SPEAKER ADAPTATION WITH DEEP LEARNING FOR TEXT-TO-SPEECH SYNTHESIS SYSTEMS

A Thesis

by

Eray Eren

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
December 2021

SPEAKER ADAPTATION WITH DEEP LEARNING FOR TEXT-TO-SPEECH SYNTHESIS SYSTEMS

Approved by:

Assoc. Prof. Dr. Cenk Demiroğlu,
Advisor
Department of Electrical and Electronics Engineering
Özyeğin University

Asst. Prof. Dr. Enis Kayış
Department of Industrial Engineering
Özyeğin University

Prof. Dr. Ümit Güz
Department of Electrical and Electronics Engineering
Işık University

Date Approved: 28 December 2021

To my dear family: Tülin Eren, Behiç Eren, Oğuz Çimir, Nurbahar Çimir, Kaan Çimir and Zeynep Çimir.

ABSTRACT

End-to-end (e2e) speech synthesis systems have become popular with the recent introduction of letter-to-spectrogram conversion systems, such as Tacotron, that use encoder-decoder-based neural architectures. Even though those sequence-to-sequence systems can produce mel-spectrograms from the letters without a text processing frontend, they require substantial amounts of well-massaged, labelled audio data that have high SNR and minimum amounts of artifacts. These data requirements make it difficult to build end-to-end systems from scratch especially for low-resource languages. Moreover, most of the e2e systems are not designed for devices with tiny memory and cpu resources. Here, we investigate using a traditional deep neural network (DNN) for acoustic modelling together with a postfilter that improves the speech features produced by the network. The proposed architectures were trained with the relatively noisy, multi-speaker, Wall Street Journal (WSJ) database and tested with unseen speakers. The thin postfilter layer was adapted with minimal data to the target speaker for testing. We investigated several postfilter architectures and compared them with both objective and subjective tests. Fully-connected and transformer-based architectures performed the best in subjective tests. The transformer-based architecture performed the best in objective tests. Moreover, it was faster than the other architectures both in training and inference speeds.

ÖZETÇE

Tacotron gibi son zamanlarda çıkan harften-spektrograma dönüşüm sistemleriyle gizyazar-gizçözer temelli sinir ağı mimarilarini kullanan uçtan-uca (uu) ses sentezi sistemleri popüler hale geldi. Bu diziden-diziye sistemler, metin işleyen önyüze gerek duymadan mel-spektrogramları üretebilse de; yüklü miktarda, iyi yoğrulmuş, yüksek sinyal-gürültü oranlı ve minimum düzeyde kusurlu etiketli ses verisine ihtiyaç duymaktadır. Bu veri ihtiyacı bilhassa düşük kaynağa sahip diller için uçtan-uca sistemleri inşa etmeyi zor duruma getirmektedir. Dahası, uu sistemlerin birçoğu düşük hafıza ve CPU kaynaklarına sahip sistemler için tasarlanmamıştır. Biz bu çalışmada, geleneksel derin sinir ağı tarafından üretilen konuşma özniteliklerini iyileştiren postfiltrelerin bu sinir ağlarıyla beraber kullanımlarının akustik modellemeye olan etkisini araştırdık. Önerilen sistemler görece gürültülü Wall Street Journal (WSJ) verisiyle eğitilip görülmemiş konuşmacılar için test edildi. İnce postfiltre katmanı minimum veri ile hedef konuşmacının testi için uyarlandı. Birkaç farklı postfiltre mimarisini araştırdık ve bunları taraflı ve tarafsız testlerle karşılaştırdık. Tam-bağlı ve transformer temelli mimariler taraflı testlerde en iyi sonucu verdi. Transformer temelli mimari tarafsız testlerde en iyi sonucu verdi. Ayrıca, diğer mimarilerden hem eğitimde hem de tahminde daha hızlıydı.

ACKNOWLEDGEMENTS

I would like to express my special thanks to my supervisor Prof. Cenk Demiroğlu for his constant support during my Master of Science. He is truly a great teacher, academician and mentor. It has been an excellent journey with him, and working with him was a great pleasure. I want to thank my teacher Prof. Emre Sefer for his huge support during my coursework, and my PhD applications. I am grateful for his generous patience and knowledge. I want to thank my teacher Prof. Enis Kayış for his excellent course, his patience, being a great teacher, and for his huge support during my PhD applications. I also want to thank Prof. Ümit Güz for his contributions and kind acceptance to be in my thesis jury. I would like to thank Prof. Okan Örsan Özener for his great course, and his friendly-attitude towards students. I am grateful to Prof. Ethem Alpaydın for the knowledge I obtained from his great machine learning and deep learning courses. I also thanful for Prof. Fatih Uğudağ's support during my Master of Science.

I am also grateful for Gizem Bakır's generous help and guidance during my studies. I would like to thank my excellent colleagues Huda Barakat and Çağıl Süslü.

Last but not least, I want to give my special thanks to my family, and my long time friends Ozan Genç, Hamdi Erkut, Yiğitcan Sezgin, Barış Özmen, Kerem Özkılıç, Sercan Esen, Göksu Öztürk, Okan Ulusoy, İhsan Serdaroğlu and Atakan Yüksel for their support during my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

ASR Automatic Speech Recognition

BAP Band Aperiodicity

BCE Binary Cross Entropy

CI Cluster-based Initialization

CNN Convolutional Neural Network

CPU Central Processing Unit

DNN Deep Neural Network

$E2E$ End-to-End

FC Fully-connected

GAN Generative Adversarial Network

GPU Graphical Processing Unit

HMM Hidden Markov Model

Hz Hertz

$LF0$ Log-Fundemental Frequency

LPC Linear Predictive Coding

$LSTM$ Long-Short Term Memory

MCD Mel Cepstral Distortion

MGC Mel-Generalized Cepstrum

$MLPG$ Maximum Likelihood Paraneter Generation

MOS Mean Opinion Score

MSE Mean Square Error

NLP Natural Language Processing

RBM Restricted Boltzmann Machine

RNN Recurrent Neural Network

RTF Real Time Factor

SI Speaker-Independent

SNR Signal-to-Noise Ratio

$T - FNN$ Transformer-based Feedforward Neural Network

TTS Text-to-Speech

VAE Variational Autoencoder

VUV Voiced/Unvoiced

WSJ Wall Street Journal

CHAPTER I

INTRODUCTION

1.1 Background

End-to-end (e2e) speech synthesis paradigm revolutionized the field by enabling natural-sounding speech synthesis without requiring linguistic features [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Using the e2e neural synthesis paradigm, for the first time, it became possible to generate speech that is indistinguishable from natural speech.

E2e synthesis systems have two main deep learning blocks. The first block converts the character (or sometimes phoneme) sequences into mel-spectrograms. Tacotron1 [1] and Tacotron2 [2] are the pioneering examples. The key idea is to use an attention mechanism to decide which character is most relevant to each of the mel-spectrogram frames and assign weights to the character embeddings accordingly.

Even though the Tacotron approach works well, it is slow because of its recursive architecture. Transformer [13], which is a non-recursive and attentive neural network, is proposed in [14] for mel-spectrogram generation. Fastspeech [3], which is also transformer-based and does not attention for alignment, is designed for the same goal. Fastspeech2 [4] removed the teacher-student distillation setting used in Fastspeech. Unlike recursive networks in [1, 6, 7], fully convolutional neural architecture is utilized for faster conversion of text to mel spectrogram in [8, 9]. Non-autoregressive flow-based neural models are proposed in [11, 12] for faster generation of the mel spectrogram from the given text.

Recently several Tacotron variants have been proposed to tackle the issues of Tacotron1 and Tacotron2 [15, 16, 17]. These issues include word-repetitions, word-skippings, non-robust attentive duration modeling, relatively slow training

and inference. Non-robust attentive duration modeling is addressed in [15]. To mitigate the problem, they replaced the attention mechanism with an explicit duration model. The duration model allows for the phoneme and sentence level control of the durations. They presented a VAE-based approach for the semi-supervised or unsupervised training of the durations when the duration information is limited. The vanilla upsampling of the Tacotron is substituted with the Gaussian upsampling in order to improve the naturalness.

A parallel Tacotron [16] architecture is suggested for the non-autoregressive training and inference. To capture the locality of the text-to-speech better and increase the efficiency, they compare an alternative with lightweight convolutions [18] to the vanilla self-attention mechanism of the transformers. Similar to [3, 4, 15], an explicit duration model is incorporated into the TTS architecture. A variational encoder is utilized to ease the one-to-many mapping problem of the TTS, and enhance the speech quality. In order to alleviate the external aligner requirement of [16], a supervision-free duration model is proposed in [17] using Soft Dynamic Time Warping [19].

Furthermore, in [5], an attention-based recursive sequence-to-sequence network is used to convert text to acoustic features. A flow-based autoregressive architecture is proposed in [10]. [20] is an autoregressive neural text to speech model as well. However, it uses shifting memory buffers with fully-connected networks for memory efficiency.

1.2 Other Efficiency-Oriented Text-to-Speech Models

Time inefficient autoregressive models lead to increased training and inference durations. Similar to [3, 4, 16, 17], this issue has been addressed by the use of non-autoregressive architectures in [21, 22, 23, 24]. Like Fastspeech1, [23] also uses a student-teacher network for learning the durations. A Fastspeech1 variant is presented in [24]. Differently from Fastspeech1, their model extracts the durations on the fly during the training enabling joint training. The architecture

is fully-convolutional, and the synthesis is real-time on a CPU. Although architecture in [21] is similar to Fastspeech1, their proposed duration model does not require an external aligner during training unlike [3, 4, 23]. They make use of a mix density network for the duration extraction in the training. Another non-autoregressive and efficient TTS model that does not require an external aligner is [22]. A monotonic alignment approach is proposed for the alignment modeling. Unlike the multi-phase training in [21], [22] has a joint training process. They compared convolution-based and flow-based neural TTS models, and showed that fully-convolutional architecture is more efficient than the flow-based counterpart with a higher mean opinion score (MOS).

1.3 Neural Vocoders

Once the mel-spectrogram is generated given an utterance, a second block is needed that will convert the mel-spectrogram to the audio waveform. WaveNet [25] is the first deep learning based autoregressive method that could successfully achieve that goal. It also paved way for substantial research in the neural vocoder field with a particular focus on real-time synthesis. One of the first fruits of that was WaveRNN [26] that could perform similar to WaveNet but could run real-time on GPU. Another autoregressive algorithm that gained interest is LPCNet [27], which uses traditional linear predictive coding (LPC) parameters of speech to synthesize speech in real-time on a CPU with a small footprint.

A variety of fast non-autoregressive vocoders that use generative adversarial networks (GANs) followed that. MelGAN [28], multi-band MelGAN [29], GAN-TTS [30], Parallel Wavegan [31], EATS [32], and HiFi-GAN [33] are examples of those fast algorithms. Similar to these non-autoregressive approaches, [34] introduces a neural acoustic model with differentiable DSP [35] for synthesizing controllable speech. Similar to [36, 37], a parallelizable flow-based acoustic model is proposed in [38] without the need for autoregressive modeling for faster inference. Unlike [36, 37], a student-teacher framework was not used in [38]. Aiming for faster

inference, in [39, 40], non-autoregressive neural acoustic source-filter models are used for creating audio waveforms. Other neural autoregressive methods include [41, 42].

A recent diffusion probabilistic text-to-speech model, [43] enables a trade-off between autoregressive and non-autoregressive neural vocoders. This model iteratively refines a Gaussian white noise with a fixed number of iterations. The number of iterations allows for a trade-off between the sample quality and inference speed.

1.4 Single-Stage End-to-End TTS

Even though a variety of neural TTS models use the term end-to-end to demonstrate their training procedures, the majority requires two-stage training. As described in Section 1.1, the first stage converts text to intermediate acoustic representations. Then, these acoustic features are used to predict the waveform. Conventionally, this stage is trained separately. Recent works in the TTS field include the direct waveform generation from the text with a single training. The work in Fastspeech2 has both single-stage and two-stage waveform generation architectures. The single-stage version [4], Fastspeech2s, uses an adversarial WaveNet-based network as the decoder with non-causal convolutions to convert intermediate acoustic representations into waveform while keeping the rest of network the same with the two-stage Fastspeech2 model. Although Fastspeech2s is fully e2e, it has a lower MOS than the Fastspeech2.

Another non-autoregressive and fully e2e model, EATS [32], also makes use of an adversarial decoder, GAN-TTS, to predict the waveform. Similar to single-stage and two-stage alternatives in Fastspeech2, [22] has also two alternatives. Similar to previous works, this fully e2e model has an adversarial non-autoregressive decoder, MelGAN, to generate the waveform. In addition to the inference speedup, the single-stage model has similar MOS with the two-stage alternatives unlike Fastspeech2s. A flow-based and mel-spectrogram-free fully e2e model is proposed in [44]. Similar to diffusion probabilistic models [43, 45], the flow architecture also

converts the random noise into the waveform iteratively. However, the flow-based decoder is autoregressive. Conversion of text to the intermediate representations is implemented with a Tacotron-based encoder. Like mel-spectrogram-free [44], [45] also does not require hand-designed intermediate representations for single-stage text-to-speech. The model consists of a Tacotron2-based encoder and WaveGrad [43] decoder.

Unlike the other single-stage TTS models, [46] does not require training with short audio clips. They make use of VAE and stochastic duration predictor to overcome the one-to-many mapping problem. Variational inference is augmented with the normalizing flow, and the training is adversarial.

1.5 Text-to-Speech Synthesis Loss Metrics

Several loss metrics have been proposed for non-autoregressive neural text-to-speech synthesis (NTTS) models for the high-fidelity speech. Since simple manhattan or euclidean loss metrics, such as mel-spectrogram or STFT distances with the ground-truth counter-parts, cannot make up for the better inductive bias in the AR models; more complex losses, which include multi-resolution acoustic representation objectives together with adversarial losses, have been devised in [28, 29, 30, 31, 32, 33]. However, manhattan or euclidean training loss metrics, and objective evaluation metrics such as mel-cepstral-distortion (MCD) are not well correlated with the generated speech quality as discussed in [47]. As a result, there is a research gap for NTTS models which requires better loss metrics which can lead to better convergence of these models and fewer data requirements.

1.6 Semi-supervised or Unsupervised Trainings in Text-to-Speech

1.6.1 ASR-TTS Feedback Loops

Recently, these gaps have been addressed by 2 approaches. The first approach is utilizing the ASR-TTS feedback loop [48, 49, 50, 51, 52]. ASR-TTS feedback loop aims to enhance ASR and TTS systems through joint training ASR and

TTS models, and through incorporating multiple objectives with the ASR-TTS feedback loop. Usually, these systems are trained in a semi-supervised fashion. Paired $< text, audio >$ data is used for the supervised training, and unpaired data from the ASR-TTS loop is used for the unsupervised training part. This way ASR and TTS models can be trained with less paired data, and they can get the same levels of quality as the baseline models that are trained with more paired data.

1.6.2 Self-supervised Learning Representations

The second approach includes the incorporation of acoustic or linguistic representations that make use of the pre-training with untranscribed speech data [53, 47, 54, 55, 56, 57, 58]. These models are also called self-supervised learning representations (SSLR) in the literature. Like the ASR-TTS feedback loop approach, usually, linguistic units or acoustic representations from these methods are learned in a semi-supervised fashion to improve speech synthesis or ASR. NTTS models are conditioned with these vectors from these pre-trained models to improve the prosody, or quality of the generated speech with less data. Unlike the other semi-supervised NTTS systems, in [58], fully unsupervised speech synthesis is used for synthesizing intelligible speech.

Although ASR-TTS feedback loop systems can take the ASR feedback into account, they cannot generalize well, and the robustness of the models together with the generated speech quality is far from being perfect [55], especially for the multi-speaker cases. Additionally, almost all of the research focused on mono-lingual NTTS models. Unsupervised pre-training approaches described above, can lead to more robust NTTS, and higher quality speech [56]. However, these models are usually more oriented towards conditioning the NTTS with the extracted linguistic units rather than replacing the conventional representations such as mel-spectrograms. In addition, poor speech quality correlation problem arising from manhattan or euclidean loss metrics with these representations remains. Moreover, almost all of the methods focused on mono-lingual cases like the ASR-TTS feedback loop methods.

More recently, a textless NLP model, GSLM [58], which utilizes self-supervised learning representations (SSLR) is used for speech synthesis. GSLM uses CPC [59], wav2vec 2.0 [60], HuBert [61] as SSLR models Unlike the other SSLR models, these models are trained with huge multi-speaker datasets. This can dramatically increase their generalization and out-of-domain capacities. Furthermore, these SSLR models can produce robust fixed-dimensional frame-level representations replacing the conventional representations. Nevertheless, the GSLM model is not used for a TTS task, and the speech synthesis experiment is limited to the monolingual case.

CHAPTER II

RELATED WORK

2.1 Speaker Adaptation

2.1.1 Speaker Adaptation Background

The other heavily-investigated aspect of e2e synthesis systems is the ability to adapt to new speakers. Comparison of model adaptation and incorporation of i-vector [62] as speaker embeddings for speaker adaptation are conducted in [63]. Similar to i-vectors, d-vectors [64] are also shown to be effective for speaker adaptation [65]. In [66], speaker embeddings are combined with phoneme-level representations for adapting to unseen speakers. Also, for fine-tuning the average voice model, word embeddings are utilized in a recent study [67] together with phoneme and speaker embeddings.

In addition to speaker encodings, gender and age encodings are fine-tuned with small adaptation data without changing the multi-speaker TTS model weights in [68]. The effect of fine-tuning the speaker embeddings together with TTS model weights is explored in [69]. A more recent work [70] studies conditional layer normalization together with speaker-level, utterance-level, and phoneme-level representations in FastSpeech2-based TTS architecture. The network and speaker embeddings are optimized for the unseen speakers. In [71], an RNN-based speaker encoder is trained to minimize the identity and the speaker feature losses. Then, a one-shot speaker adaptation setting is used for predicting the acoustic features.

2.1.2 Cross-Lingual Speaker Adaptation

In [72], DNN-based SPSS is adapted for the unseen speaker using fine-tuning on the average voice model. Cross-lingual speaker adaptation of the same speaker is also explored for Chinese and English languages. In [73], a convolutional speaker

encoder is trained separately using a speaker verification model to extract speaker embeddings. Tacotron2-based architecture is employed to convert phonemes and speaker embeddings to mel-spectrograms. Both English and Chinese datasets are used in the training and the adaptation of the base model to a new speaker.

Another cross-language speaker adaptation for TTS is implemented with 4 languages in [74]. An acoustic model consisting of fully-connected layers is trained with multiple speakers and multiple languages. Language codes, d-vectors, and linguistic features are used as features. A separate neural network consisting of fully-connected layers is used for duration modeling in [75]. Similar to the duration model, an average voice model with only fully-connected layers is trained. These models are fine-tuned for speaker adaptation.

2.1.3 VAE-based Speaker Adaptation

Variational autoencoders are also used to extract speaker embeddings for speaker adaptation [76]. In [76], a separately trained RNN-based VAE network is trained in a multi-speaker setting to predict duration, fundamental frequency, and energy. Mean speaker vectors from the encoder are used as speaker embeddings. Acoustic models are pretrained separately for a male single speaker and a female single speaker. The model that is initialized with the corresponding gender is fine-tuned with the unseen speaker's voice.

2.1.4 Few-shot Speaker Adaptation

Several zero-shot and few-shot speaker adaptation approaches, which utilize d-vector or x-vector variants, have been studied for text-to-speech synthesis recently. Zero-shot speaker adaptation techniques for speech synthesis are explored in [77, 78] by conditioning the TTS network with the speaker embeddings that are trained for speaker verification on large amounts of data. In [79], a Tacotron2-based architecture, which is conditioned on the speaker embeddings, is pretrained with a multi-speaker dataset. For the adaptation of the model to a new speaker, the pretrained TTS model and the speaker encoder model are fine-tuned with the new

speaker data, and hyperparameters are optimized with a Bayesian approach.

A semi-supervised approach in [80] makes use of pretrained transformer-based ASR and TTS models for the cases where text is not available for new speakers. ASR is used to extract text from the data and then, audio and ASR-generated text are used for adapting the TTS model. They show that this approach outperforms the zero-shot x-vector conditioning, and comparable with adaptation where paired data is available for the new speaker.

2.2 Neural Postfilters

Postfiltering is used for improving the output of the mel-spectrogram generating neural networks. A variety of DNN-based postfiltering approaches have been investigated for speech synthesis. In [81], RBM-based postfiltering is used for HMM-based speech synthesis. Conditional generative adversarial network (CGAN) [82] with CNNs was shown to be effective for enhancing the spectral texture for speech synthesis in [83, 84]. Other GAN-based postfiltering studies [85, 86] include the use of cycle-consistent adversarial networks(CycleGAN) [87].

Inspired by CycleGAN, in [88], a cyclical postfiltering approach without adversarial training is proposed. In [89], neural vocoder and WaveNet-based postfilter are jointly trained by the combination of multi-scale STFT loss and negative log-likelihood of the training data. They show that fewer parameters are needed for the neural vocoder with the postfiltering approach.

CHAPTER III

PROBLEM STATEMENT

Despite the success of the e2e synthesis systems, they have several problems that still need to be solved. One of the problems is the ability to run on offline devices with tiny memory and cpu resources, which is not possible for most of the existing e2e systems. Moreover, even though adaptation with few shots is possible if the speaker is in the training set (seen speaker), performance degrades significantly if the target is an unseen speaker [77, 78, 90]. Furthermore, collecting tens of hours of studio-quality data to train e2e systems is not easy especially for low-resource languages. Data problem is even more challenging when developing good quality datasets with many speakers.

Here, we step back from the end2end approach to a more classic, fast architecture for generating mel-spectrograms. Not only those architectures are faster and have smaller footprints, but also they can be trained with few hours of data that does not have to be carefully recorded in studio with strict requirements on silence durations as typically needed to train e2e systems. Because the speech quality degrades with those lightweight networks, we investigated postfiltering techniques to improve the quality. Four different postfilter architectures were compared. The novel transformer-based architecture was shown to be fastest and produced the best speech quality. Moreover, the overall architecture was shown to be substantially faster with far smaller parameters compared to two of the popular e2e systems.

Another novelty in our work is that the proposed postfilters are speaker-adaptive and they can adapt to new speakers with only a few seconds of data. Thus, speaker adaptation was done at the postfiltering block in addition to the

transfer learning at the speaker-independent block in the synthesis pipeline. Because that block is a thin network, adaptation with very limited data became possible.

The proposed system is faster and has substantially smaller footprint compared to e2e systems and can run real-time on many embedded devices with tiny resources. To complete our synthesis pipeline, generated spectrograms are converted to waveform using the LPCNet vocoder, which was also designed for low-resource devices.

CHAPTER IV

PROPOSED SYSTEMS

4.1 System Overview

An overview of the proposed system is shown in Figure 1. Linguistic features are extracted from the text and fed into the speaker-independent (SI) acoustic model together with the i-vector of the target speaker. The acoustic features are then generated with the maximum likelihood parameter generation (MLPG) algorithm [91] and smooth mel-generalized cepstrum (MGC) parameters are input to a postfilter for enhancement. Enhanced features are then converted into the LPCNet feature set and the waveform is synthesized with the LPCNet vocoder.

Postfiltering algorithms that were investigated in this work are described below. Speaker adaptation methods with the postfilters are described in Section 5.

4.2 Postfilters

Four different postfilters are investigated for enhancing the MGC features: fully-connected network, LSTM network, CNN network, and transformer-based feedforward network. The number of layers and the number of nodes/layers for each network were tuned independently using the objective mel-cepstral distortion (MCD) scores.

All postfilters were designed to predict the enhanced MGC features. Even though inputs of the postfilters differ based on their architectures, they all have the MGC parameters as input. Cepstral coefficients are usually incorporated with delta (differential) and double-delta (acceleration) features to add dynamic information to the static features. Although networks, such as RNN and transformer, which can inherently model the time independent information may not need this extra information, dynamic features can lead to better convergence or enhance the networks that do not have inherent time dependency. Since the conventional

magnitude spectrum ignores the phase, another hypothesis [92] indicates that dynamic features can help to recover the phase information. We also incorporated differential (Equation 1) and acceleration (Equation 2) features as follows:

$$\Delta\hat{x}_t = -0.5\hat{x}_t + 0.5\hat{x}_{t-2} \tag{1}$$

$$\Delta\Delta\hat{x}_t = (\hat{x}_t - \hat{x}_{t-1}) - (\hat{x}_{t-1} - \hat{x}_{t-2}) = \hat{x}_t - 2\hat{x}_{t-1} + \hat{x}_{t-2} \tag{2}$$

$$X = \left[\; \hat{x} \;\middle|\; \Delta\hat{x} \;\middle|\; \Delta\Delta\hat{x} \; \right] \tag{3}$$

where $\hat{x} \in \mathbb{R}^{T \times D_{\hat{x}}}$ indicates the mel-generalized-cepstral coefficients (MGC) for all the frames in the given utterance. T and $D_{\hat{x}}$ symbols denote the temporal and cepstral coefficient dimension sizes, respectively, and $\hat{x}_t$ is the t^{th} frame of that vector. Equation 3 shows the overall cepstro-temporal feature matrix, $X \in \mathbb{R}^{T \times 3D_{\hat{x}}}$, that contains both static and dynamic features of the given utterance, and $\hat{x} \in \mathbb{R}^{T \times D_{\hat{x}}}$, $\Delta\hat{x} \in \mathbb{R}^{T \times D_{\hat{x}}}$, $\Delta\Delta\hat{x} \in \mathbb{R}^{T \times D_{\hat{x}}}$ vectors are concatenated in the cepstral coefficient dimension. In order to smooth parameter trajectories maximum likelihood parameter generation (MLPG) algorithm is applied using the static and dynamic features.

In addition to the MGC parameters, the CNN, the LSTM and the fully-connected postfilters use state (s_t) and phoneme-level (p_t) features at each time t to enrich the context information available to the network. Those are both 1-of-K vectors representing the state number and the phoneme number at time t. The size of the p_t vector is D_p and the size of the s_t vector is D_s.

The postfilter architectures are described in more detail below.

4.2.1 Fully-connected Network

For the fully-connected network, aiming for enriched input which is text-aware, we concatenated cepstral coefficients with the phoneme and state information as

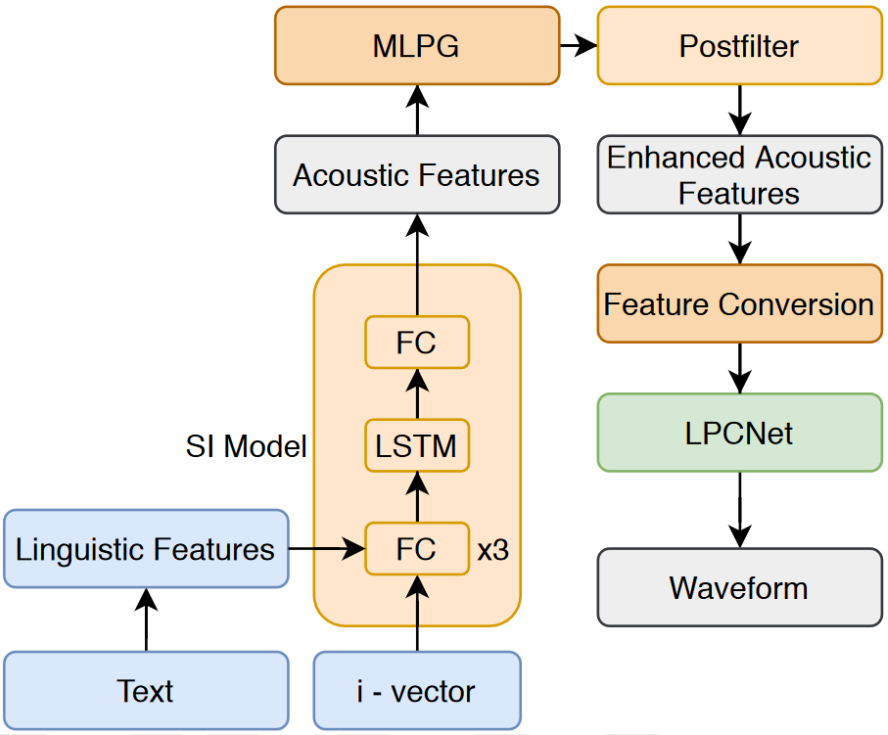


Figure 1: Overview of the proposed text-to-speech system.

shown in Equation 4, where the vector $\hat{C}_t \in \mathbb{R}^{D_X+D_p+D_s}$ is the enriched feature vector at the time t.

$$\hat{C}_t = \left[\begin{array}{c|c|c} X_t & p_t & s_t \end{array} \right] \tag{4}$$

Also, FC networks cannot model time dependencies inherently. Although dynamic features are provided, it can be difficult for a FC network to learn of the context information. Therefore, to further enhance the convergence of the low data training, we include previous and next frame's feature vectors to the current frame as follows:

$$C_t = \left[\begin{array}{c|c|c} \hat{C}_{t-1} & \hat{C}_t & \hat{C}_{t+1} \end{array} \right] \tag{5}$$

where $\hat{C}_t \in \mathbb{R}^{3(D_X+D_p+D_s)}$ is the resulting input feature vector. This postfilter, shown in Figure 2b, is a shallow feedforward model that consists of fully-connected layers. The number of parameters is 21k.

4.2.2 LSTM Postfilter

In a similar manner, the enriched feature matrix, $\hat{C} \in \mathbb{R}^{D_X+D_p+D_s}$, is used as input to the recurrent network for the given utterance. The recurrent network consists of an LSTM layer and a fully-connected layer as shown in Figure 2c. Due to the recurrent nature of the network, the input features are fed to network sequentially, one frame at a time. The output is $D_{\hat{x}}$-dimensional enhanced cepstral features like the FC postfilter. The number of parameters for the LSTM network is 39k.

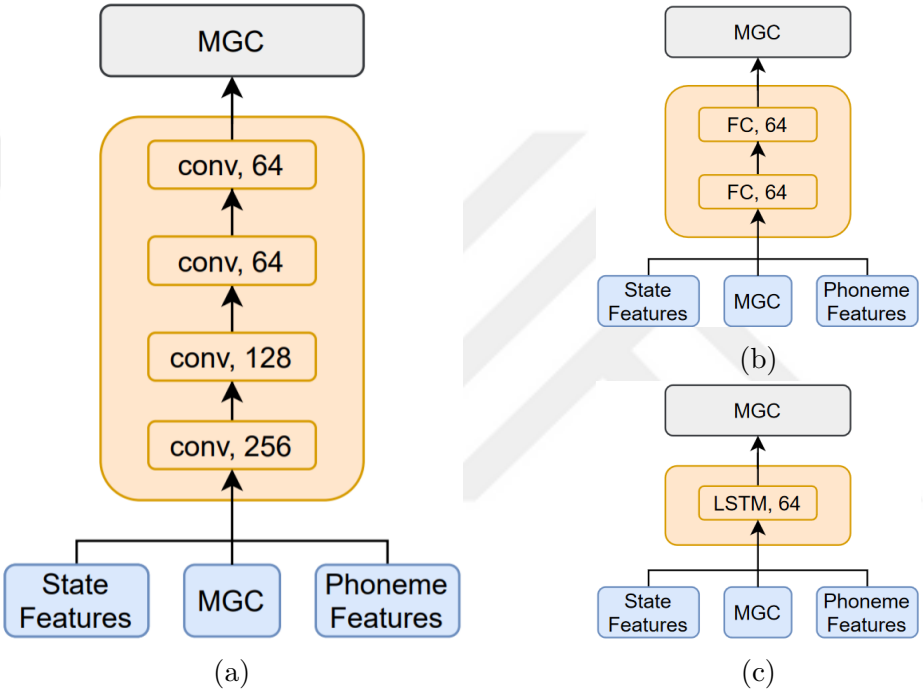


Figure 2: (a) CNN-based postfilter's input, output, and hidden layers are shown with the corresponding output channel dimensions. The kernel size is set to 3. (b) Fully-connected postfilter's input, output, and hidden layers are shown with the corresponding hidden layer unit sizes. (c) LSTM-based postfilter's input, output, and hidden layer are shown with the corresponding hidden layer unit size.

4.2.3 CNN Postfilter

A convolutional network that consists of 1-dimensional convolution blocks is used for the CNN postfilter to generate the enhanced cepstral features from the enriched feature matrix, $\hat{C} \in \mathbb{R}^{D_X+D_p+D_s}$, similar to the FC and LSTM-based postfilters. For the CNN postfilter, shown in Figure 2a, ReLU is used as the activation function; and kernel size of 3, and stride of 1 are used as the parameters of the 1D

convolutions. The number of parameters for the CNN postfilter is 159k.

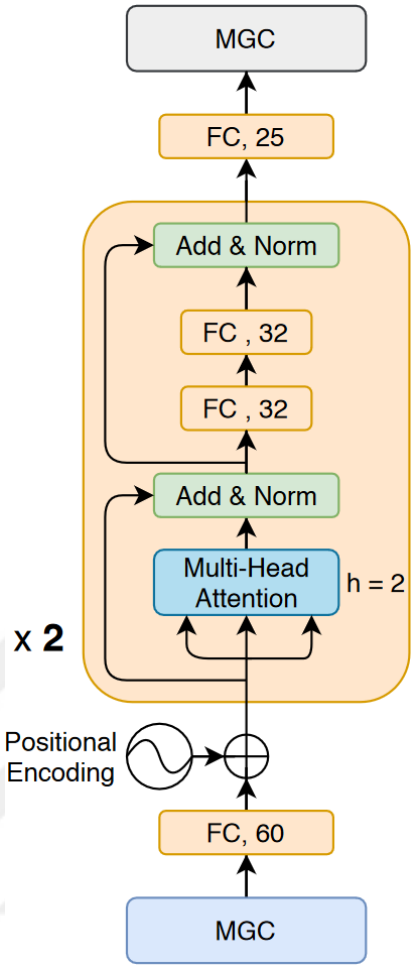


Figure 3: Transformer-based feedforward (T-FNN) postfilter.

4.2.4 Transformer-Based FFN (T-FNN) Postfilter

The architecture of the transformer-based [13] feedforward network (T-FNN) is shown in Figure 3. Only MGC features are the input of the network; i.e., state- or phoneme-level information were not used.

Unlike the other postfilters, the input to the system is the cepstro-temporal feature matrix, $X \in \mathbb{R}^{T \times D_X}$. Similar to the CNN-based postfilter, the whole utterance is represented as a single feature matrix in the transformer network. We chose only MGC features for T-FNN postfilter based on our empirical findings since we observed that using state- or phoneme-level information did not affect the postfilter performance. We leave theoretical investigation of the alternative options to future work.

The number of the transformer block is 2 for the transformer-based postfilter as shown in Figure 3. The number of attention heads, h, is 2. MGC inputs are followed by a fully-connected layer. Then, positional encoding is used in order to utilize position information. Layer normalization is used for the normalization of the network. The number of parameters for the T-FNN postfilter is 38k.

CHAPTER V

ADAPTATION

As a first step in the adaptation process, the SI model is adapted to the target speaker. To that end, the i-vector that is extracted from the target speaker is used as an input to the SI network. Moreover, transfer learning was used to adapt the LSTM and output layers of the network to the target speaker. Because we consider the case where minimal data is available for adaptation, learning rates for the other layers were set to zero to avoid overfitting as shown in Figure 4.

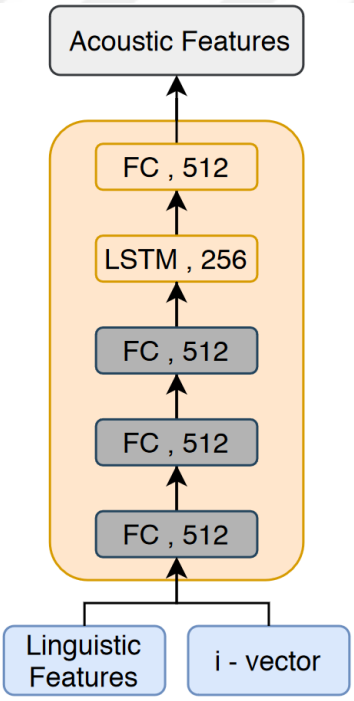


Figure 4: Adaptation of the speaker-independent system model. The first three fully-connected layers are fixed during speaker adaptation. Thus, only the LSTM layer and the fully-connected output layers are adapted.

Even though some gain was possible through adapting the base SI model to the target speakers, quality and similarity of the synthesized audio were still substantially lower than what is possible with the end-to-end systems. Thus, we investigated adaptation of the postfilters to the target speakers to further boost

the speaker similarity and quality of the synthesized speech. Speaker adaptation algorithms for the postfilters are described below.

The postfilter architectures proposed here are relatively low-complexity filters that can be trained with small amounts of data. We investigated three strategies to improve the effectiveness of the speaker-adapted postfilters. Those strategies are explained below.

5.1 Cluster-based Initialization

In a multispeaker dataset, there can be speakers with similar speech style, prosody, and fundemental frequency. Those parallel speech features of the speakers can be utilized. Therefore, instead of initializating the network with random weights and biases, we inherit a cluster-based initialization approach. Since the amount training data is scarce, a guided initialization can be critical in terms of faster convergence. With this motivation, the postfilters are pretrained with the similar speaker clusters using the training set. For the target speaker, the pretrained postfilter that is closest to that speaker's cluster is used as the starting point of the training.

Since i-vectors can encode the speaker characteristics, the training speakers were split into 5 clusters by the k-means clustering algorithm using those vectors. Euclidean similarity metric was used during clustering using i-vectors. Hence, the i-vector of target speaker is compared with the all cluster means. The most similar cluster is chosen for the initialization before the adaptation training. Cluster mean vectors were computed averaging the i-vectors of the training speakers in the clusters.

5.2 Adversarial Training

Adversarial training has been incorporated into many speech synthesis applications as described in Section 1.3, Section 1.4, Section 2.2. The waveform generation stage of the text-to-speech especially requires high fidelity synthesis. This requirements have been met with auxiliary losses and adversarial training in the

recent state-of-the-art neural vocoders. Adversarial single-stage text-to-speech and postfilters are also good examples of the GAN applications.

The GAN network that was used here is shown in Fig 5. GANs consist of two seperate networks: a generator (G) and a discrimininator (D). Without the adversarial training, i.e. D network is not available, , the training loss of the postfilter G is defined as follows:

$$L_G(X^{nat}, X^{gen}) = \frac{1}{T}\sum_{t=0}^{T}(X_t^{nat} - X_t^{gen})^2 \tag{6}$$

where $X^{nat} \in \mathbb{R}^{T\times D}$ is the natural and $X^{gen} \in \mathbb{R}^{T\times D}$ is the generated feature vector for the time step t.

In the adversarial approach, a discriminator network is used at the output of the generator. The discriminator decides whether the generated features are synthetic (fake) or natural (real). Thus, the discriminator enforces the generator to not only reduce the MSE loss but also generate features that are as close as possible to natural features.

The D network uses the binary cross entropy loss:

$$L_D(\hat{o}, o) = -\frac{1}{T}\sum_{t=0}^{T}\sum_{i=1}^{C=2} o_t^i log(\hat{o}_t^i) \tag{7}$$

where $\hat{o}_t \in [0, 1]$ is the prediction and $o_t \in \{0, 1\}$ is the target for the time step t. o_t is 0 if the feature vector is synthetic and y_t is 1 if the feature vector is natural.

The discriminator loss is propagated to the generator network and the loss used in the generator network is

$$\mathcal{L} = L_G(X^{nat}, X^{gen}) + \lambda(G, D)L_D(\hat{o}, o) \tag{8}$$

$$\lambda(G, D) = \frac{\mathbb{E}_{L_G}}{\mathbb{E}_{L_D}} \tag{9}$$

where $\mathcal{L}$ indicates the total loss, and λ indicates the adaptive coefficient, which is updated in every epoch, of the D loss. $\mathbb{E}_{L_G}$ and $\mathbb{E}_{L_D}$ are the expected values of

L_G and L_D, respectively. Therefore, the loss function tries to make the synthetic feature vectors harder to discriminate from natural ones while trying to minimize the generator's mean square error loss.

Since the adversarial training is very unstable, G or D losses can overwhelm each other. The collapse of either loss can dramatically affect the convergence of training. Therefore, an adaptive coefficient, $\lambda(G, D)$, is introduced for stabilization and normalization of the L_D.

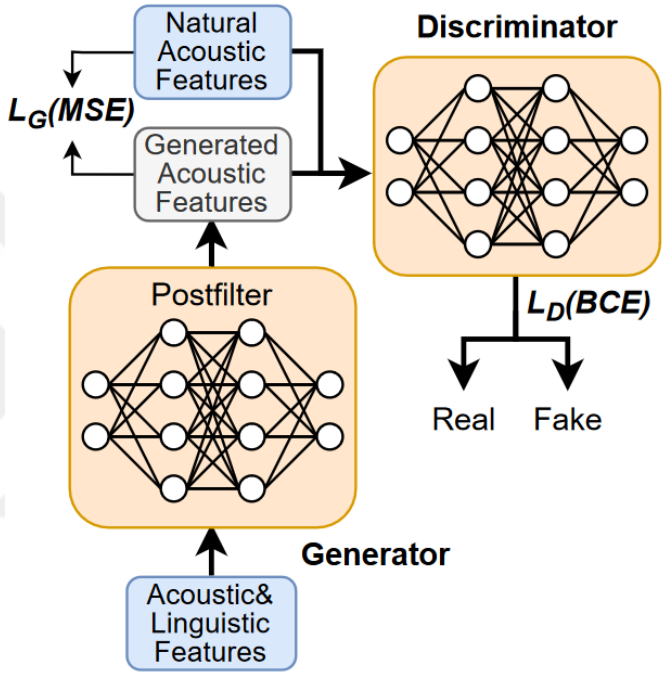


Figure 5: Postfiltering process with adversarial training. The PF model benefits from both the MSE loss and the adversarial loss.

Both synthetic and natural features are available to the discriminator at each iteration. Thus, it can use either one of the following two BCE loss functions:

$$\mathcal{L}_{D_G} = L_D(\hat{o}^{gen}, \mathbf{0}) \tag{10}$$

$$\mathcal{L}_{D_{nat}} = L_D(\hat{o}^{nat}, \mathbf{1}) \tag{11}$$

where $\mathbf{0}$ and $\mathbf{1}$ vectors denote the fake, from G, and real vectors, respectively.

The real, $\mathcal{L}_{D_{nat}}$ and fake, $\mathcal{L}_{D_G}$ mini-batch losses are subsequently backpropogated to the generator for the network weight update. Also, we observed that

adversarial training did not improve SI models. Therefore, we confined the use of adversarial trainings only to the speaker-dependent models.

5.3 Synthetic-to-synthetic feature mapping

Postfiltering of speech features with minimal data is a difficult regression problem partly because even though the synthetic features are smoothly-varying, the target natural speech features are highly variant, inconsistent, and noisy. Thus, even when synthetic segments from two different speech utterances are very similar, corresponding time-aligned speech segments can be very different as shown in Figure 6. That will force the network to learn on some average target vector if it does not rapidly get trapped on a random local optimum during training.

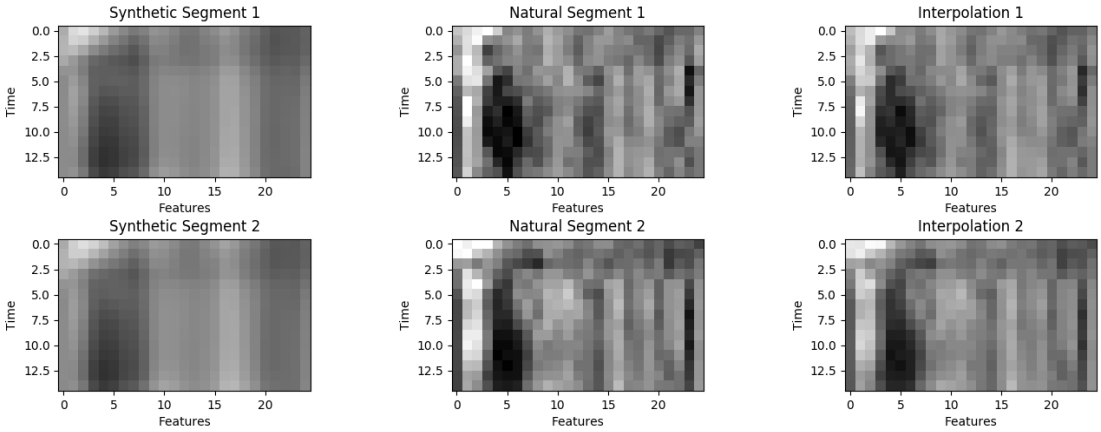


Figure 6: Spectrograms of time-aligned synthetic speech segments (first column) and natural (second column) are compared for segments that are thirteen frames long. Even though synthetic spectrograms are very similar, target natural spectrograms vary significantly. Distance between the target spectrograms are reduced after interpolation as shown in the last column. Time in y-axes are in terms of number of frames.

In addition to the noise and variations in the natural speech features, outliers can also be present in natural speech. That will further cause significant bias during learning since they will introduce high loss. Given the minimal training data used for adaptation, those outliers can have a significant impact on the network weights.

Our approach to those problems is to morph the natural features so that more consistent output vectors can be presented to the learning algorithm given similar

input vectors. In that case, the DNN system is expected to learn a more effective regression function that can both fit better to the adaption data and generalize better on the test data.

Each natural speech feature i at time t is transformed with

$$f_{i,tar}(t) = \alpha f_{i,nat}(t) + (1 - \alpha) f_{i,syn}(t). \tag{12}$$

where $f_{i,nat}(t)$ is the natural feature, $f_{i,syn}(t)$ is the synthetic feature, and $f_{i,tar}(t)$ is the target feature after interpolation. Interpolation factor α is tuned experimentally.

The effect of the interpolation algorithm is shown in Figure 6. The target spectrograms for very similar input spectrograms get closer to each other after interpolation.

CHAPTER VI

EXPERIMENTS

6.1 Experimental Setup

All experiments were implemented with the relatively noisy Wall Street Journal (WSJ) speech dataset. A total of 156 speakers were available to be used in the experiments. 135 of them were used for the training and 21 speakers were used for the testing. Adaptation was performed with 5, 10, and 15 seconds of data.

The conventional World vocoder features were used with a 16 kHz sampling rate and 5 miliseconds of frame rate. Those include MGC, BAP, LF0, and VUV features with 25, 25, 1, 1 dimensions; respectively. For MGC, BAP, and LF0 features; differential and acceleration features were also extracted. Dynamic features were not computed for the binary VUV features. A 5-state HMM model was used and state-level alignment of the data was done using the HMM/DNN-based Speech Synthesis System (HTS) tool [93].

The speaker-independent (SI) acoustic model is a RNN network with 3 FC layers and 1 LSTM layer on top along with a FC projection layer. The hidden layer dimensions are 512, 512, 512, 256, and 154, respectively. The SI model is trained with 5 hours of data from 135 male speakers where each speaker had 50 utterances each of which had a duration of 4-5 seconds. Optimizations were conducted using the Adam optimizer [94]. The number of training epochs was tuned as 50. Mini-batch size was set to 4. The experiments were performed on a single machine with one Nvidia GTX 1080 GPU, and one Intel® Core™ i7-9700 CPU @ 3.00GHz × 8 CPU.

Prior to training, the text features were normalized to the range of [0, 1] by min-max normalization, whereas the acoustic features were normalized by subtracting the mean and dividing by the standard deviation.

Both subjective and objective tests were performed to assess the speaker adaptation performance of the proposed systems. Objective tests were done using the mel-cepstral distortion (MCD) measure. Subjective listening tests were done using AB and ABX tests [1]. AB test was used to compare speech quality of two systems A and B. ABX test was used to compare the speaker similarity of system A and system B compared to the original sample X. T-test was used to assess the statistical significance of differences between the systems. 15 listeners took the tests. All listeners assessed 1 utterance from each of the 21 target speakers for each adaptation test condition.

6.2 Adaptation of the Base SI Model

Before training the postfilter, the base SI model was adapted to the target speakers. Transfer learning was used and only the final LSTM layer and the fully-connected output layer parameters were updated. Moreover, i-vectors [62] augmented with the text features were used as input to the DNN to further adapt the model to the target speakers.

6.3 LPCNet Training Setup

LPCNet vocoder was used to synthesize speech. The LPCNet architecture in [27] was used with slightly different hyperparameters than the original settings in order to convert postfiltered and enhanced acoustic features to speech waveform. GRU_A layer dimension in Figure 7 is changed to 640. Batch size was set to 64 and the number of epochs were set to 20. Adam optimizer [94] was used with a constant learning rate of 0.001.

We have found that DC shift and background noise were present in some of the speech files degraded the training performance of the LPCNet vocoder. Thus, the audio files were preprocessed as follows. Speech waveform for each utterance, $\mathbf{s}$, is amplitude-normalized by scaling it with $0.9/\max(\text{abs}(\mathbf{s}))$. To suppress the

[1] Synthesized samples can be found at https://erayee.github.io/adaptive-embedded-tts/index.html

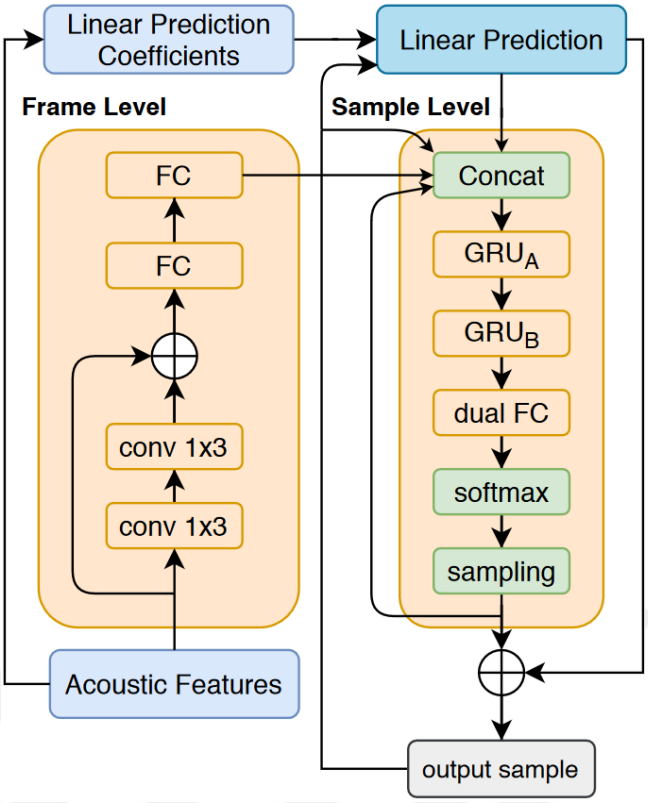


Figure 7: Architecture of the LPCNet neural vocoder. Bark-Scale cepstral coefficients and 2 pitch parameters are used as the acoustic features. "Frame Level" and "Sample Level" represent the processing level of the features. In the frame level, inputs are processed for each frame. In the sample level, inputs are processed for each sample. In the sample level, the outputs of the frame level network are repeated for the corresponding samples.

background noise, spectral subtraction was used. Silences at the start and end were removed from the speech samples.

For training the LPCNet vocoder, the preprocessed natural speech was used as the target. 18 Bark-Scale [95] cepstral coefficients, and 2 pitch parameters extracted from the natural speech were used as input. Approximately 4.5 hours of WSJ speech data was used for the training. At the inference time, enhanced acoustic features from the output of the postfilter were converted to Bark-scale cepstral coefficients and pitch parameters.

6.4 Stability of Adversarial Training

The convergence of the adversarial training algorithm relies heavily on how the generator and discriminator networks are initialized. Because limited training

data was used here to train the postfilters, stability issues of adversarial training became even more important. In fact, we have found that randomly initializing postfilters for adversarial training did not result in a good performance.

To improve the adversarial training algorithm, we followed a three-step process. In the first step, the generator was first trained with the cluster of speakers that is closest to the target speaker, i.e. a cluster-based initialization was used. Then, the parameters of the generator network were fixed and the discriminator network was trained with the same speakers. Finally, when both networks were well-initialized, in the third step, adversarial training was used to adapt both generator and discriminator using adaptation data from the target speaker.

Table 1: Mel cepstral distortion (MCD) scores of the SI postfilters with CI and without CI are shown.

Postfilter	MCD
SI-Baseline	5.19
FC	5.89
RNN	5.16
CNN	5.45
T-FNN	5.13
FC+CI	5.60
RNN+CI	5.23
CNN+CI	5.47
T-FNN+CI	**5.03**

We have found that the algorithm outlined above worked better than using the same three-step approach with only the target speaker's data. Thus, adversarial training was always used with cluster-based initialization in our experiments.

CHAPTER VII

RESULTS AND DISCUSSION

7.1 Performance of the Speaker-Independent Postfilters

Objective tests: Mel-cepstral distortion scores of all SI postfilters are shown in Table 1, and adaptation to target speakers was not applied. When cluster-based initialization (CI) was not used, T-FNN postfilter outperformed the other three postfilters. T-FNN and FC postfilters decreased the distortion while CNN and FC postfilters increased the mel-cepstral distortion. RNN and T-FNN postfilters performed closed to each other.

When there is no adaptation, the cluster-based initialization of the postfilters did not enhance the performance of the CNN postfilter considerably and slightly degraded the RNN postfilter. However, the performance of the FC and T-FNN postfilters significantly improved with the cluster-based initialization method. The effect was most clear in the FC postfilter because it has significantly fewer parameters than the other postfilters and has no recursion, which enables better adaptation performance with small amounts of data. FC-based postfilter still performed worse than the other postfilters with and without cluster-based initialzation method.

Comparison of variances was done using the scatter diagrams of the second and third coefficients as shown in Figure 8. RNN postfilter generated features that have larger variances compared to the other cases. The T-FNN postfilter could not generate as long tails as the two other postfilters when adaptation was not used.

Spectrograms for a speech sample are compared to assess the effect of postfilter. A comparison of RNN, CNN, and T-FNN postfilters are shown in Figure 9.

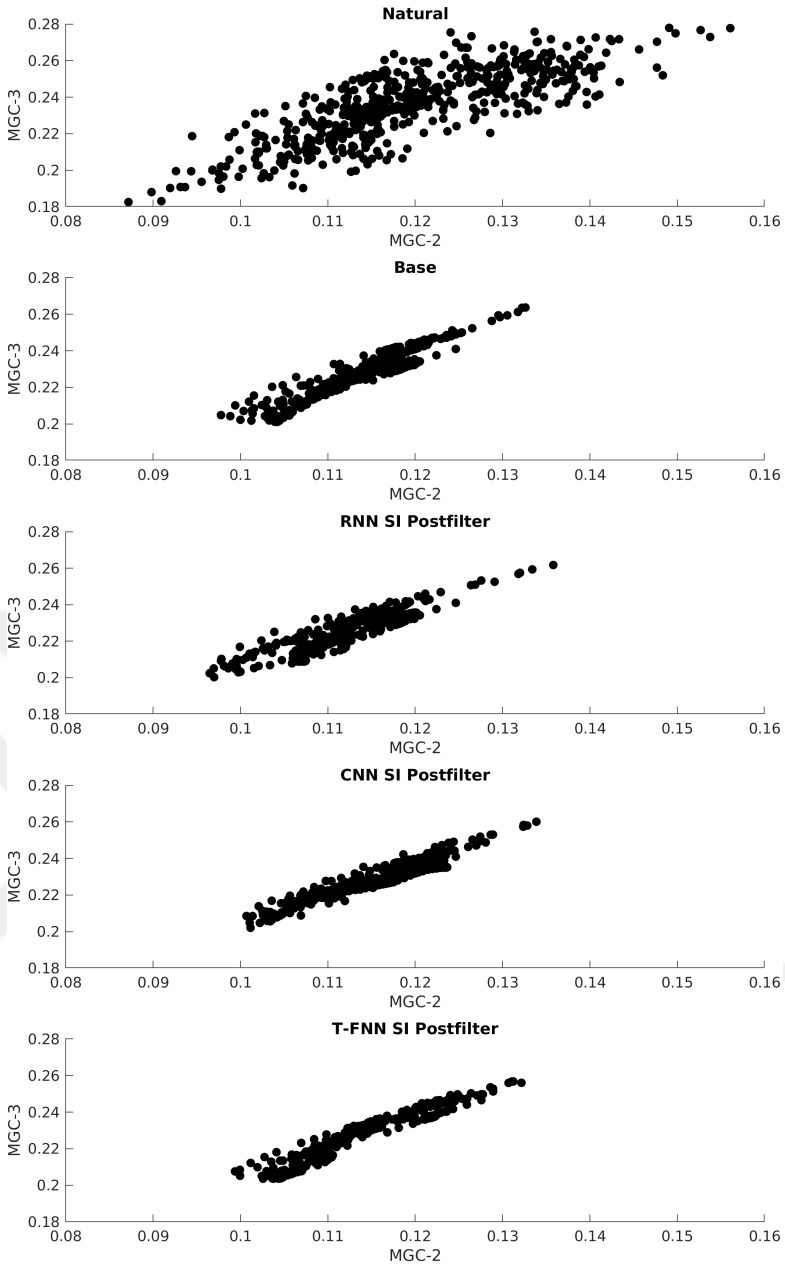


Figure 8: Scattered MGC values for the second and the third MGC coefficients of several speaker independent postfilter predictions for an utterance in the test set are compared with the natural speech and SI Baseline model prediction. Note that; since we reserve the MGC-1 for log-energy feature, mel-cepstral features start from MGC-2. The top two figures belong to the natural speech and SI Baseline model, respectively. The following three figures belong to speaker independent postfilters of RNN, CNN, and T-FNN models, respectively. From the top to the bottom, variances along MGC-2 dimension are 1.28×10^{-4}, 0.28×10^{-4}, 0.31×10^{-4}, 0.33×10^{-4}, and 0.40×10^{-4}, respectively. From the top to the bottom, variances along MGC-3 dimension are 3.43×10^{-4}, 0.95×10^{-4}, 0.70×10^{-4}, 0.77×10^{-4}, and 0.12×10^{-4}, respectively.

Formant trajectories were resolved better with the RNN and CNN postfilters compared to the T-FNN postfilter. CNN postfilter filtered a significant part of the

spectrogram and only retained the high-energy sections. T-FNN postfilter was more successful at recovering the higher frequency harmonics.

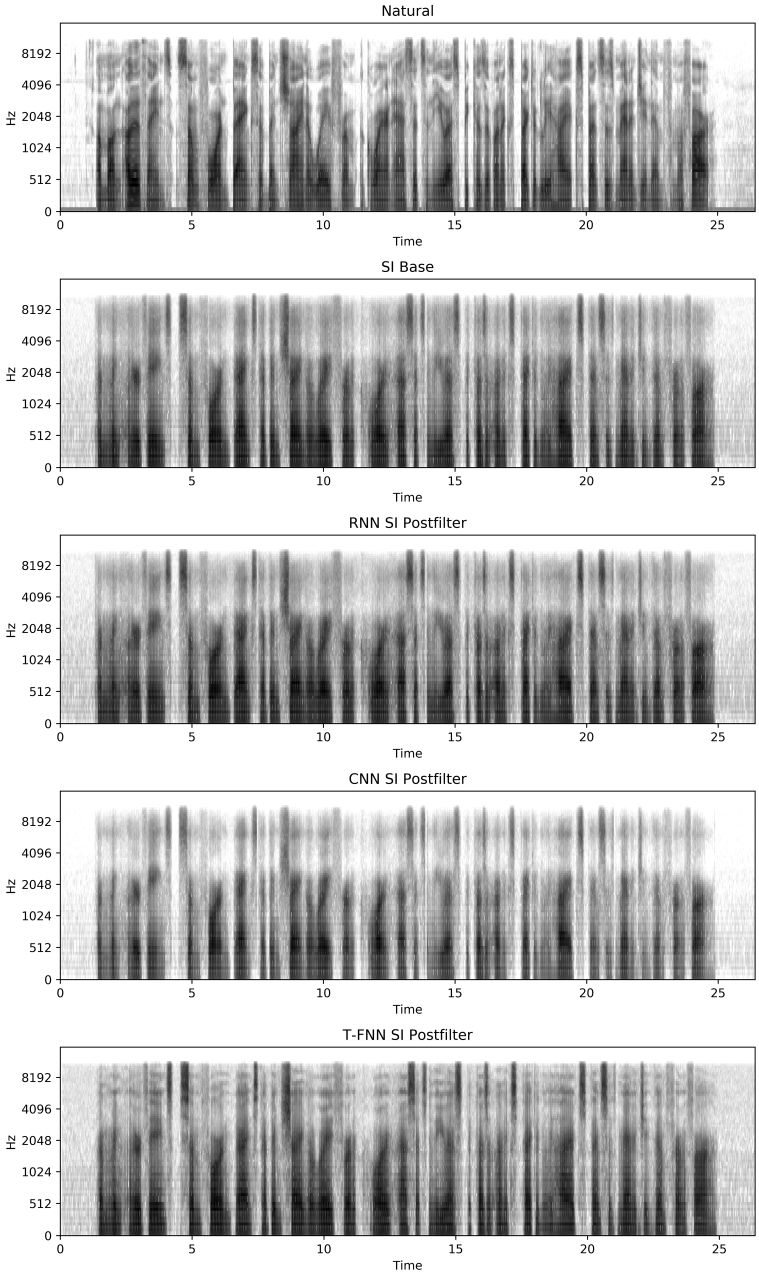


Figure 9: Mel spectrograms of several speaker independent postfilter predictions for an utterance in the test set are compared with the natural speech and SI Baseline model prediction. The top two mel spectrograms belong to the natural speech and SI Baseline model, respectively. The following three mel spectrograms belong to speaker independent postfilters of RNN, CNN, and T-FNN models, respectively.

Subjective tests: Because the base network that generates acoustic features creates overly smooth trajectories, the variance of the synthetic features is typically

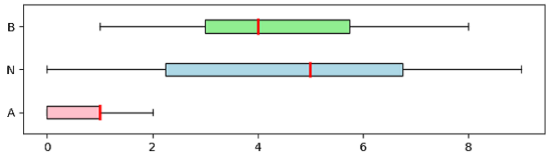
lower than the natural features. Thus, the postfilters are expected to increase the variance and get it closer to the natural feature sequences.

We performed AB and ABX tests for comparison of postfilters and results are presented below. In listening tests, improvements in formant trajectories and variance of features translated into crisper, higher quality synthesized speech with the RNN and CNN postfilters. They both improved the speech quality in the listening tests as shown in Figure 10. Moreover, even though speaker adaptation was not performed, speaker similarity was also found to improve with those two postfilters thanks to the quality improvements. Speech quality and speaker similarity obtained with the CNN and RNN postfilters were not found to be significantly different as shown in Figure 11.

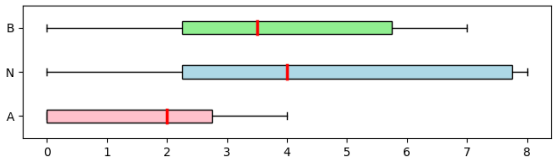
T-FNN postfilter could not improve the quality or speaker similarity compared to the baseline system as shown in Figure 12. Moreover, a comparison of the T-FNN postfilter with the best alternative RNN postfilter was done and RNN postfilter was found to perform better than the RNN postfilter as shown in Figure 12. Even though MCD scores with the T-FNN postfilter were large on average, because it could not improve formant trajectories or increase feature variances, it could not improve the speech quality.

Table 2: Using 5, 10, and 15 seconds of adaptation data, Mel cepstral distortion scores are shown for the speaker-adapted postfilters.

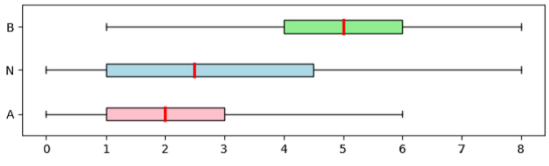
Postfilter	5 sec	10 sec	15 sec
FC	5.74	5.68	5.65
FC+CI	5.45	5.35	5.31
FC+CI+ADV	5.40	5.16	5.11
RNN	5.15	5.14	5.15
RNN+CI	5.21	5.20	5.20
RNN+CI+ADV	5.35	5.15	5.07
CNN	5.45	5.31	5.27
CNN+CI	5.45	5.29	5.25
CNN+CI+ADV	7.24	7.01	6.99
T-FNN	5.41	5.20	5.07
T-FNN+CI	5.18	**5.10**	5.05
T-FNN+CI+ADV	**5.13**	5.14	**5.01**



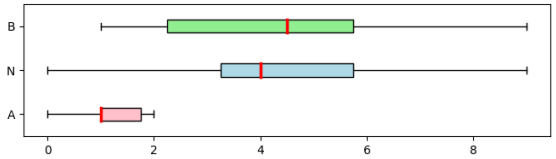
(a) B: RNN SI postfilter. p-val=0.01



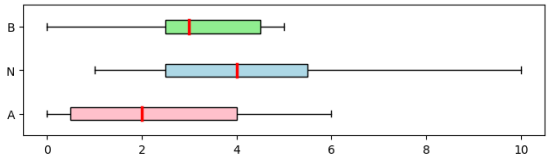
(b) B: RNN SI postfilter. p-val=0.03



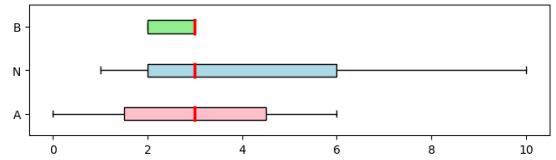
(c) B: CNN SI postfilter. p-val=0.01.



(d) B: CNN SI postfilter. p-val=0.01.

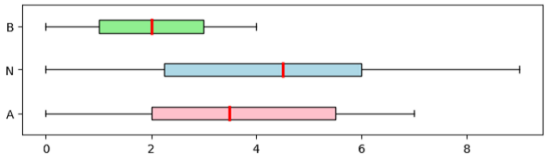


(e) B: T-FNN SI postfilter. p-val=0.5

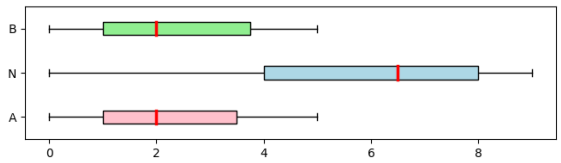


(f) B: T-FNN SI postfilter. p-val=0.8

Figure 10: A: SI-base (common for all the subfigures), N: Neutral (common notation for all AB and ABX test figures). The figures in the left column are AB tests, while the figures in the right column are ABX tests. In the AB tests, Neutral indicates that A and B are not distinguishable in terms of the sample quality. In the ABX tests, Neutral indicates that A and B are not seperable in terms of their similarity to X. x-axes indicate the number of preffered test samples (common for all AB and ABX test figures). p-val symbolizes the significance (p-values) of the corresponding tests.



(a) AB test. p-val=0.55



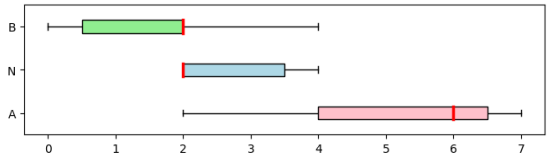
(b) ABX test. p-val=0.91

Figure 11: A: RNN SI postfilter, B: CNN SI postfilter.

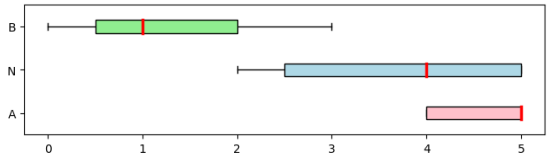
7.2 Performance of the Speaker-Adapted Postfilters

Both objective and subjective tests were used to assess the performance of the proposed algorithms. Results are presented and discussed below.

Objective tests: MCD scores of all postfilters with adaptation are shown



(a) AB test. p-val=0.002



(b) ABX test. p-val=0.006

Figure 12: A: RNN SI postfilter, B: T-FNN SI postfilter.

in Table 2. Objective scores of the FC, CNN, and T-FNN postfilters improved with increasing amounts of adaptation data for all configurations. RNN postfilter could not adapt even when 15 seconds of adaptation data was used because of the recursions in its structure. Its MCD score decreased only when adversarial training was used with 15 seconds of adaptation data.

The CNN postfilter could not adapt with 5 seconds of data, but, as opposed to the RNN postfilter, its adaptation performance improved when more data was available. Interestingly, adversarial training had a detrimental effect on the CNN postfilter as opposed to other postfilters.

Cluster-based initialization significantly improved the adaptation capability of the T-FNN and FC postfilters, slightly degraded the performance of the RNN postfilter, and did not affect the CNN postfilter. Since both RNN and RNN+CI system performances remain almost constant for all adaptation data sizes, we concluded that the RNN postfilter cannot adapt with limited data regardless of the initialization method.

Adversarial training enhanced the performance of the FC and T-FNN postfilters as shown in Table 2. However, it significantly degraded the CNN postfilter performance. Similarly, performance of the RNN-based postfilter degraded with adversarial training except for the 15sec case.

Similar to the speaker-independent case, a scatter plot analysis was performed

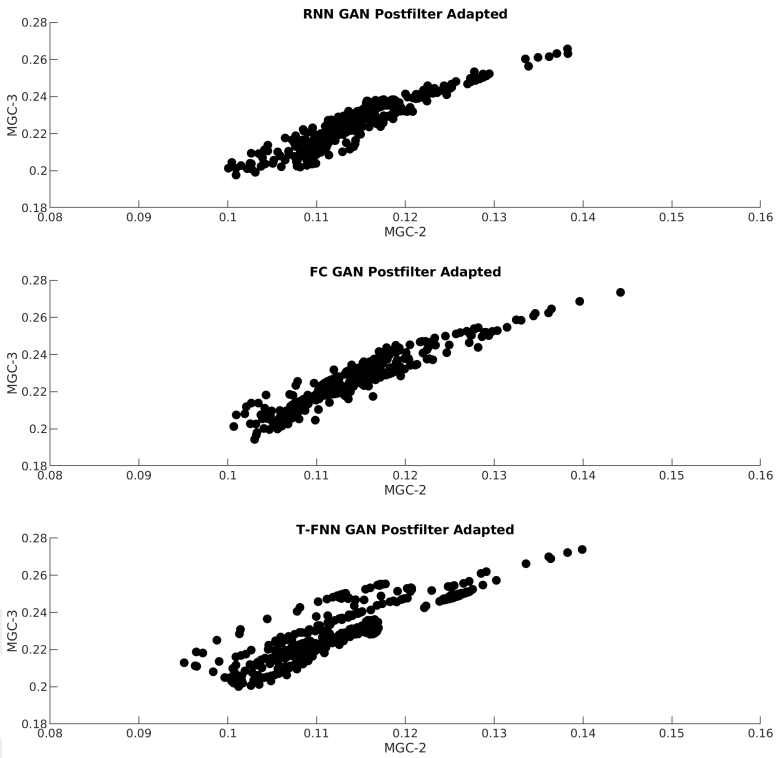


Figure 13: Scattered values of the second and the third MGC coefficients for the adapted postfilter predictions. From the top to the bottom, the figures belong to cluster-based initialized adversarial postfilters of RNN, FC, and T-FNN models, respectively. The postfilters are adapted with 3 utterances. From the top to the bottom, variances along MGC-2 dimension are 0.26×10^{-4}, 0.29×10^{-4}, and 0.69×10^{-4}, respectively. From the top to the bottom, variances along MGC-3 dimension are 1.08×10^{-4}, 1.10×10^{-4}, and 2.14×10^{-4}, respectively.

to gain further insight into the performance results. Scatter plots of the second and third MGC features were compared for the FC, RNN, and T-FNN postfilters with and without adaptation using 15 seconds of data in Figure 13. Variances of the features are higher for the FC postfilter compared to the RNN postfilter. Moreover, they increase further the T-FNN postfilters. Variance improvements with those two postfilters are partly responsible for the the speech quality improvements in the listening tests.

Mel spectrograms of the RNN, FC, and T-FNN postfilters for a single test utterance are shown in Figure 14. FC and T-FNN postfilters are particularly better at resolving the harmonics at higher frequencies. Moreover, they both have better formant resolution than the RNN postfilter. Even though the T-FNN postfilter had the best formant resolution, it did not outperform the FC postfilter

in listening tests.

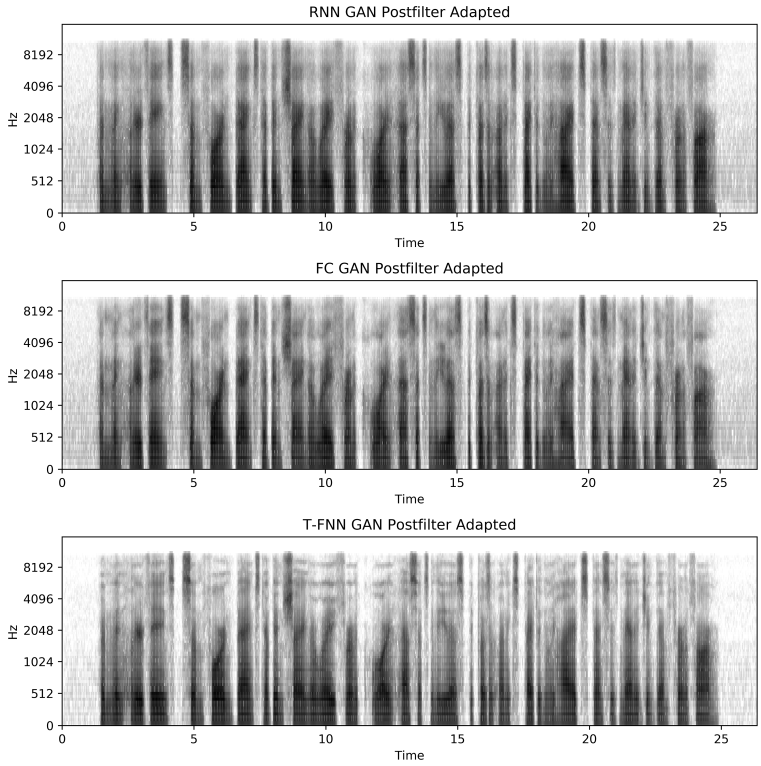
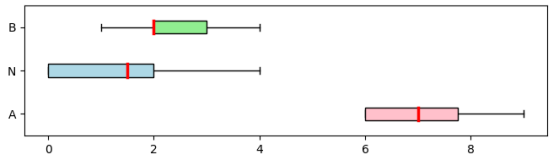


Figure 14: Mel spectrograms of several adapted postfilter predictions for an utterance in the test set. From top to bottom, the mel spectrograms are from cluster-based initialized adversarially trained postfilters of RNN, FC, and T-FNN models, respectively. The postfilters are adapted with 3 utterances.

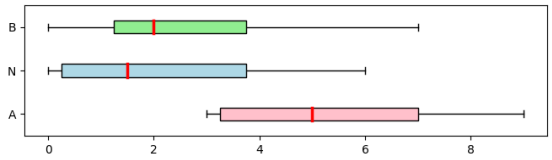
Subjective tests: In the objective test results, adaptation was not successful in the case of CNN for all methods and all adaptation data sizes. The CNN postfilter is learning to map large spectral textures and attempting to cluster-adapt it with only a few shots of data had a detrimental effect because of the large number of parameters that need to be adapted.

To confirm our findings with the objective test results, we also conducted listening tests with the CNN postfilter. The speaker-independent CNN postfilter is compared with the adapted CNN postfilter for the 15-second case, as this was when adaptation was most successful. Even for that case, adapted CNN postfilter was significantly outperformed by the SI CNN postfilter as shown in Figure 15.

In objective tests, the CI method was not effective for the case of RNN. However, both CI and adversarial training were effective for the FC postfilter. In



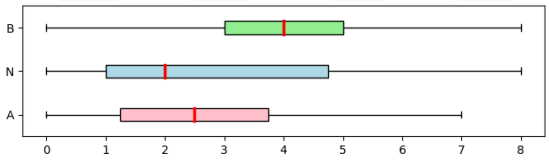
(a) AB test. p-val=0.01



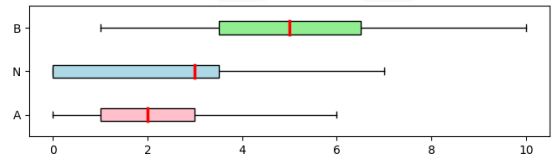
(b) ABX test. p-val=0.01

Figure 15: A: CNN SI postfilter, B: Adapted CNN SI postfilter. Adapted CNN SI postfilter is adapted with 15 seconds of data.

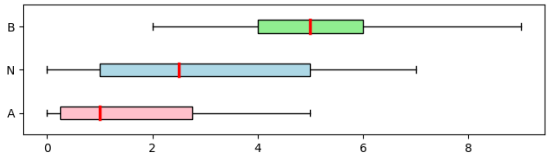
subjective tests, we first compared the performance of RNN and FC+CI+ADV postfilters. Adversarial training was used for RNN only for the 3-utterance case because that was the only case where it improved the performance. The AB quality and the ABX speaker similarity tests for FC+CI+ADV and RNN postfilters are shown in Figure 16. FC+CI+ADV postfilter significantly outperformed the RNN-based postfilter at all data sizes for both ABX and AB tests.



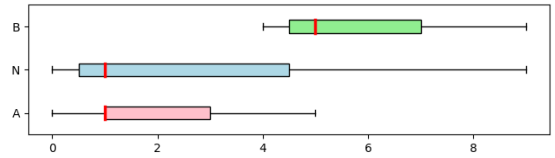
(a) AB test. 5 sec adaptation. p-val=0.09



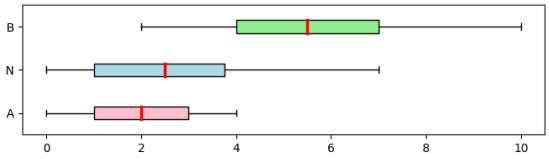
(b) ABX test. 5 sec adaptation. p-val=0.01



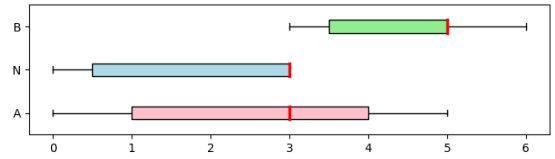
(c) AB test. 10 sec adaptation. p-val=0.01



(d) ABX test. 10 sec adaptation. p-val=0.01



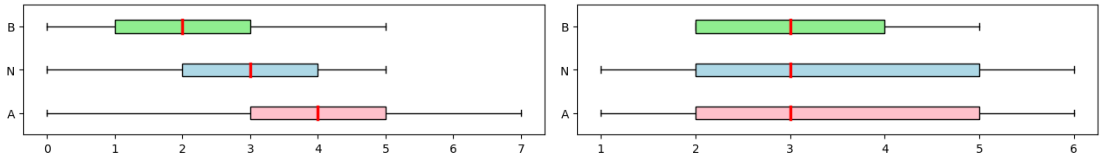
(e) AB test. 15 sec adaptation. p-val=0.01



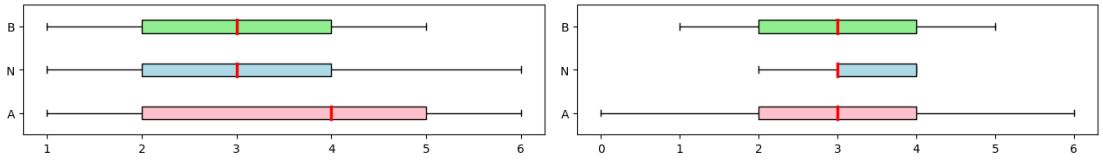
(f) ABX test. 15 sec adaptation. p-val=0.03

Figure 16: A: RNN postfilter, B: FC cluster-based GAN postfilter. Note that RNN postfilter is trained adversarially.

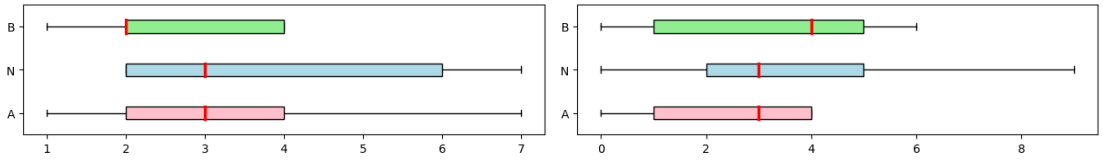
Based on the results discussed above, we concluded that in a speaker-adaptive



(a) AB test. 5 sec adaptation. p-val=0.23 (b) ABX test. 5 sec adaptation. p-val=1



(c) AB test. 10 sec adaptation. p-val=0.88 (d) ABX test. 10 sec adaptation. p-val=1



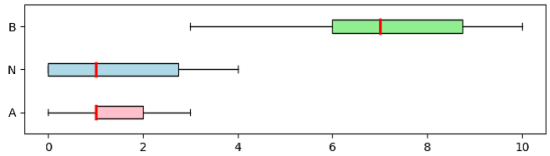
(e) AB test. 15 sec adaptation. p-val=0.2 (f) ABX test. 15 sec adaptation. p-val=0.3

Figure 17: A: T-FNN cluster-based GAN postfilter with interpolation, B: FC cluster-based GAN postfilter with interpolation.

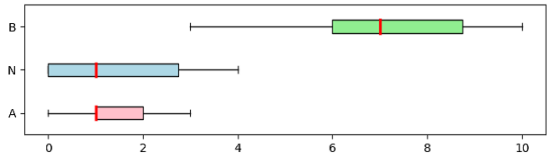
postfilter setting with a few seconds of adaptation data, the FC postfilter together with CI and adversarial training significantly outperforms the CNN and RNN postfilters. Moreover, the adaptation of CNN postfilter is not feasible with limited data.

Next, the FC postfilter was compared with the T-FNN postfilter, which performed the best in objective tests. Results are shown in Figure 17. In listening tests, there was no significant difference between the two systems in terms of speech quality or speaker similarity.

Adaptation of the SI acoustic model using transfer learning together with i-vectors as input to the network is commonly used for adaptation of DNNs using limited data. That approach is also compared with the performance of the FC postfilter and results are shown in Figure 18. Proposed FC postfilter significantly outperformed the commonly used adaptation technique for the 15-second case both at speaker similarity and speech quality. Because transfer learning was found to be ineffective for the 5sec and 10sec cases, those results are not shown in Figure 18.



(a) AB test. p-val=0.01



(b) ABX test. p-val=0.01

Figure 18: A: Transfer Learning, B: FC cluster-based GAN postfilter. 15 seconds (3-utterance) of adaptation.

7.3 Comparison of Training and Inference Speed

Training and inference speeds of the proposed filters were measured and results are shown in Table 3. The T-FNN postfilter clearly outperformed the other postfilters both in training and inference speeds. The recursive RNN postfilter was slower than the CNN and FC postfilters.

Both T-FNN and CNN postfilters batch process the input matrix while the FC and RNN postfilters process the input feature vectors one at a time. Thus, non-autoregressive architectures of T-FNN and CNN postfilters make them faster compared to the RNN and FC postfilters. However, the T-FNN postfilter was faster than the CNN postfilter because it has a significantly fewer number of parameters which affects the number of operations needed to postfilter the synthetic acoustic features as discussed in Section 4.2

Memory footprint and speed are compared in Table for the proposed mel-spectrogram generator versus the popular Tacotron 2 and Fastspeech 2 architectures in Table 4. About 14x improvement is obtained in memory size, which is critical in edge devices. Similarly, the system is 3x faster than the Fastspeech 2 algorithm.

Table 3: Comparison of training and inference speeds of the best performing speaker-adapted postfilters of each neural network type. Speed of n kHz in traning means that network can train the acoustic features corresponding to n x 1000 audio samples per second. Speed of n kHz in prediction means that postfilter network can produce enhanced MGC features corresponding to n x 1000 samples per second on a single Nvidia GTX 1080 GPU.

Postfilter	Training Speed (kHz)	Inference Speed (kHz)
T-FNN+CI+ADV	**11.5**	**2157**
FC+CI+ADV	3.11	1070
RNN+CI+ADV	2.14	723
CNN+CI	7.3	1280

Table 4: Comparison of model size and speed with the current state-of-the-art TTS systems. As a postfilter, cluster-based initialized adversarial T-FNN architecture, which has 38k trainable parameters, is used for our TTS model. Real-time-factor (RTF) is calculated for generating one second of audio.

Model	Parameter Size (M)	Inference Time (RTF)
Tacotron 2	28	0.106
FastSpeech 2	27	0.018
Proposed	**2**	**0.006**

7.4 Performance of the Interpolation Algorithm

Comparison of the first MGC feature extracted from a natural speech sample and its synthesized versions using FC postfilter with GAN, and FC postfilter with GAN and interpolation methods is shown in Figure 6. A synthetic sample was generated from the same speaker's model with 15 seconds of adaptation data.

Table 5: Mel cepstral distortion (MCD) scores of the speaker-adapted FC postfilter with and without Interpolation using 5, 10, and 15 seconds of adaptation data.

Postfilter	5 sec	10 sec	15 sec
FC+CI+ADV	5.40	5.16	5.11
FC+CI+ADV+Interpolation	5.20	5.03	4.98
T-FNN+CI+ADV	5.13	5.14	5.01
T-FNN+CI+ADV+Interpolation	**5.03**	**5.00**	**4.95**

After interpolation, the variance of the MGC feature significantly improved. Moreover, the overall synthetic feature trajectory got closer to the natural trajectory when interpolation was used. Those results are also reflected in the objective

scores presented in Table 5.

Even though the MCD scores significantly improved, speaker similarity or quality did not improve in the listening tests when the interpolation algorithm was used.

CHAPTER VIII

CONCLUSION

End-to-end systems perform well for letter-to-spectrogram mapping when large amounts of studio-quality labelled audio is available. Thus, the challenge of creating a state-of-the-art TTS system has shifted to having access to such high-quality data, which is not always easy/inexpensive to obtain especially for low-resources languages and multi-speaker datasets. Moreover, e2e systems are typically not suitable for devices with limited memory and cpu resources. Thus, investigating the more traditional DNN-based acoustic models as an alternative, less data-hungry approach is still an active area of research.

In this work, we proposed and comparatively experimented with FC-, RNN-, CNN-, and T-FNN-based postfilter architectures to enhance the output of a traditional speaker-adaptive DNN-based acoustic model. The proposed postfilters were adapted to the unseen target speakers with only a few utterances.

Even though RNN and CNN postfilters outperformed the FC postfilter when adaptation was not used, FC postfilter outperformed them when it is adapted in an adversarial manner. CNN and RNN postfilters have significantly larger number of parameters to adapt than the FC postfilter. Moreover, recursions in the RNN postfilter make adaptation more challenging. Those two factors play a major role in the superior adaptation performance of the FC postfilter. The novel T-FNN system with multi-head attention did not perform well in speaker-independent tests. However, it performed as well as the best performing FC postfilter in speaker-adapted settings. Moreover, its training and inference speeds were also significantly faster than the other postfilters.

The interpolation algorithm significantly improved the synthesized feature trajectories as measured with MCD but that did not result in audible quality or

similarity improvements.

Cluster-based initialization of the network parameters improved the performance of the FC and T-FNN postfilters. However, it did not significantly affect the RNN and CNN postfilters. Adversarial training substantially degraded the performance of the CNN postfilter at all data sizes. It also slightly degraded the performance of the RNN postfilter except with 15 seconds of adaption data where it was helpful. Still, adversarial training significantly improved the FC postfilter with increasing adaptation data sizes and the T-FNN postfilter with decreasing data sizes.

CHAPTER IX

APPENDIX

Pseudocode of the custom transformer block for the T-FNN postfilter:

```
import torch
import torch.nn as nn


class TransformerModelCustom(nn.Module):
    def __init__(self,
                 ntokens_in,
                 ntoken,
                 ninp,
                 nhead,
                 nhid,
                 nlayers,
                 dropout=0.5):
        super().__init__()
        from torch.nn import TransformerEncoder, TransformerEncoderLayer

        self.src_mask, self.pos_enc = None, PosEnc(ninp, 0.05)
        enc_layers = TransformerEncoderLayer(ninp, nhead, nhid, dropout)
        self.transformer_enc = TransformerEncoder(enc_layers, nlayers)

        self.enc = nn.Linear(ntokens_in, ninp)
        self.dec = nn.Linear(ninp, ntoken)

        self.norm = nn.LayerNorm(ninp)
```

```
        self.drop = nn.Dropout(0.05)
        self.is_mask = False
        self.ninp = ninp

        init_weights()

    def forward(self,
                src):
        src = src.transpose(0, 1)
        if self.is_mask:
            if self.src_mask is None or self.src_mask.size(0) != len(src):
                device = src.device
                mask = generate_square_subsequent_mask(len(src)).to(device)
                self.src_mask = mask

        src = self.norm(self.enc(src)) * torch.sqrt(torch.tensor(self.ninp,
                                                                 dtype=torch.float))
        src = self.pos_enc(src)
        output = self.transformer_enc(src, self.src_mask)
        output = self.dec(output)
        return output
```

Pseudocode of the discriminator:

```
def train_disc(model,
               netD,
               is_sil=False):
    netD.train()
    total_disc_real_pred = 0.0
    total_disc_fake_pred = 0.0
```

```
for ind in range(updates_per_epoch):
    train_x, train_y, non_sil_indices, _ = train_generator_gan.__next__()

    train_x = train_x.unsqueeze(dim=0).to(device)
    train_y = train_y.unsqueeze(dim=0).to(device)

    netD.zero_grad()

    b_size = len(train_y.clone().squeeze())
    label = torch.full( (b_size, 1), real_label,
                dtype=torch.float, device=device )

    # Forward pass
    output = netD(train_y.clone()[:, non_sil_indices, :].squeeze())

    # loss real
    errD_real = criterion_gan(output, label[non_sil_indices, :])

    # gradients
    errD_real.backward()
    torch.nn.utils.clip_grad_norm_(netD.parameters(), 0.8)
    D_x = output.mean().item()

    ## Train
    if is_sil:
        fake = model(train_x.clone()[:, non_sil_indices, :]).transpose(0,
                                                                1).squeeze()
    else:
        fake = model(train_x.clone()).transpose(0, 1).squeeze()
```

```
        label.fill_(fake_label)
        if is_sil:
            output = netD(fake.detach())
        else:
            output = netD(fake.detach()[non_sil_indices, :])

        # Calculate loss
        errD_fake = criterion_gan(output, label[non_sil_indices, :])

        # gradients
        errD_fake.backward()
        torch.nn.utils.clip_grad_norm_(netD.parameters(), 0.8)
        D_G_z1 = output.mean().item()

        errD = errD_real + errD_fake
        optimizerD.step()
        total_disc_real_pred += D_x
        total_disc_fake_pred += D_G_z1

        del errD_real, errD_fake

    return total_disc_real_pred, total_disc_fake_pred
```

Pseudocode of the generator:

```
def train_gen(model,
            netD,
            avg_w,
            is_sil=False):
    model.train()
```

```
netD.train()

total_loss = 0.0
total_loss_adv = 0.0
total_loss_com = 0.0
for ind in range(updates_per_epoch):
    train_x, train_y, non_sil_indices, _ = train_generator.__next__()

    train_x = train_x.unsqueeze(dim=0).to(device)
    train_y = train_y.unsqueeze(dim=0).to(device)

    netD.train()
    netD.zero_grad()

    b_size = len(train_y.clone().squeeze())
    label = torch.full( (b_size, 1), real_label,
                dtype=torch.float, device=device )

    output = netD(train_y.clone()[:, non_sil_indices, :].squeeze())

    # Calculate loss
    errD_real = criterion_gan(output, label[non_sil_indices, :])

    # gradients
    errD_real.backward()
    torch.nn.utils.clip_grad_norm_(netD.parameters(), 0.8)
    D_x = output.mean().item()

    if is_sil:
```

```
        fake = model(train_x.clone()[:, non_sil_indices, :]).transpose(0,
                                                                1).squeeze()
    else:
        fake = model(train_x.clone()).transpose(0, 1).squeeze()

    label.fill_(fake_label)

    if is_sil:
        output = netD(fake.detach())
    else:
        output = netD(fake.detach()[non_sil_indices, :]
        )

    # Calculate loss
    errD_fake = criterion_gan(output, label[non_sil_indices, :])

    # gradients
    errD_fake.backward()
    torch.nn.utils.clip_grad_norm_(netD.parameters(), 0.8)
    D_G_z1 = output.mean().item()

    errD = errD_real + errD_fake
    optimizerD.step()
    optimizer.zero_grad()
    netD.eval()

    label.fill_(real_label)
    if is_sil:
        output = netD(fake)
```

```
else:
    output = netD(fake[non_sil_indices, :])

# Calculate loss
errG = criterion_gan(output, label[non_sil_indices, :])
errG = avg_w*errG

D_G_z2 = output.mean().item()

if is_sil:
    output = model(train_x[:, non_sil_indices, :])
else:
    output = model(train_x)

if is_sil:
    loss = criterion(output.transpose(0, 1),
                train_y[:,non_sil_indices, :])
else:
    loss = criterion(output.transpose(0, 1)[:,non_sil_indices, :],
                train_y[:,non_sil_indices, :])

com_loss = loss + errG

com_loss.backward()
torch.nn.utils.clip_grad_norm_(model.parameters(), 0.8)
optimizer.step()

total_loss += loss.item()
total_loss_adv += errG.item()
```

```
        total_loss_com += com_loss.item()

        del loss, errG, errD_real, errD_fake

    return total_loss_com, total_loss, total_loss_adv
```

Bibliography

[1] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: A fully end-to-end text-to-speech synthesis model," *CoRR*, vol. abs/1703.10135, 2017.

[2] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions," in *IEEE Int. Conf., Acoust., Speech and Signal Process.*, pp. 4779–4783, 2018.

[3] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu, "Fastspeech: Fast, robust and controllable text to speech," in *Advances in Neural Inf. Process. Syst.*, pp. 3165–3174, 2019.

[4] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," *CoRR*, vol. abs/2006.04558, 2020.

[5] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. C. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," in *Proc. Int. Conf. Learn. Repr.*, 2017.

[6] S. Ö. Arik, M. Chrzanowski, A. Coates, G. F. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Y. Ng, J. Raiman, S. Sengupta, and M. Shoeybi, "Deep voice: Real-time neural text-to-speech," in *Proc. Int. Conf. Mach. Learn.*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 195–204, PMLR, 2017.

[7] A. Gibiansky, S. Ö. Arik, G. F. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Advances in Neural Inf. Process. Syst.*, pp. 2962–2970, 2017.

[8] W. Ping, K. Peng, A. Gibiansky, S. Ö. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep voice 3: Scaling text-to-speech with convolutional sequence learning," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2018.

[9] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4784–4788, 2018.

[10] R. Valle, K. J. Shih, R. Prenger, and B. Catanzaro, "Flowtron: an autoregressive flow-based generative network for text-to-speech synthesis," *CoRR*, vol. abs/2005.05957, 2020.

[11] C. Miao, S. Liang, M. Chen, J. Ma, S. Wang, and J. Xiao, "Flow-tts: A non-autoregressive network for text to speech based on flow," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 7209–7213, 2020.

[12] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," in *Advances in Neural Inf. Process. Syst.*, 2020.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Inf. Process. Syst.*, pp. 5998–6008, 2017.

[14] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proc. AAAI*, pp. 6706–6713, 2019.

[15] J. Shen, Y. Jia, M. Chrzanowski, Y. Zhang, I. Elias, H. Zen, and Y. Wu, "Non-attentive tacotron: Robust and controllable neural TTS synthesis including unsupervised duration modeling," *CoRR*, vol. abs/2010.04301, 2020.

[16] I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. J. Weiss, and Y. Wu, "Parallel tacotron: Non-autoregressive and controllable TTS," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pp. 5709–5713, IEEE, 2021.

[17] I. Elias, H. Zen, J. Shen, Y. Zhang, J. Ye, R. J. Skerry-Ryan, and Y. Wu, "Parallel tacotron 2: A non-autoregressive neural TTS model with differentiable duration modeling," *CoRR*, vol. abs/2103.14574, 2021.

[18] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli, "Pay less attention with lightweight and dynamic convolutions," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[19] M. Cuturi and M. Blondel, "Soft-dtw: a differentiable loss function for time-series," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 894–903, PMLR, 2017.

[20] Y. Taigman, L. Wolf, A. Polyak, and E. Nachmani, "Voiceloop: Voice fitting and synthesis via a phonological loop," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2018.

[21] Z. Zeng, J. Wang, N. Cheng, T. Xia, and J. Xiao, "Aligntts: Efficient feed-forward text-to-speech system without explicit alignment," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pp. 6714–6718, IEEE, 2020.

[22] C. Miao, S. Liang, Z. Liu, M. Chen, J. Ma, S. Wang, and J. Xiao, "Efficienttts: An efficient and high-quality text-to-speech architecture," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 7700–7709, PMLR, 2021.

[23] J. Vainer and O. Dusek, "Speedyspeech: Efficient neural speech synthesis," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (H. Meng, B. Xu, and T. F. Zheng, eds.), pp. 3575–3579, ISCA, 2020.

[24] D. Lim, W. Jang, G. O, H. Park, B. Kim, and J. Yoon, "JDI-T: jointly trained duration informed transformer for text-to-speech without explicit alignment," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (H. Meng, B. Xu, and T. F. Zheng, eds.), pp. 4004–4008, ISCA, 2020.

[25] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *The 9th ISCA Speech Synthesis Workshop*, p. 125, 2016.

[26] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *Proc. Int. Conf. Mach. Learn.*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 2415–2424, PMLR, 2018.

[27] J. Valin and J. Skoglund, "LPCNET: improving neural speech synthesis through linear prediction," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 5891–5895, 2019.

[28] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Inf. Process. Syst.*, pp. 14881–14892, 2019.

[29] G. Yang, S. Yang, K. Liu, P. Fang, W. Chen, and L. Xie, "Multi-band melgan: Faster waveform generation for high-quality text-to-speech," *CoRR*, vol. abs/2005.05106, 2020.

[30] M. Binkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, "High fidelity speech synthesis with adversarial networks," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2020.

[31] R. Yamamoto, E. Song, and J. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 6199–6203, 2020.

[32] J. Donahue, S. Dieleman, M. Binkowski, E. Elsen, and K. Simonyan, "End-to-end adversarial text-to-speech," *CoRR*, vol. abs/2006.03575, 2020.

[33] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Inf. Process. Syst.*, 2020.

[34] G. Fabbro, V. Golkov, T. Kemp, and D. Cremers, "Speech synthesis and control using differentiable DSP," *CoRR*, vol. abs/2010.15084, 2020.

[35] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: differentiable digital signal processing," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2020.

[36] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel wavenet: Fast high-fidelity speech synthesis," *arXiv preprint arXiv:1711.10433*, 2017.

[37] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2019.

[38] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 3617–3621, 2019.

[39] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter-based waveform model for statistical parametric speech synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 5916–5920, 2019.

[40] X. Wang and J. Yamagishi, "Using cyclic noise as the source signal for neural source-filter-based speech waveform model," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 1992–1996, ISCA, 2020.

[41] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2017.

[42] Z. Jin, A. Finkelstein, G. J. Mysore, and J. Lu, "Fftnet: A real-time speaker-dependent neural vocoder," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 2251–2255, 2018.

[43] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[44] R. J. Weiss, R. J. Skerry-Ryan, E. Battenberg, S. Mariooryad, and D. P. Kingma, "Wave-tacotron: Spectrogram-free end-to-end text-to-speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pp. 5679–5683, IEEE, 2021.

[45] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, N. Dehak, and W. Chan, "Wavegrad 2: Iterative refinement for text-to-speech synthesis," *CoRR*, vol. abs/2106.09660, 2021.

[46] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 5530–5540, PMLR, 2021.

[47] A. Katakkar and A. W. Black, "Towards language modelling in the speech domain using sub-word linguistic units," *CoRR*, vol. abs/2111.00610, 2021.

[48] J. Xu, X. Tan, Y. Ren, T. Qin, J. Li, S. Zhao, and T. Liu, "Lrspeech: Extremely low-resource speech synthesis and recognition," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020* (R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, eds.), pp. 2802–2812, ACM, 2020.

[49] M. K. Baskar, L. Burget, S. Watanabe, R. F. Astudillo, and J. H. Cernocký, "Eat: Enhanced ASR-TTS for self-supervised speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pp. 6753–6757, IEEE, 2021.

[50] Y. Ren, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu, "Almost unsupervised text to speech and automatic speech recognition," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 5410–5419, PMLR, 2019.

[51] A. Tjandra, S. Sakti, and S. Nakamura, "Listening while speaking: Speech chain by deep learning," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2017, Okinawa, Japan, December 16-20, 2017*, pp. 301–308, IEEE, 2017.

[52] A. Tjandra, S. Sakti, and S. Nakamura, "Machine speech chain with one-shot speaker adaptation," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018* (B. Yegnanarayana, ed.), pp. 887–891, ISCA, 2018.

[53] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 27, no. 12, pp. 2041–2053, 2019.

[54] L. Chen, Y. Deng, X. Wang, F. K. Soong, and L. He, "Speech bert embedding for improving prosody in neural TTS," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pp. 6563–6567, IEEE, 2021.

[55] A. H. Liu, T. Tu, H. Lee, and L. Lee, "Towards unsupervised speech recognition and synthesis with quantized speech representation learning," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pp. 7259–7263, IEEE, 2020.

[56] T. Tu, Y. Chen, A. H. Liu, and H. Lee, "Semi-supervised learning for multi-speaker text-to-speech synthesis using discrete speech representation," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (H. Meng, B. Xu, and T. F. Zheng, eds.), pp. 3191–3195, ISCA, 2020.

[57] H. Zhang and Y. Lin, "Unsupervised learning for sequence-to-sequence text-to-speech for low-resource languages," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (H. Meng, B. Xu, and T. F. Zheng, eds.), pp. 3161–3165, ISCA, 2020.

[58] K. Lakhotia, E. Kharitonov, W. Hsu, Y. Adi, A. Polyak, B. Bolte, T. A. Nguyen, J. Copet, A. Baevski, A. Mohamed, and E. Dupoux, "Generative spoken language modeling from raw audio," *CoRR*, vol. abs/2102.01192, 2021.

[59] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.

[60] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[61] W. Hsu, B. Bolte, Y. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *CoRR*, vol. abs/2106.07447, 2021.

[62] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 19, no. 4, pp. 788–798, 2011.

[63] Z. Wu, P. Swietojanski, C. Veaux, S. Renals, and S. King, "A study of speaker adaptation for dnn-based speech synthesis," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 879–883, ISCA, 2015.

[64] E. Variani, X. Lei, E. McDermott, I. Lopez-Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4052–4056, 2014.

[65] R. Doddipatla, N. Braunschweiler, and R. Maia, "Speaker adaptation in dnn-based speech synthesis using d-vectors," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 3404–3408, ISCA, 2017.

[66] R. Fu, J. Tao, Z. Wen, and Y. Zheng, "Phoneme dependent speaker embedding and model factorization for multi-speaker speech synthesis and adaptation," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 6930–6934, 2019.

[67] X. Zhou, X. Tian, G. Lee, R. K. Das, and H. Li, "End-to-end code-switching TTS with cross-lingual language model," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 7614–7618, 2020.

[68] H. Luong, S. Takaki, G. E. Henter, and J. Yamagishi, "Adapting and controlling dnn-based speech synthesis using input codes," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4905–4909, 2017.

[69] Y. Chen, Y. M. Assael, B. Shillingford, D. Budden, S. E. Reed, H. Zen, Q. Wang, L. C. Cobo, A. Trask, B. Laurie, Ç. Gülçehre, A. van den Oord, O. Vinyals, and N. de Freitas, "Sample efficient adaptive text-to-speech," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2019.

[70] M. Chen, X. Tan, B. Li, Y. Liu, T. Qin, S. Zhao, and T. Liu, "Adaspeech: Adaptive text to speech for custom voice," *CoRR*, vol. abs/2103.00993, 2021.

[71] T. Wang, J. Tao, R. Fu, J. Yi, Z. Wen, and C. Qiang, "Bi-level speaker supervision for one-shot speech synthesis," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 3989–3993, ISCA, 2020.

[72] Y. Fan, Y. Qian, F. K. Soong, and L. He, "Unsupervised speaker adaptation for dnn-based TTS synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 5135–5139, 2016.

[73] M. Chen, M. Chen, S. Liang, J. Ma, L. Chen, S. Wang, and J. Xiao, "Cross-lingual, multi-speaker text-to-speech synthesis using neural speaker embedding," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 2105–2109, ISCA, 2019.

[74] I. Himawan, S. Aryal, I. Ouyang, S. Kang, P. Lanchantin, and S. King, "Speaker adaptation of a multilingual acoustic model for cross-language synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 7629–7633, 2020.

[75] G. Sivaraman, P. Nagarsheth, and E. Khoury, "Speech synthesis in the wild," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 3217–3218, ISCA, 2018.

[76] Z. Kons, S. Shechtman, A. Sorin, C. Rabinovitz, and R. Hoory, "High quality, lightweight and adaptable TTS using lpcnet," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 176–180, ISCA, 2019.

[77] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. Lopez-Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Advances in Neural Inf. Process. Syst.*, pp. 4485–4495, 2018.

[78] E. Cooper, C. Lai, Y. Yasuda, F. Fang, X. Wang, N. Chen, and J. Yamagishi, "Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 6184–6188, 2020.

[79] H. B. Moss, V. Aggarwal, N. Prateek, J. González, and R. Barra-Chicote, "BOFFIN TTS: few-shot speaker adaptation by bayesian optimization," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 7639–7643, 2020.

[80] K. Inoue, S. Hara, M. Abe, T. Hayashi, R. Yamamoto, and S. Watanabe, "Semi-supervised speaker adaptation for end-to-end speech synthesis with pretrained models," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 7634–7638, 2020.

[81] L. Chen, T. Raitio, C. Valentini-Botinhao, Z. Ling, and J. Yamagishi, "A deep generative architecture for postfiltering in statistical parametric speech synthesis," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 23, no. 11, pp. 2003–2014, 2015.

[82] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[83] T. Kaneko, H. Kameoka, N. Hojo, Y. Ijima, K. Hiramatsu, and K. Kashino, "Generative adversarial network-based postfilter for statistical parametric speech synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4910–4914, 2017.

[84] T. Kaneko, S. Takaki, H. Kameoka, and J. Yamagishi, "Generative adversarial network-based postfilter for STFT spectrograms," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 3389–3393, ISCA, 2017.

[85] K. Tanaka, T. Kaneko, N. Hojo, and H. Kameoka, "Wavecyclegan: Synthetic-to-natural speech waveform conversion using cycle-consistent adversarial networks," *CoRR*, vol. abs/1809.10288, 2018.

[86] K. Tanaka, H. Kameoka, T. Kaneko, and N. Hojo, "Wavecyclegan2: Time-domain neural post-filter for speech waveform generation," *CoRR*, vol. abs/1904.02892, 2019.

[87] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE Int. Conf. Computer Vision, ICCV*, pp. 2242–2251, IEEE Computer Society, 2017.

[88] Y. Wu, P. L. Tobing, K. Yasuhara, N. Matsunaga, Y. Ohtani, and T. Toda, "A cyclical post-filtering approach to mismatch refinement of neural vocoder for text-to-speech systems," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 3540–3544, ISCA, 2020.

[89] P. Hsu and H. Lee, "Wg-wavenet: Real-time high-fidelity speech synthesis without GPU," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 210–214, ISCA, 2020.

[90] E. Cooper, C. Lai, Y. Yasuda, and J. Yamagishi, "Can speaker augmentation improve multi-speaker end-to-end tts?," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH* (H. Meng, B. Xu, and T. F. Zheng, eds.), pp. 3979–3983, ISCA, 2020.

[91] K. Tokuda, T. Kobayashi, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *Proc. ICASSP*, pp. 1315–1318, 2000.

[92] H. Yu, "Phase-space representation of speech," in *INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing, Jeju Island, Korea, October 4-8, 2004*, ISCA, 2004.

[93] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, "The hmm-based speech synthesis system (HTS) version 2.0," in *Sixth ISCA Workshop on Speech Synthesis*, pp. 294–299, ISCA, 2007.

[94] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[95] E. Zwicker, "Subdivision of the audible frequency range into critical bands (frequenzgruppen)," *J. Acoust. Soc. America*, vol. 33, no. 2, pp. 248–248, 1961.

[96] C. Jones, A. Smith, and E. Roberts, "Article title," in *Proceedings Title*, vol. II, pp. 803–806, IEEE, 2003.

[97] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Inf. Process. Syst.*, NIPS'14, (Cambridge, MA, USA), pp. 2672–2680, MIT Press, 2014.

[98] Y. Saito, S. Takamichi, and H. Saruwatari, "Statistical parametric speech synthesis incorporating generative adversarial networks," *CoRR*, vol. abs/1709.08041, 2017.

[99] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proceedings of the Acoustics, Speech, and Signal Processing, 1996. On Conference Proceedings., 1996 IEEE International Conference - Volume 01*, ICASSP '96, (Washington, DC, USA), pp. 373–376, IEEE Computer Society, 1996.

[100] A. W. Black, H. Zen, and K. Tokuda, "Statistical parametric speech synthesis," *Speech Communication,Volume 51, Issue 11, November 2009, Pages 1039–1064*, 2009.

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

[102] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 1964–1968, 2014.

[103] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4470–4474, 2015.

[104] S. Hochreiter and J. Schmidhuber, "Long short-term memory," 1995.

[105] R. Kubichek, "Mel-cepstral distance measure for objective speech quality assessment," in *Communications, Computers and Signal Processing, 1993., IEEE Pacific Rim Conference on*, vol. 1, pp. 125–128, IEEE, 1993.

[106] Y. Fan, Y. Qian, F. K. Soong, and L. He, "Multi-speaker modeling and speaker adaptation for dnn-based TTS synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4475–4479, 2015.

[107] S. Pascual and A. Bonafonte, "Multi-output rnn-lstm for multiple speaker speech synthesis and adaptation," 2016.

[108] Z. Wu and S. King, "Investigating gated recurrent neural networks for speech synthesis," *CoRR*, 2016.

[109] A. Mohammadi, 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 Trans. Audio, Speech, and Lang. Process.*, vol. 22, no. 12, pp. 2146–2157, 2014.

[110] A. Mohammadi and C. Demiroglu, "Nearest neighbor approach in speaker adaptation for hmm-based speech synthesis," in *21st Signal Processing and Communications Applications Conference, SIU 2013, Haspolat, Turkey, April 24-26, 2013*, pp. 1–4, 2013.

[111] T. K. S. O. T. Kobayashi, "A speaker adaptation technique for gaussian process regression based speech synthesis using feature space transform," 2016.

[112] L.-H. Chen, Z.-H. Ling, L.-J. Liu, and L.-R. Dai, "Voice conversion using deep neural networks with layer-wise generative training," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 22, pp. 1859–1872, Dec. 2014.

[113] L. Torrey and J. Shavlik, "Transfer learning."

[114] "Festival tool, available at: http://www.cstr.ed.ac.uk/projects/festival/,"

[115] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[116] C. Wu, P. Karanasou, and M. J. Gales, "Combining i-vector representation and structured neural networks for rapid adaptation," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 5000–5004, IEEE, 2016.

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

[118] H. Zen, A. W. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 7962–7966, 2013.

[119] S. King and V. Karaiskos, "The Blizzard Challenge 2013," in *Proc. Blizzard Challenge Workshop*, (Edinburgh, UK), September 2013.

[120] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," 1995.

[121] J. luc Gauvain and C. hui Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 2, pp. 291–298, 1994.

[122] J. Yamagishi, 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," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 17, pp. 66–83, Jan. 2009.

[123] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *Signal Processing Magazine*, 2012.

[124] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7962–7966, 2013.

[125] Z. hua Ling, L. Deng, and D. Yu, "Modeling spectral envelopes using restricted boltzmann machines for statistical parametric speech synthesis," icassp," 2013.

[126] S. Kang, X. Qian, and H. Meng, "Multi-distribution deep belief network for speech synthesis."

[127] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, "On the training aspects of deep neural network (dnn) for parametric tts synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 3829–3833, IEEE, 2014.

[128] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 3872–3876, 2014.

[129] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks.," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 1964–1968, 2014.

[130] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4460–4464, IEEE, 2015.

[131] B. Uria, I. Murray, S. Renals, C. Valentini-Botinhao, and J. Bridle, "Modelling acoustic feature dependencies with artificial neural networks: Trajectory-RNADE," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4465–4469, 2015.

[132] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 7893–7897, 2013.

[133] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors.," in *ASRU*, pp. 55–59, IEEE, 2013.

[134] P. Karanasou, Y. Wang, M. J. Gales, and P. C. Woodland, "Adaptation of deep neural network acoustic models using factorised i-vectors.," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 2180–2184, 2014.

[135] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 22, pp. 1713–1725, Dec. 2014.

[136] V. Gupta, P. Kenny, P. Ouellet, and T. Stafylakis, "I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 6334–6338, IEEE, 2014.

[137] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proc. IEEE Workshop on Spoken Language Technology*, (Lake Tahoe, USA), December 2014.

[138] B. Potard, P. Motlicek, and D. Imseng, "Preliminary work on speaker adaptation for dnn-based speech synthesis," tech. rep., Idiap, 2015.

[139] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems.," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, vol. 2011, pp. 249–252, 2011.

[140] A. Larcher, J.-F. Bonastre, B. G. B. Fauve, K.-A. Lee, C. Lévy, H. Li, J. S. D. Mason, and J.-Y. Parfait, "Alize 3.0 - open source toolkit for state-of-the-art speaker recognition.," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 2768–2772, ISCA, 2013.

[141] P. Swietojanski and S. Renals, "Differentiable pooling for unsupervised speaker adaptation," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4305–4309, 2015.

[142] O. Abdel-Hamid and H. Jiang, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition.," in *Interspeech*, pp. 1248–1252, 2013.

[143] T. Toda, A. W. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 15, no. 8, pp. 2222–2235, 2007.

[144] C. Veaux, J. Yamagishi, and S. King, "The voice bank corpus: Design, collection and data analysis of a large regional accent speech database," in *Oriental COCOSDA held jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE), 2013 International Conference*, pp. 1–4, IEEE, 2013.

[145] H. Kawahara, I. Masuda-Katsuse, and A. De 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, no. 3, pp. 187–207, 1999.

[146] V. Mnih, "Cudamat: a cuda-based matrix class for python," *Department of Computer Science, University of Toronto, Tech. Rep. UTML TR*, vol. 4, 2009.

[147] S. Pascual and A. Bonafonte, "Multi-output RNN-LSTM for multiple speaker speech synthesis with α-interpolation model,"

[148] Y. Zhao, D. Saito, and N. Minematsu, "Speaker representations for speaker adaptation in multiple speakers' BLSTM-RNN-based speech synthesis," *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, vol. 08-12-September-2016, pp. 2268–2272, 2016.

[149] R. Doddipatla, N. Braunschweiler, and R. Maia, "Speaker adaptation in dnn-based speech synthesis using d-vectors," *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 3404–3408, 2017.

[150] S. Takaki, S. Kim, and J. Yamagishi, "Speaker adaptation of various components in deep neural network based speech synthesis," in *9th ISCA Speech Synthesis Workshop*, pp. 153–159, 2016.

[151] M. Wan, G. Degottex, and M. J. Gales, "Integrated speaker-adaptive speech synthesis," in *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pp. 705–711, IEEE, 2017.

[152] H.-T. Luong, S. Takaki, G. E. Henter, and J. Yamagishi, "Adapting and controlling dnn-based speech synthesis using input codes," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4905–4909, IEEE, 2017.

[153] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[154] F. Lastname1, F. Lastname2, and F. Lastname3, "Title of your INTERSPEECH 2017 publication," in INTERSPEECH [155], pp. 100–104.

[155] *INTERSPEECH 2017 – 18th Annual Conference of the International Speech Communication Association, August 20–24, Stockholm, Sweden, Proceedings*, 2017.

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

[157] S. B. Davis and P. Mermelstein, "Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[158] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning – Data Mining, Inference, and Prediction*. New York: Springer, 2009.

[159] G. N. Meenakshi and P. K. Ghosh, "Whispered speech to neutral speech conversion using bidirectional lstms," *Proc. Interspeech 2018*, pp. 491–495, 2018.

[160] A. Tjandra, S. Sakti, and S. Nakamura, "Machine speech chain with one-shot speaker adaptation," *arXiv preprint arXiv:1803.10525*, 2018.

[161] B. Sisman, M. Zhang, and H. Li, "A voice conversion framework with tandem feature sparse representation and speaker-adapted wavenet vocoder," *Proc. Interspeech 2018*, pp. 1978–1982, 2018.

[162] L.-J. Liu, Z.-H. Ling, Y. Jiang, M. Zhou, and L.-R. Dai, "Wavenet vocoder with limited training data for voice conversion," *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 1983–1987, 2018.

[163] K. Chen, B. Chen, J. Lai, and K. Yu, "High-quality voice conversion using spectrogram-based wavenet vocoder," *Proc. Interspeech 2018*, pp. 1993–1997, 2018.

[164] H.-T. Luong and J. Yamagishi, "Multimodal speech synthesis architecture for unsupervised speaker adaptation," *arXiv preprint arXiv:1808.06288*, 2018.

[165] S. Liu, J. Zhong, L. Sun, X. Wu, X. Liu, and H. Meng, "Voice conversion across arbitrary speakers based on a single target-speaker utterance," *Proc. Interspeech 2018*, pp. 496–500, 2018.

[166] J.-c. Chou, C.-c. Yeh, H.-y. Lee, and L.-s. Lee, "Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations," *arXiv preprint arXiv:1804.02812*, 2018.

[167] C. Zhou, M. Horgan, V. Kumar, C. Vasco, and D. Darcy, "Voice conversion with conditional samplernn," *arXiv preprint arXiv:1808.08311*, 2018.

[168] O. Watts, C. Valentini-Botinhao, F. Espic, and S. King, "Exemplar-based speech waveform generation," *Proc. Interspeech 2018*, pp. 2022–2026, 2018.

[169] S. O. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, "Neural voice cloning with a few samples," *arXiv preprint arXiv:1802.06006*, 2018.

[170] T. Toda, M. Nakagiri, and K. Shikano, "Statistical voice conversion techniques for body-conducted unvoiced speech enhancement," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 20, no. 9, pp. 2505–2517, 2012.

[171] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," *arXiv preprint arXiv:1712.05884*, 2017.

[172] Y.-J. Hu, Z.-H. Ling, and L.-R. Dai, "Deep belief network-based post-filtering for statistical parametric speech synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 5510–5514, IEEE, 2016.

[173] T. Okamoto, K. Tachibana, T. Toda, Y. Shiga, and H. Kawai, "Deep neural network-based power spectrum reconstruction to improve quality of vocoded speech with limited acoustic parameters," *Acoustical Science and Technology*, vol. 39, no. 2, pp. 163–166, 2018.

[174] J. Wu, D. Huang, L. Xie, and H. Li, "Denoising recurrent neural network for deep bidirectional lstm based voice conversion," *Proc. Interspeech 2017*, pp. 3379–3383, 2017.

[175] T. Kaneko, H. Kameoka, N. Hojo, Y. Ijima, K. Hiramatsu, and K. Kashino, "Generative adversarial network-based postfilter for statistical parametric speech synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, vol. 2017, pp. 4910–4914, 2017.

[176] L.-H. Chen, T. Raitio, C. Valentini-Botinhao, Z.-H. Ling, and J. Yamagishi, "A deep generative architecture for postfiltering in statistical parametric speech synthesis," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 23, no. 11, pp. 2003–2014, 2015.

[177] L.-H. Chen, T. Raitio, C. Valentini-Botinhao, J. Yamagishi, and Z.-H. Ling, "Dnn-based stochastic postfilter for hmm-based speech synthesis.," in *INTERSPEECH*, pp. 1954–1958, 2014.

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

[179] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[181] K. Tanaka, T. Kaneko, N. Hojo, and H. Kameoka, "Wavecyclegan: Synthetic-to-natural speech waveform conversion using cycle-consistent adversarial networks," *CoRR*, vol. abs/1809.10288, 2018.

[182] K. Tanaka, H. Kameoka, T. Kaneko, and N. Hojo, "Wavecyclegan2: Time-domain neural post-filter for speech waveform generation," *CoRR*, vol. abs/1904.02892, 2019.

[183] T. Kaneko, S. Takaki, H. Kameoka, and J. Yamagishi, "Generative adversarial network-based postfilter for STFT spectrograms," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 3389–3393, ISCA, 2017.

[184] Z. Kons, S. Shechtman, A. Sorin, R. Hoory, C. Rabinovitz, and E. D. S. Morais, "Neural TTS voice conversion," in *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*, pp. 290–296, IEEE, 2018.

[185] Y. Wang, D. Stanton, Y. Zhang, R. J. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 5167–5176, PMLR, 2018.

[186] Z. Kons, S. Shechtman, A. Sorin, C. Rabinovitz, and R. Hoory, "High quality, lightweight and adaptable TTS using lpcnet," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 176–180, ISCA, 2019.

[187] Y. Chen, Y. M. Assael, B. Shillingford, D. Budden, S. E. Reed, H. Zen, Q. Wang, L. C. Cobo, A. Trask, B. Laurie, Ç. Gülçehre, A. van den Oord, O. Vinyals, and N. de Freitas, "Sample efficient adaptive text-to-speech," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2019.

[188] W. Hsu, Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen, P. Nguyen, and R. Pang, "Hierarchical generative modeling for controllable speech synthesis," in *Proc. Int. Conf. Learn. Repr.*, OpenReview.net, 2019.

[189] Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Z. Chen, R. J. Skerry-Ryan, Y. Jia, A. Rosenberg, and B. Ramabhadran, "Learning to speak fluently in a foreign language: Multilingual speech synthesis and cross-language voice cloning," in *Proc. Annu. Conf. Int. Speech Commun. Assoc., INTERSPEECH*, pp. 2080–2084, ISCA, 2019.

[190] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. Lopez-Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Advances in Neural Inf. Process. Syst.*, pp. 4485–4495, 2018.

[191] J. Yamagishi, T. Nose, H. Zen, Z. Ling, T. Toda, K. Tokuda, S. King, and S. Renals, "Robust speaker-adaptive hmm-based text-to-speech synthesis," *IEEE Trans. Speech Audio Process.*, vol. 17, no. 6, pp. 1208–1230, 2009.

[192] I. Himawan, S. Aryal, I. Ouyang, S. Kang, P. Lanchantin, and S. King, "Speaker adaptation of a multilingual acoustic model for cross-language

synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 7629–7633, 2020.

[193] Y. Fan, Y. Qian, F. K. Soong, and L. He, "Multi-speaker modeling and speaker adaptation for dnn-based TTS synthesis," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 4475–4479, 2015.

[194] J. Parker, Y. Stylianou, and R. Cipolla, "Adaptation of an expressive single speaker deep neural network speech synthesis system," in *IEEE Proc. Int. Conf., Acoust., Speech and Signal Process.*, pp. 5309–5313, 2018.

[195] S. Ö. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, "Neural voice cloning with a few samples," in *Advances in Neural Inf. Process. Syst.*, pp. 10040–10050, 2018.

[196] H. Tamaru, Y. Saito, S. Takamichi, T. Koriyama, and H. Saruwatari, "Generative moment matching network-based neural double-tracking for synthesized and natural singing voices," *IEICE Trans. Inf. Syst.*, vol. 103-D, no. 3, pp. 639–647, 2020.

[197] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pino, A. Baevski, A. Conneau, and M. Auli, "XLS-R: self-supervised cross-lingual speech representation learning at scale," *CoRR*, vol. abs/2111.09296, 2021.

[198] P. Oplustil-Gallegos, J. O'Mahony, and S. King, "Comparing acoustic and textual representations of previous linguistic context for improving text-to-speech," in *Proc. 11th ISCA Speech Synthesis Workshop (SSW 11)*, pp. 205–210, 2021.

[199] A. Tjandra, B. Sisman, M. Zhang, S. Sakti, H. Li, and S. Nakamura, "VQ-VAE unsupervised unit discovery and multi-scale code2spec inverter for zerospeech challenge 2019," in *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019* (G. Kubin and Z. Kacic, eds.), pp. 1118–1122, ISCA, 2019.

[200] X. Chang, T. Maekaku, P. Guo, J. Shi, Y. Lu, A. S. Subramanian, T. Wang, S. Yang, Y. Tsao, H. Lee, and S. Watanabe, "An exploration of self-supervised pretrained representations for end-to-end speech recognition," *CoRR*, vol. abs/2110.04590, 2021.

[201] E. Cooper, W.-C. Huang, T. Toda, and J. Yamagishi, "Generalization ability of mos prediction networks," *arXiv preprint arXiv:2110.02635*, 2021.

[202] J. Pan, T. Lei, K. Kim, K. J. Han, and S. Watanabe, "SRU++: pioneering fast recurrence with attention for speech recognition," *CoRR*, vol. abs/2110.05571, 2021.

[203] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pp. 5329–5333, IEEE, 2018.

[204] J. Shen, Y. Jia, M. Chrzanowski, Y. Zhang, I. Elias, H. Zen, and Y. Wu, "Non-attentive tacotron: Robust and controllable neural TTS synthesis including unsupervised duration modeling," *CoRR*, vol. abs/2010.04301, 2020.

[205] C. Veaux, J. Yamagishi, and K. MacDonald, "Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit," 2017.

[206] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "Mls: A large-scale multilingual dataset for speech research," *ArXiv*, vol. abs/2012.03411, 2020.