i^{th} time step of an utterance. In other words, $pn_i^T = [pn_{i1}, pn_{i2}, ..., pn_{ij}, ..., pn_{ik_{pn}}]$ such that $pn_{ij} = 1$ and $pn_{ij'} = 0, j' \neq j$ where k_{pn} is the number of possible phonemes and j is the id of the phoneme corresponding to i_{th} time step. Similarly, $st_i^T = [st_{i1}, st_{i2}, ..., st_{ij}, ..., st_{ik_{st}}]$ such that $st_{ij} = 1$ and $st_{ij'} = 0, j' \neq j$ where k_{st} is the number of possible states and j is the id of the state corresponding to the i_{th} time step. Consequently, an input vector for the PF model is $I_i = [c_{i-1}^T, c_i^T, c_{i+1}^T, pn_i^T, st_i^T]^T$ whereas the corresponding output vector is cpf_i.

6.6.4. Cluster-based initialization for the postfilter training (CLSI)

Although the baseline adaptation model is fine-tuned on top of the SI model, our proposed PF model's training starts with random weights. At this point, we investigated whether there is any room for improvement to the PF model's initialization. We hypothesized that a PF model that is pre-trained with the reference speakers whose

voice characteristics are similar to a target speaker might be a better initialization for the target speaker's PF model than a random initialization. Our aim here is to help the PF model parameters to settle at a possibly better local minimum by only setting the initial values of the weights.

We divided the reference speakers into 5 groups by clustering them according to their i-vectors using the k-means clustering method since 5 groups performed the best in preliminary experiments. Then, for each group, one model is trained in the same way as described in section 6.6.3 with the utterances of the speakers belonging to that group. Then, during the PF adaptation for a target speaker, we initialize the model with one of these 5 pre-trained models. As a selection criterion, the euclidean distance of each group's mean vector to the target speaker's i-vector is taken into consideration.

6.6.5. Inclusion of adversarial network into postfiltering (ADV)

Generative Adversarial Networks (GAN) have been discussed and probed by many researchers due to its ability to generate distinct fake samples that are similar to a set of given samples from an unknown distribution [84]. In addition to the adversarial training's success in computer vision, the application of this idea can contribute to the naturalness of the samples from other distributions such as those in the speech context. Indeed, in the work of Saito *et al.* (2017), it is shown that incorporating adversarial training contributes to the quality of the speech [85]. There are a several variations of the adversarial training that are experimented by Saito *et al.* (2017) but we adopted only the main idea and integrated to our PF system as shown in Figure 6.5 [85]. In the first PF model, the network is trained using the standard mean squared error (MSE) loss as below:

$$L_{MSE}(\hat{y}, y) = \frac{1}{T}\sum_{t=0}^{T}(y_i - \hat{y}_i)^2 \tag{6.16}$$

where $\hat{y}_i$ is a vector prediction for the time step i and y_i vector is the target parameters for the same time step. By introducing an adversarial network, we include a binary

cross entropy loss function to our PF model training, which is:

$$L_{BCE}(\hat{\mathrm{y}}, \mathrm{y}) = \frac{1}{T}\sum_{t=0}^{T} -(\mathrm{y}_i log(\hat{\mathrm{y}}_i) + (1-\mathrm{y}_i) log(1-\hat{\mathrm{y}}_i)) \tag{6.17}$$

where $\hat{\mathrm{y}}_i$ is a scalar prediction for the time step i and $\hat{\mathrm{y}}_i \in \{x \in R \mid 0 < x < 1\}$ while y_i is the target value for the time step i and $\mathrm{y}_i \in (0, 1)$ denoting the label as either fake (prediction) or real (natural). The weights of our PF model and the adversarial network are updated consecutively as in a standard GAN training. The loss function for the overall PF model here is as follows:

$$\mathcal{L}_{PF} = L_{MSE}(cpf, cnat) + \frac{E_{L_{MSE}}}{E_{L_{BCE}}} L_{BCE}(ap_{cpf}, \mathbf{1}) \tag{6.18}$$

where $E_{L_{MSE}}$ and $E_{L_{BCE}}$ are the expected values of MSE and binary cross entropy losses. ap_{cpf} is the prediction of the adversarial network when its input is the parameters generated by the PF network and $\mathbf{1}$ is a vector of ones that has the same length as ap_{cpf}. In other words, in addition to the conventional MSE loss (Equation 6.16), Equation 6.17 is plugged such that the weights of the PF model can be updated in the direction that the generated parameters cpf becomes indistinguishable from the natural parameters $cnat$ when evaluated by the adversarial network. $\frac{E_{L_{MSE}}}{E_{L_{BCE}}}$ can be considered as the scaling factor to boost the effect of L_{BCE} since $E_{L_{MSE}} >> E_{L_{BCE}}$ in the training, which accordingly makes the gradients from L_{MSE} and L_{BCE} to differ greatly in magnitude. On the other hand, the weights of the adversarial network are updated based on the following 2 loss functions:

$$\mathcal{L}^1_{ADV} = L_{BCE}(ap_{cpf}, \mathbf{0}) \tag{6.19}$$

$$\mathcal{L}^2_{ADV} = L_{BCE}(ap_{cnat}, \mathbf{1}) \tag{6.20}$$

where ap_{cpf} and ap_{cnat} are the scalar predictions for, respectively, the generated parameters of the PF model and the natural parameters. $\mathbf{0}$ and $\mathbf{1}$ are the vectors of zero

and one. The decision to pick either $\mathcal{L}^1_{ADV}$ or $\mathcal{L}^2_{ADV}$ for the weight update is made in a balanced and mutually exclusive manner as carried out typically in the training process of a GAN model's discriminator component.

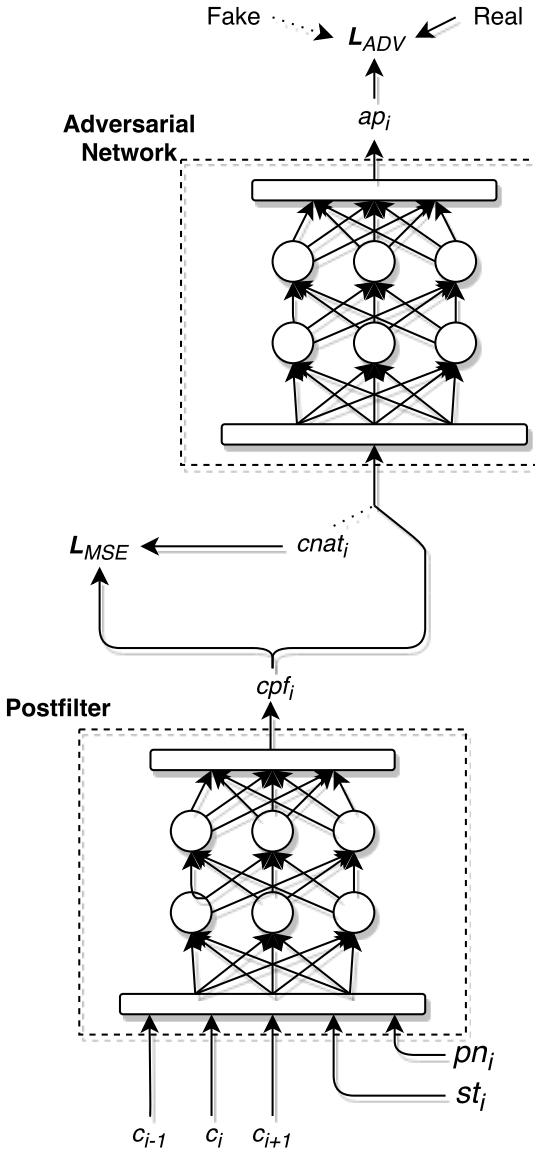

Figure 6.5. Postfiltering process with adversarial training. The input of the adversarial network is switched between natural and generated parameters. The PF model benefits both the MSE loss and the loss coming from the prediction of the adversarial network regarding the PF model's generated parameters.

7. RESULTS

After each experiment, generated parameters are de-normalized before evaluation and objective results for MCD values are reported as described in 4.4.

In the first set of experiments, we experimented with the speaker codes without adapting the network weights. Results are shown in Table 7.1. 5 cases are considered. NN i-vector is used in the first case (SI-NN), target speaker's i-vector is used in the second case (SI-tar), nearest-neighbor's model (selected with i-vector) is used in the third case (NN), nearest-neighbor's model (selected by brute force, the model that has the minimum generation error, MGE) is used in the fourth case (NN-MGE), and cluster-ID is used in the last case. Using the i-vector of the nearest-neighbor outperformed using the i-vector of the target speaker. However, using the cluster-id as the speaker code performed the best. The same trend is observed both for the network that generates only the MGC features and the network that generates all features.

When the model of the nearest-neighbor is used instead of the speaker-independent (SI) model, performance is lower regardless of the speaker distance measure as shown in Table 7.1. The nearest-neighbor models are trained with a relatively small number of utterances compared to the training of the SI model and these NN models fail to outperform the SI model for other speakers that are acoustically similar.

Another interesting observation in Table 7.1 is that the network that generates all features (MGC+BAP+LF0) performs better than the network that only generates the MGC features. Since the number of LSTM cells is the same for both networks, generating all features through regression of the LSTM outputs is a more complex problem and a higher MCD is expected. However, the extra weights, especially the BAP features, seem to be playing a regularization role and helping the network reach a better local optima during optimization.

Adaptation performance of the MGC-generating network is shown in Table 7.2. Similar to results in Table 7.1, cluster-id worked the best. Although NN-based i-vector performed slightly better for the very low adaptation data, it performed the same as using the i-vector of the target speaker for the 50utt and 100utt cases.

Adaptation performance of the network that generates all features is shown in Table 7.3. Not only this network was better trained without adaptation as discussed above, it could also adapt more effectively to the target speaker as shown in Table 7.3. Moreover, using the NN's i-vector substantially outperformed the other speaker codes which is likely to be related to the better optima found for the nearest-neighbors during the SI training.

Since adapting a network with many weights is difficult when the training data is small, we assessed the importance of parameters in the output and LSTM layers during adaptation. Results are shown in Table 7.4. Adapting LSTM parameters only performed better than adapting the output and the LSTM layer for the 5-utterance and 15-utterance cases. Adapting all the weights of the network did not perform well for the 5-utterance case but, performed better than others for the 15-utterance and 30-utterance cases.

We next analyzed the variance of the weights to understand which weights store the most information and have the highest variance between the training speakers. The LSTM parameters that regulate the effect of the cell memory on the new output have the highest variance as shown in Figure 7.1. Weights and biases of the output layer is shown in green whereas the LSTM weights are shown in blue. wcf belongs to the forget gate and decides how much of the information will be allowed from the previous time step's cell state to affect the forget gate's output. High variation in wcf suggests that temporal information may be discarded or taken into consideration for the forget gate's respond depending on the incoming feature. wco is part of the output gate function and decides which features of the cell state should affect the output gate's respond. One conjecture can be that only some features affect the degree of the cell state's reflection on the output. Whc is the matrix that transforms the previous time

Table 7.1. Performance of the SI models are shown without any adaptation. One of the SI models is trained to produce only the MGC features. The other SI model is trained to produce all features.

	SI-NN	SI-tar	NN	NN-MGE	Cluster
MGC	5.46	5.53	5.53	5.53	5.22*
All	5.33	5.42	5.37	5.37	5.29*

p-values for MGC	SI-NN	SI-tar	NN	NN-MGE	Cluster
SI-NN	-	0.32	0.32	0.32	0.00*
SI-tar		-	0.97	0.96	0.00*
NN			-	0.96	0.00*
NN-MGE				-	0.00*
Cluster					-

p-values for ALL	SI-NN	SI-tar	NN	NN-MGE	Cluster
SI-NN	-	0.04*	0.55	0.55	0.25
SI-tar		-	0.16	0.95	0.00*
NN			-	0.95	0.09
NN-MGE				-	0.09
Cluster					-

Table 7.2. Adaptation performance when the target speaker's i-vector is used, NN speaker's i-vector is used, and the cluster ID is used at the input. SI model is trained to generate MGC features only and the last two layers (the LSTM and the output layer) of the network are adapted.

	5utt	15utt	30utt	50utt	100utt
NN i-vector	5.34	5.19	5.14	5.06	4.98
Target i-vector	5.36	5.20	5.15	5.06	4.98
Cluster ID	5.24*	5.12*	5.06*	4.99*	4.91*

p-values for 5utt	NN i-vector	Target i-vector	Cluster ID
NN i-vector	-	0.60	0.02*
Target i-vector		-	0.00*
Cluster ID			-

p-values for 15utt	NN i-vector	Target i-vector	Cluster ID
NN i-vector	-	0.57	0.09
Target i-vector		-	0.02*
Cluster ID			-

p-values for 30utt	NN i-vector	Target i-vector	Cluster ID
NN i-vector	-	0.81	0.07
Target i-vector		-	0.04*
Cluster ID			-

Table 7.3. Adaptation performance when the target speaker's i-vector is used, NN speaker's i-vector is used, and the cluster ID is used at the input. SI model is trained to generate all features (MGC+LF0+BAP) and only the last two layers of the network are adapted.

	5utt	15utt	30utt
NN i-vector	5.06*	4.93*	4.87*
Target i-vector	5.26	5.17	5.09
Cluster ID	5.24	5.16	5.12

p-values for 5utt	NN i-vector	Target i-vector	Cluster ID
NN i-vector	-	0.00*	0.00*
Target i-vector		-	0.93
Cluster ID			-

p-values for 15utt	NN i-vector	Target i-vector	Cluster ID
NN i-vector	-	0.00*	0.00*
Target i-vector		-	0.80
Cluster ID			-

p-values for 30utt	NN i-vector	Target i-vector	Cluster ID
NN i-vector	-	0.00*	0.00*
Target i-vector		-	0.58
Cluster ID			-

Table 7.4. Adaptation performance of the SI model that produces only the MGC features. I-vectors of the NN speakers are used for input. Only a subset of the network parameters in the output and LSTM layers. LSTM cell-weights correspond to LSTM weights that correspond to the memory of the LSTM cell.

	5utt	15utt	30utt
Output-only	5.33	5.24	5.19
LSTM-only	5.32*	5.20	5.16
Output+LSTM	5.37	5.20	5.15
Output+LSTM-cell-weights	5.34	5.20	5.15
LSTM-cell-weights	5.36	5.24	5.18
All-weights	5.35	5.18*	5.13*

p-values for 5utt	Output	LSTM	Output LSTM	Output LSTM-cw	LSTM-cw	All
Output	-	0.68	0.55	0.92	0.52	0.79
LSTM		-	0.31	0.61	0.29	0.50
Output LSTM			-	0.63	0.96	0.75
Output LSTM-cw				-	0.60	0.87
LSTM-cw					-	0.71
All						-

p-values for 15utt	Output	LSTM	Output LSTM	Output LSTM-cw	LSTM-cw	All
Output	-	0.30	0.30	0.30	0.95	0.12
LSTM		-	0.96	0.96	0.30	0.61
Output LSTM			-	0.96	0.30	0.61
Output LSTM-cw				-	0.30	0.61
LSTM-cw					-	0.12
All						-

Table 7.5. Table 7.4 continued.

p-values for 30utt	Output	LSTM	Output LSTM	Output LSTM-cw	LSTM-cw	All
Output	-	0.51	0.26	0.26	0.94	0.22
LSTM		-	0.57	0.57	0.46	0.57
Output LSTM			-	0.95	0.23	0.92
Output LSTM-cw				-	0.23	0.92
LSTM-cw					-	0.18
All						-

step's output whose high variation may be an implication that only some features from the previous time step are amplified. High variation in biases of the forget gate and the cell state, bf and bc, can also be an implication that the network is likely to filter the flow of some features through time and prioritize some features over the other while deciding the cell state. The weights of the output layer may be expected to have higher variation than the LSTM weights since they tend to be optimized by higher gradients because they are closer to the eventual output. When only those LSTM cell weights are adapted whereas the others are frozen, reasonable adaptation performance is obtained as shown in Table 7.4.

Since adapting all network parameters did not cause generalization issues, as shown in Table 7.4, we compared it with adaptation of the last two layers using the NN i-vector and cluster id for speaker codes. Cluster-based approach performed better as shown in Table 7.6. Moreover, even though generalization did not become an issue, adapting only the LSTM and output layers of the network was found to be sufficient.

7.1. Weight Examination of the Adapted Layers

Every adapted model differs from each other in its output and LSTM layer weights. It might be beneficial to study those variances for better understanding the

Table 7.6. Adaptation performance of the SI model that produces only the MGC features. I-vectors of the NN speakers are used as input in the first two systems, cluster-based approach is used in the other two systems. Adaptation of all network weights and only the weights of the last two layers are compared.

	5utt	15utt	30utt
NN-I-vector Output+LSTM	5.37	5.20	5.15
NN-I-vector All-weights	5.35	5.18	5.13
Cluster Output+LSTM	5.24*	5.12*	5.06*
Cluster-All	5.25	5.12*	5.06*

p-values for 5utt	NN-I-vector Output+LSTM	NN-I-vector All-weights	Cluster Output+LSTM	Cluster-All
NN-I-vector Output+LSTM	-	0.75	0.00*	0.01*
NN-I-vector All-weights		-	0.01*	0.02*
Cluster Output+LSTM			-	0.92
Cluster-All				-

p-values for 15utt	NN-I-vector Output+LSTM	NN-I-vector All-weights	Cluster Output+LSTM	Cluster-All
NN-I-vector Output+LSTM	-	0.61	0.02*	0.02*
NN-I-vector All-weights		-	0.07	0.07
Cluster Output+LSTM			-	0.96
Cluster-All				-

Table 7.7. Table 7.6 continued. Rest of the p-values for comparisons shown in Table 7.6 are given.

p-values for 30utt	NN-I-vector Output+LSTM	NN-I-vector All-weights	Cluster Output+LSTM	Cluster-All
NN-I-vector Output+LSTM	-	0.92	0.04*	0.04*
NN-I-vector All-weights		-	0.04*	0.04*
Cluster Output+LSTM			-	0.96
Cluster-All				-

adaptation process. The variance among the adapted weights of every target speaker model is shown in Figure 7.1. Invariant weights can be interpreted as resistant to adaptation or not informative about speaker characteristic whereas more variant weights are said to be more suitable for adaptation.

7.2. Subjective results for the postfiltering mechanism

We investigate the performance of the PF model and the other two algorithms we proposed by comparing the objective and the subjective results. The listening tests as depicted in the first bar chart of Figures 7.2, 7.3 and 7.4 suggest that the PF model when applied with the BA improves the performance significantly. The p-value for the mean difference between A and B is less than 0.05 for the three figures although the choice of neutral option becomes more common as the number of utterances for the adaptation increases. This implies that the effect of PF is at its peak when the data for adaptation is very little. The same trend can be seen in the second bar chart of the same figures. Here, we hypothesize that CLSI is a significantly better choice with respect to a random weight initialization because the weights are pre-trained and the new training starts from a non-random and learned point in the search space thus, tends to stop at a better local minimum. However, that is not the case for the adversarial training. Even

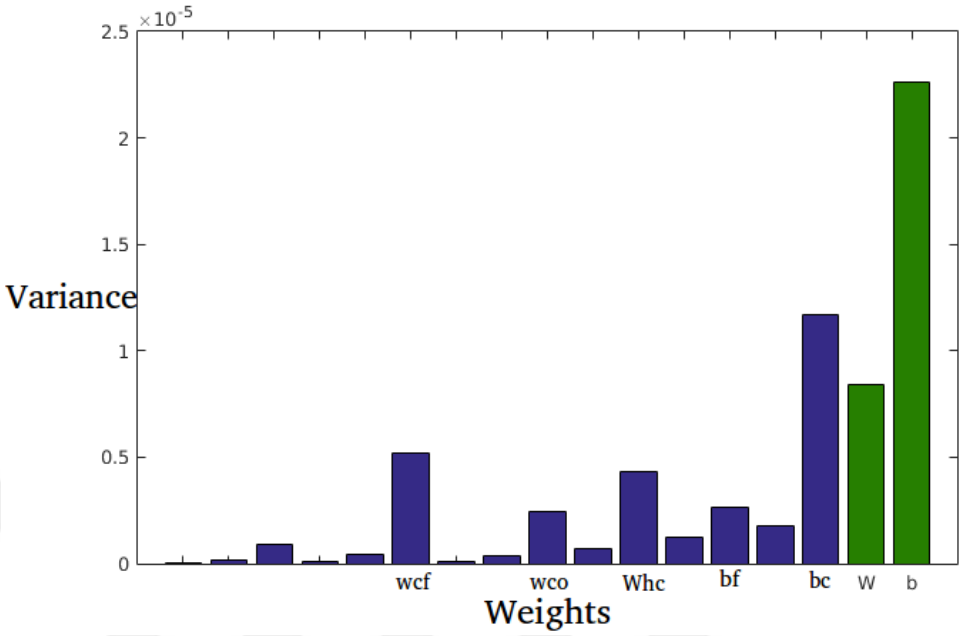

Figure 7.1. Variations in LSTM and Output layer weights are as above.

though it has a better mean value than the one without it for the 1-utterance case as shown in the third column of Figure 7.2, this difference is not significant. In other words, it can be argued that the subjective results suggest that the PF and the CLSI algorithms have significantly beneficial effects on adaptation, unlike the performance of the ADV.

In Table 7.14, unlike ABX test results, it can be seen that the PF and the CLSI extensions have a deteriorating effect on BA whereas there is a significant drop in error when the ADV is included. It even outperforms the rest for the 1-utterance adaptation case.

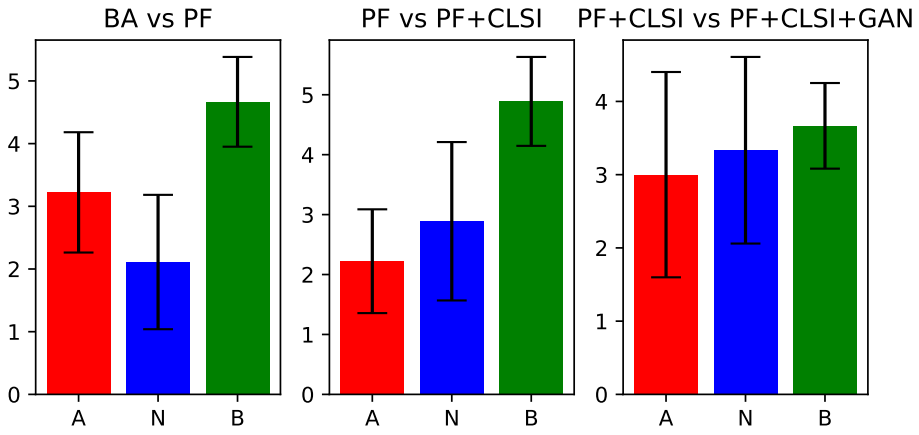

Figure 7.2. ABX test results for 1-utterance case. The PF and the CLSI algorithms have significant contribution for adaptation.

Table 7.8. t-test p-values for ABX results for 1 utterance

	BA vs PF	PF vs PF+CLSI	PF+CLSI vs PF+CLSI+GAN
P-value	0.05*	0.00*	0.45

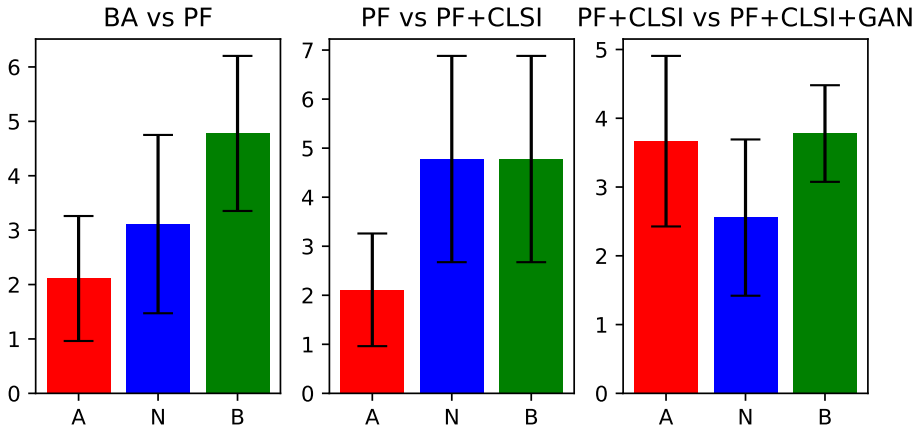

Figure 7.3. ABX test results for 2-utterance case. Similar to 1-utterance case, the PF and the CLSI algorithms have significant contribution.

Table 7.9. t-test p-values for ABX results for 2 utterances

	BA vs PF	PF vs PF+CLSI	PF+CLSI vs PF+CLSI+GAN
P-value	0.02*	0.07	0.89

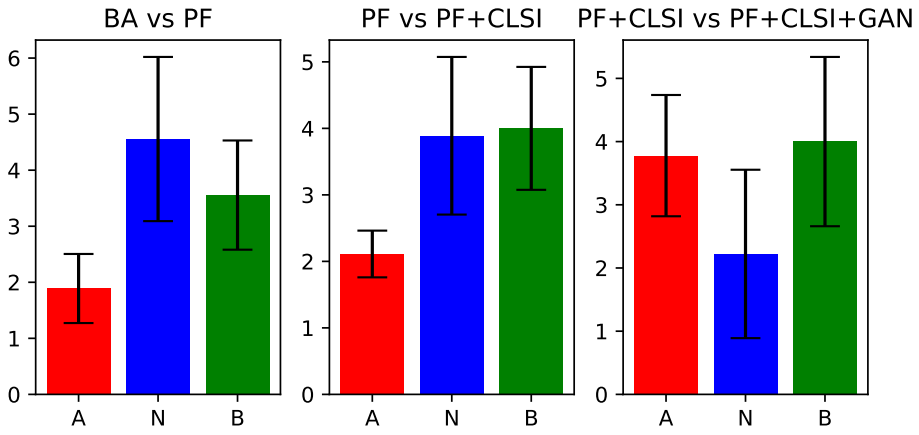

Figure 7.4. ABX test results for 3-utterance case. Although the PF and the CLSI algorithms have still significantly better contribution, the rise in neutral option suggests that they become less effective as more data is used.

Table 7.10. t-test p-values for ABX results for 3 utterances

	BA vs PF	PF vs PF+CLSI	PF+CLSI vs PF+CLSI+GAN
P-value	0.02*	0.00*	0.82

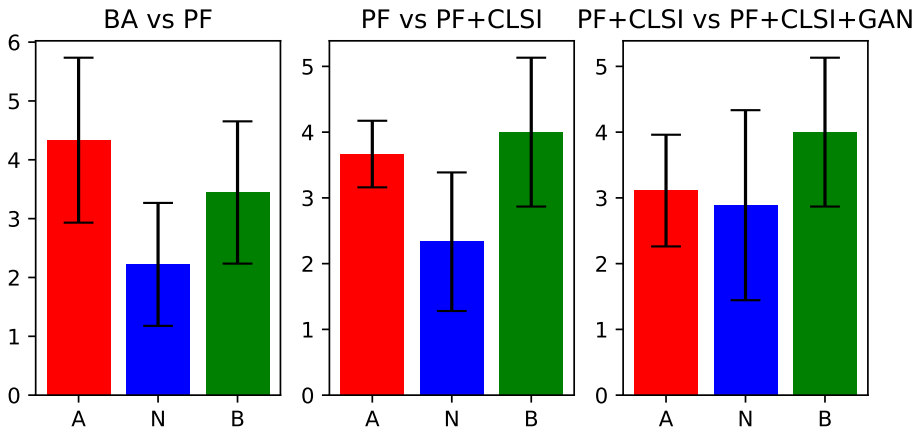

Figure 7.5. AB test results for 1-utterance case.

Table 7.11. t-test p-values for AB results for 1 utterance

	BA vs PF	PF vs PF+CLSI	PF+CLSI vs PF+CLSI+GAN
P-value	0.41	0.64	0.29

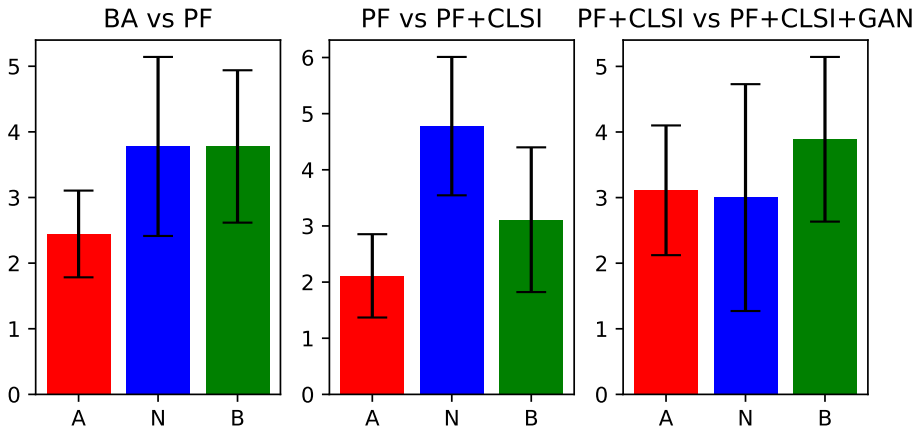

Figure 7.6. AB test results for 2-utterance case.

Table 7.12. t-test p-values for AB results for 2 utterances

	BA vs PF	PF vs PF+CLSI	PF+CLSI vs PF+CLSI+GAN
P-value	0.10	0.26	0.41

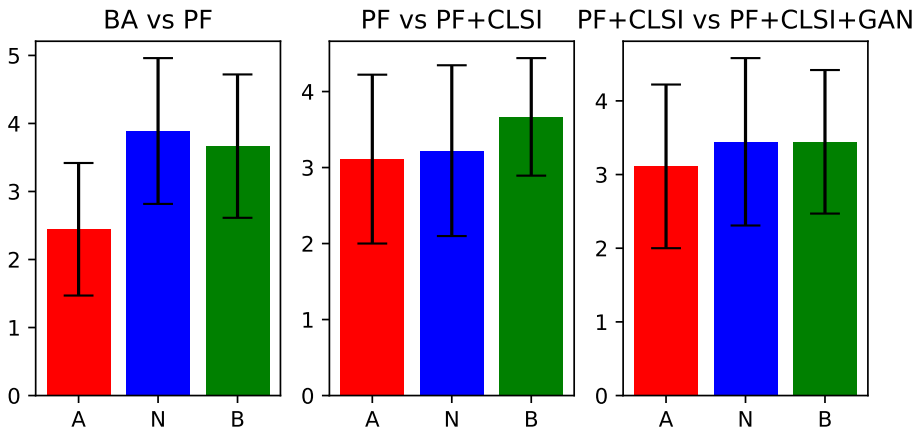

Figure 7.7. AB test results for 3-utterance case.

Table 7.13. t-test p-values for AB results for 3 utterances

	BA vs PF	PF vs PF+CLSI	PF+CLSI vs PF+CLSI+GAN
P-value	0.15	0.48	0.70

7.3. Postfiltering Variance Examination

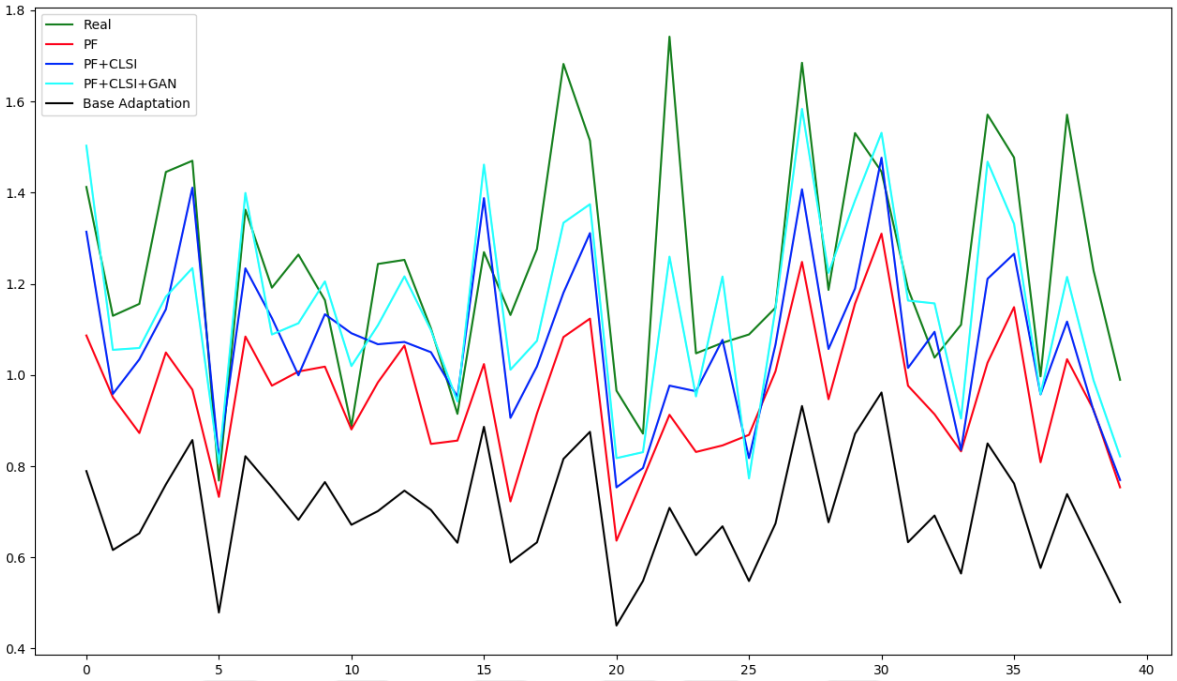

Figure 7.8. The variances of the first dimension of MGC features (energy) taken from 40 test utterances of a target speaker are plotted as above. The most realistic variance plot is obtained when the 3 algorithms are used together (in cyan).

Another aspect we investigated is the variance of the generated parameters. 40 test utterances of a random target speaker are taken for analysis. We calculated the variance of the energy features (first dimension of MGC features) of each of the generated parameters since the energy has the largest range among the MGC features. 40 variance values corresponding to the 40 utterances are plotted for each of the 4 systems together with the variances of the real parameters in Figure 7.8. The method whose variance plot is most visibly distant from that of the real parameters (shown in green) is the baseline adaptation (shown in black) and the closest variance plot belongs to the fourth system where all the 3 algorithms are applied together (shown in cyan). Based on Figure 7.8, it is fair to say that each of these proposed methods has a noticeably positive effect on the variance in features of a sequence and that the 3 algorithms together have the most realistic variance plot.

Table 7.14. MCD scores for 4 systems are given. The proposed algorithms have the best performance when the available data is very limited.

Utt	BA	BA+PF	BA+PF+CLSI	BA+PF+CLSI+ADV
1	5.48	5.96	5.75	5.32*
2	5.21*	5.63	5.47	5.34
3	5.13*	5.65	5.44	5.33

p-values for 1utt	BA	BA+PF	BA+PF+CLSI	BA+PF+CLSI+ADV
BA	-	0.00*	0.00*	0.00*
BA+PF		-	0.00*	0.00*
BA+PF+CLSI			-	0.00*
BA+PF+CLSI+ADV				-

p-values for 2utt	BA	BA+PF	BA+PF+CLSI	BA+PF+CLSI+ADV
BA	-	0.00*	0.00*	0.00*
BA+PF		-	0.00*	0.00*
BA+PF+CLSI			-	0.00*
BA+PF+CLSI+ADV				-

p-values for 3utt	BA	BA+PF	BA+PF+CLSI	BA+PF+CLSI+ADV
BA	-	0.00*	0.00*	0.00*
BA+PF		-	0.00*	0.00*
BA+PF+CLSI			-	0.04*
BA+PF+CLSI+ADV				-

8. CONCLUSION AND FUTURE WORK

8.1. Conclusion

In this study, we proposed novel adaptation techniques, extended the existing ones and evaluated their performances. We also analyzed the results and considered different aspects of models such as behaviors and effects of certain weights. The proposed techniques are shown to improve the adaptation performance in terms of both objective and subjective measures.

The adaptation techniques in this study were adopted in the context of DNNs which are the prominent models not only for the TTS systems but also other fields such as NLP and Computer Vision. Thus, our adaptation techniques can be embraced by those fields encompassing problems on which a transfer learning approach with DNNs may be applied. The proposed methods are two independent adaptation process that are implemented on the acoustic model and the postfilter and they can be applied independently. The adaptation on the acoustic model increased the quality of the sound whereas the postfilter adaptation increased the similarity to the target speaker's voice.

The DNN model framework, hence the proposed methods, has several issues that are hard to address. Firstly, there are infinitely many possible combinations of network architectures with infinitely many combinations of hidden unit size selections. Unfortunately, this is still an open issue and there are no well-accepted guidelines for selecting the optimum network for a given problem except for a few intuitive and empirical suggestions. Additionally, the cost of training is another concern especially for networks with a large number of weights, which prohibits an extensive hyperparameter optimization. One other problem of DNNs is that they consist of stacked weights separated by non-linear functions, which makes the interpretation of weights very difficult, hence, complicates the analysis.

8.2. Future work

The experiments in this study were conducted on networks that contain recurrent and fully-connected feedforward layers. However, new neural network structures have still been being introduced and some of them can be presumably transferred to the TTS context. Our proposed techniques can be incorporated into these new DNN types. Moreover, our postfilter adaptation methods can be also experimented even on non-DNN TTS models that generate acoustic parameters.

The i-vector-based speaker verification is one of the state-of-the-art methods in speaker verification. However, other types of speaker codes such as x-vectors [86] can be an alternative or a replacement for the i-vectors in the future. Our techniques can be applied to these speaker codes as well without extra assumption or modification to the overall model.

TTS and speech recognition models are usually very similar and it is not uncommon that a model that is shown to be successful in one of these two fields is later used by the other. Therefore, our techniques or slight modifications of them can be applied to adaptation problems in speech recognition as well.

REFERENCES

1. O'Cinneide, D. D. . G. M., A., "A Brief Introduction to Speech Synthesis and Voice Modification", *Sounds Electric Conference*, 2007.

2. Dutoit, T., "High-quality text-to-speech synthesis : an overview", *Journal of Electrical & Electronics Engineering*, 1997.

3. Taylor, P., *Text-to-Speech Synthesis*, Cambridge University Press, New York, NY, USA, 1st edn., 2009.

4. Kratzenstein, C. G., *Tentamen resolvendi problema ab Acad. Petropolit. 1780 propositu qualis sit natura litterarum vocalium a, e, i, o, u*, 1781.

5. Karjalainen, M., "Review of Speech Synthesis Technology", *Helsinki University of Technology, Department of Electrical and Communications Engineering*, 1999.

6. Stewart, J. Q., "An electrical analogue of the vocal organs", *Nature*, Vol. 110, No. 2757, p. 311, 1922.

7. Dudley, H., R. Riesz and S. Watkins, "A synthetic speaker", *Journal of the Franklin Institute*, Vol. 227, No. 6, pp. 739–764, 1939.

8. Teranishi, R. and N. Umeda, "Use of pronouncing dictionary in speech synthesis experiments", *Proc. Int. Congr. Acoust. B-5-2, Tokyo, Japan*, 1968.

9. Umeda, N., E. Matsui, T. Suzuki and H. Omura, "Synthesis of fairy tales using an analog vocal tract", *Proceedings of 6th International Congress on Acoustics*, pp. B159–162, 1968.

10. Kurzweil, R., "The Kurzweil reading machine: A technical overview", *Science, Technology and the Handicapped*, pp. 3–11, 1976.

11. Alan W Black, K. T., Heiga Zen, "Statistical Parametric Speech Synthesis", *Speech Communication,Volume 51, Issue 11, November 2009, Pages 1039–1064*, 2009.

12. Hunt, A. J. and A. W. Black, "Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database", *Proceedings of the Acoustics, Speech, and Signal Processing, 1996. On Conference Proceedings., 1996 IEEE International Conference - Volume 01*, ICASSP '96, pp. 373–376, IEEE Computer Society, Washington, DC, USA, 1996, `http://dx.doi.org/10.1109/ICASSP.1996.541110`.

13. Bennett, C. L., "Large Scale Evaluation of Corpus-based Synthesizers: Results and Lessons from the Blizzard Challenge", *Interspeech, 2005, pp. 105–108.*, 2005.

14. Alan W Black, C. L. B., "The Blizzard Challenge 2006", *Blizzard Challenge Workshop, 2006.*, 2006.

15. K. Tokuda, T. T. H. Z. H. Y., Y. Nankaku and K. Oura, "Speech synthesis based on hidden markov models", *Proceedings of the IEEE, 101(5):1234–1252*, 2013.

16. Yoshimura, T., K. Tokuda, T. Masuko, T. Kobayashi and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis", *Sixth European Conference on Speech Communication and Technology*, 1999.

17. H. Kawahara, I. M.-K. and A. Cheveigne, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based f0 extraction: possible role of a repetitive structure in sounds", *Speech Communication, vol. 27, pp. 187–207*, 1999.

18. O. Karaali, G. C. and I. Gerson, "Speech synthesis with neural networks", *Proc. World Congress on Neural Networks, 1996, pp. 45–50.*, 1996.

19. Raina, R., A. Madhavan and A. Y. Ng, "Large-scale Deep Unsupervised Learning Using Graphics Processors", *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 873–880, ACM, New York, NY, USA,

2009, http://doi.acm.org/10.1145/1553374.1553486.

20. H. Zen, A. S. and M. Schuster, "Statistical parametric speech synthesis using deep neural networks", *Proc. ICASSP, 2013, pp. 7962–7966*, 2013.

21. Z.-H. Ling, L. D. and D. Yu, "Modeling spectral envelopes using restricted Boltzmann machines and deep belief networks for statistical parametric speech synthesis", *EEE Trans. Acoust. Speech Lang. Pro- cess., vol. 21, no. 10, pp. 2129–2139*, 2013.

22. S.-Y. Kang, X.-J. Q. and H. Meng, "Multi-distribution deep belief network for speech synthesis", *Proc. ICASSP, 2013, pp. 8012–8016*, 2013.

23. R. Fernandez, B. R., A. Rendel and R. Hoory, "f0 contour prediction with a deep belief network-Gaussian process hybrid model", *Proc. ICASSP, 2013, pp. 6885–6889*, 2013.

24. H. Lu, S. K. and O. Watts, "Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis", *Proc. ISCA SSW8, 2013, pp. 281–285*, 2013.

25. Qian, Y., Y. Fan, W. Hu and F. K. Soong, "On the training aspects of Deep Neural Network (DNN) for parametric TTS synthesis", *ICASSP*, 2014.

26. Mohammadi, A., S. S. Sarfjoo and C. Demiroglu, "Eigenvoice speaker adaptation with minimal data for statistical speech synthesis systems using a MAP approach and nearest-neighbors", *IEEE/ACM Trans. Audio, Speech & Language Processing*, Vol. 22, No. 12, pp. 2146–2157, 2014, http://dx.doi.org/10.1109/TASLP.2014.2362009.

27. Mohammadi, A. and C. Demiroglu, "Nearest neighbor approach in speaker adaptation for HMM-based speech synthesis", *21st Signal Processing and Communications Applications Conference, SIU 2013, Haspolat, Turkey, April 24-26, 2013*,

pp. 1–4, 2013, http://dx.doi.org/10.1109/SIU.2013.6531576.

28. Tamura, M., T. Masuko, K. Tokuda and T. Kobayashi, "Speaker adaptation for HMM-based speech synthesis system using MLLR", *the third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*, 1998.

29. M. Tamura, K. T., T. Masuko and T. Kobayashi, "Adaptation of pitch and spectrum for HMM-based speech synthesis using MLLR", *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing 2:805-808 · January 2001*, 2001.

30. Yamagishi, J., T. Kobayashi, Y. Nakano, K. Ogata and J. Isogai, "Analysis of Speaker Adaptation Algorithms for HMM-Based Speech Synthesis and a Constrained SMAPLR Adaptation Algorithm", *Trans. Audio, Speech and Lang. Proc.*, Vol. 17, No. 1, pp. 66–83, Jan. 2009, http://dx.doi.org/10.1109/TASL.2008.2006647.

31. Yamagishi, J. and T. Kobayashi, "Average-Voice-Based Speech Synthesis Using HSMM-Based Speaker Adaptation and Adaptive Training", *IEICE - Trans. Inf. Syst.*, Vol. E90-D, No. 2, pp. 533–543, Feb. 2007, http://dx.doi.org/10.1093/ietisy/e90-d.2.533.

32. Gales, M., "Cluster Adaptive Training Of Hidden Markov Models", *IEEE Transactions on Speech and Audio Processing*, Vol. 8, pp. 417–428, 1999.

33. H. Zen, S. B. M. J. F. G. K. K. S. K., N. Braunschweiler and J. Latorre, "Statistical Parametric Speech Synthesis Based on Speaker and Language Factorization", *IEEE Trans. Audio, Speech, Lang. Process., vol. 20, no. 6, pp. 1713–1724, Aug. 2012*, 2012.

34. J. Latorre, M. J. F. G. L. C. K. K. C. K. K., V. Wan and M. Akamine, "Speech factorization for HMM-TTS based on cluster adaptive training", *Proc. Interspeech, 2012*, 2012.

35. V. Wan, K. K. C. L. C. M. J. F. G. H. Z. K. K., J. Latorre and M. Akamine, "Combining multiple high quality corpora for improving HMM-TTS", *Proc. Interspeech, 2012*, 2012.

36. Sin-Horng Chen, S.-H. H. and Y.-R. Wang, "An RNN-based prosodic information synthesizer for Mandarin text-to-speech", *IEEE Transactions on Speech and Audio Pro- cessing, vol. 6, no. 3, pp. 226–239, 1998*, 1998.

37. Achanta, S., T. Godambe and S. V. Gangashetty, "An investigation of recurrent neural network architectures for statistical parametric speech synthesis", *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pp. 859–863, 2015.

38. van den Oord, A., S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio", *Arxiv*, 2016, `https://arxiv.org/abs/1609.03499`.

39. Wang, Y., R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: A Fully End-to-End Text-To-Speech Synthesis Model", *arXiv preprint arXiv:1703.10135*, 2017.

40. Arik, S. O., M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, J. Raiman, S. Sengupta and M. Shoeybi, "Deep Voice: Real-time Neural Text-to-Speech", *CoRR*, Vol. abs/1702.07825, 2017, `http://arxiv.org/abs/1702.07825`.

41. Potard, B., P. Motlicek and D. Imseng, *Preliminary Work on Speaker Adaptation for DNN-Based Speech Synthesis*, Idiap-RR Idiap-RR-02-2015, Idiap, 1 2015.

42. Fan, Y., Y. Qian, F. K. Soong and L. He, "Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis", *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pp. 4475–4479, 2015,

http://dx.doi.org/10.1109/ICASSP.2015.7178817.

43. Pascual, S. and A. Bonafonte, "Multi-output RNN-LSTM for multiple speaker speech synthesis and adaptation", , 2016.

44. Chen, L.-H., Z.-H. Ling, L.-J. Liu and L.-R. Dai, "Voice Conversion Using Deep Neural Networks with Layer-wise Generative Training", *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, Vol. 22, No. 12, pp. 1859–1872, Dec. 2014, http://dx.doi.org/10.1109/TASLP.2014.2353991.

45. Wu, Z., P. Swietojanski, C. Veaux, S. Renals and S. King, "A study of speaker adaptation for DNN-based speech synthesis", *Interspeech*, 2015.

46. Fan, Y., Y. Qian, F. K. Soong and L. He, "Unsupervised speaker adaptation for DNN-based TTS synthesis", *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pp. 5135–5139, 2016, http://dx.doi.org/10.1109/ICASSP.2016.7472656.

47. Wu, Z. and S. King, "Investigating gated recurrent neural networks for speech synthesis", *CoRR*, Vol. abs/1601.02539, 2016, http://arxiv.org/abs/1601.02539.

48. Jokinen, E., P. Alku and M. Vainio, "Comparison of post-filtering methods for intelligibility enhancement of telephone speech.", *EUSIPCO*, pp. 2333–2337, 2012.

49. Tokuda, K., T. Kobayashi, T. Masuko, T. Kobayashi and T. Kitamura, "Speech Parameter Generation Algorithms for HMM-Based Speech Synthesis", *Proc. ICASSP*, pp. 1315–1318, 2000.

50. Toda, T. and K. Tokuda, "A speech parameter generation algorithm considering global variance for HMM-based speech synthesis", *IEICE TRANSACTIONS on Information and Systems*, Vol. 90, No. 5, pp. 816–824, 2007.

51. Chen, L.-H., T. Raitio, C. Valentini-Botinhao, J. Yamagishi and Z.-H. Ling, "DNN-

based stochastic postfilter for HMM-based speech synthesis.", *INTERSPEECH*, pp. 1954–1958, 2014.

52. Muthukumar, P. K. and A. W. Black, "Recurrent neural network postfilters for statistical parametric speech synthesis", *arXiv preprint arXiv:1601.07215*, 2016.

53. Kaneko, T., H. Kameoka, N. Hojo, Y. Ijima, K. Hiramatsu and K. Kashino, "Generative adversarial network-based postfilter for statistical parametric speech synthesis", *Proc. ICASSP*, Vol. 2017, pp. 4910–4914, 2017.

54. Young, S. J., D. Kershaw, J. Odell, D. Ollason, V. Valtchev and P. Woodland, *The HTK Book Version 3.4*, Cambridge University Press, 2006.

55. Tokuda, K., *HMM-based speech synthesis system (HTS)*, 2012, `http://hts.sp.nitech.ac.jp/`, accessed at June 2016.

56. Zen, H., "An example of context-dependent label format for HMM-based speech synthesis in English", *The HTS CMUARCTIC demo*, Vol. 133, 2006.

57. Martin, J. H. and D. Jurafsky, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, Pearson/Prentice Hall, 2009.

58. Tokuda, K., T. Kobayashi, T. Masuko and S. Imai, "Mel-generalized cepstral analysis-a unified approach to speech spectral estimation", *Third International Conference on Spoken Language Processing*, 1994.

59. Morise, M., "D4C, a band-aperiodicity estimator for high-quality speech synthesis", *Speech Communication*, Vol. 84, pp. 57–65, 2016.

60. Valentini-Botinhao, C., J. Yamagishi and S. King, "Evaluation of objective measures for intelligibility prediction of HMM-based synthetic speech in noise", *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference*

on, pp. 5112–5115, May 2011.

61. Rix, A. W., J. G. Beerends, M. P. Hollier and A. P. Hekstra, "Perceptual Evaluation of Speech Quality (PESQ)-a New Method for Speech Quality Assessment of Telephone Networks and Codecs", *Proceedings of the Acoustics, Speech, and Signal Processing, 200. On IEEE International Conference - Volume 02*, ICASSP '01, pp. 749–752, IEEE Computer Society, Washington, DC, USA, 2001, http://dx.doi.org/10.1109/ICASSP.2001.941023.

62. Remes, U., R. Karhila and M. Kurimo, "Objective evaluation measures for speaker-adaptive HMM-TTS systems", *8th ISCA Workshop on Speech Synthesis*, pp. 197–201, Barcelona, Spain, August 2013.

63. Rec, I., "P. 800.1, Mean opinion score (MOS) terminology", *International Telecommunication Union, Geneva*, 2006.

64. Munson, W. and M. B. Gardner, "Standardizing auditory tests", *The Journal of the Acoustical Society of America*, Vol. 22, No. 5, pp. 675–675, 1950.

65. Kraft, S. and U. Zölzer, "BeaqleJS: HTML5 and JavaScript based framework for the subjective evaluation of audio quality", *Linux Audio Conference, Karlsruhe, DE*, 2014.

66. Pascanu, R., T. Mikolov and Y. Bengio, "Understanding the exploding gradient problem", *CoRR*, Vol. abs/1211.5063, 2012, http://arxiv.org/abs/1211.5063.

67. Pascanu, R., T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks", *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 1310–1318, 2013, http://jmlr.org/proceedings/papers/v28/pascanu13.html.

68. Bengio, Y., P. Simard and P. Frasconi, "Learning Long-term Dependencies with Gradient Descent is Difficult", *Trans. Neur. Netw.*, Vol. 5, No. 2, pp. 157–166,

Mar. 1994, http://dx.doi.org/10.1109/72.279181.

69. Hochreiter, S. and J. Schmidhuber, "Long short-term memory", *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

70. Fernandez, R., A. Rendel, B. Ramabhadran and R. Hoory, "Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks", *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pp. 2268–2272, 2014.

71. Fan, Y., Y. Qian, F. Xie and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks", *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pp. 1964–1968, 2014.

72. Zen, H. and H. Sak, "Unidirectional Long Short-Term Memory Recurrent Neural Network with Recurrent Output Layer for Low-Latency Speech Synthesis", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4470–4474, 2015.

73. Torrey, L. and J. Shavlik, "Transfer learning", *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pp. 242–264, IGI Global, 2010.

74. Alpaydin, E., *Introduction to Machine Learning*, The MIT Press, 2nd edn., 2010.

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

76. Cho, K., B. van Merrienboer, D. Bahdanau and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches", *CoRR*, Vol. abs/1409.1259, 2014, http://arxiv.org/abs/1409.1259.

77. Nickolls, J., I. Buck, M. Garland and K. Skadron, "Scalable Parallel Programming with CUDA", *Queue*, Vol. 6, No. 2, pp. 40–53, Mar. 2008, http://doi.acm.org/10.1145/1365490.1365500.

78. Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions", *arXiv e-prints*, Vol. abs/1605.02688, May 2016, http://arxiv.org/abs/1605.02688.

79. Dehak, N., P. J. Kenny, R. Dehak, P. Dumouchel and P. Ouellet, "Front-End Factor Analysis for Speaker Verification", *Trans. Audio, Speech and Lang. Proc.*, Vol. 19, No. 4, pp. 788–798, May 2011, http://dx.doi.org/10.1109/TASL.2010.2064307.

80. 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.

81. 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)*, pp. 1–38, 1977.

82. 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.

83. Karpathy, A., J. Johnson and F. Li, "Visualizing and Understanding Recurrent Networks", *CoRR*, Vol. abs/1506.02078, 2015, http://arxiv.org/abs/1506.02078.

84. Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Nets", *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2672–2680, MIT Press, Cambridge, MA, USA, 2014,

http://dl.acm.org/citation.cfm?id=2969033.2969125.

85. Saito, Y., S. Takamichi and H. Saruwatari, "Statistical Parametric Speech Synthesis Incorporating Generative Adversarial Networks", *CoRR*, Vol. abs/1709.08041, 2017, http://arxiv.org/abs/1709.08041.

86. Snyder, D., D. Garcia-Romero, G. Sell, D. Povey and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition", *Submitted to ICASSP*, 2018.