NON-PARALLEL MANY-TO-MANY VOICE CONVERSION BY KNOWLEDGE TRANSFER FROM A PRE-TRAINED TEXT-TO-SPEECH MODEL

by

XINYUAN YU

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

August 2020, Hong Kong

Authorization

XINYUAN YU

27 August 2020

NON-PARALLEL MANY-TO-MANY VOICE CONVERSION BY KNOWLEDGE TRANSFER FROM A PRE-TRAINED TEXT-TO-SPEECH MODEL

by

XINYUAN YU

This is to certify that I have examined the above M.Phil. thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

Dr. Brian MAK, Thesis Supervisor

Prof. Dit-Yan YEUNG, Head of Department

Department of Computer Science and Engineering

27 August 2020

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my supervisor, Dr. Brian Mak, for his mentoring and advice. His invaluable advice and comprehensive knowledge have benefited me tremendously. I have learned from him how to develop, evaluate, express, and defend my ideas.

I would like to thank the committee members of my thesis examination, Prof. Fangzhen Lin and Prof. Nevin Lianwen Zhang, for their insightful comments on this work.

I appreciate all the group members of the speech group to share their valuable insights on various interesting research topics.

I also appreciate all the participants of the speaker similarity and speech naturalness tests for their precious time.

Finally, I would like to express my deepest gratitude to my parents and my girlfriend for their love and support throughout my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NON-PARALLEL MANY-TO-MANY VOICE CONVERSION BY KNOWLEDGE TRANSFER FROM A PRE-TRAINED TEXT-TO-SPEECH MODEL

by

XINYUAN YU

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Voice conversion (VC) is the task of converting a source speaker's speech such that the output speech sounds like it is uttered by a different target speaker. Earlier approaches focus on finding a direct mapping function between a pair of source and target speakers, which requires pairs of utterances with the same content to be available in the training set. However, collecting pairs of utterances is often costly and time-consuming. Thus, training VC models with unconstrained speech data is more desirable; this is sometimes known as non-parallel VC. Recently, various deep learning methods like autoencoder, variational autoencoder and generative adversarial network are proposed for non-parallel VC. However, most of them cannot be easily trained and perform well at the same time. In this thesis, we present a simple but novel framework to train a non-parallel many-to-many VC model based on the encoder-decoder framework that can convert (seen or unseen) speech between any speaker pairs in a non-parallel speech corpus. We propose to transfer knowledge from the state-of-the-art multi-speaker text-to-speech (TTS) model, Mellotron, to the VC model by adopting Mellotron's decoder as the VC decoder. The model is trained on LibriTTS dataset with simple loss terms. Subjective evaluation shows that our proposed model is able to generate naturally sounding speech and out-perform the state-of-the-art non-parallel VC model, AUTO-VC.

CHAPTER 1

INTRODUCTION

Speech is the most natural and important way for human communication. Thus, people's passion for speech research has never stopped. Over the past decades, the rapid development of machine learning and deep learning techniques has brought revolutionary advances to speech research field. A speech recognition system can now achieve accuracy over 90 percent, while a state-of-the-art text-to-speech system can generate speech close to human in terms of naturalness. Despite the success, there are still many unsolved problems in the speech field worth investigating. Voice conversion is one of them.

Voice conversion (VC) is the task of converting a source speaker's speech such that the output speech sounds like it is uttered by a different target speaker while all the linguistic content of the original speech is maintained. Speech inherently carries different aspect of a speaker's information like timbre, prosody, or pitch to name a few. Ideally, all these factors should be converted from source speaker to target speaker.

Achieving voice conversion is not a trivial task as the potential application of voice conversion lies in a broad set of fields. In movie industry or game industry, voice dubbing is a common and important task. However, it is often expensive to get the voice of a famous actor or actress, and sometimes the actor or actress would need to dub in a non-native language. With voice conversion technique, one can convert a professional voice actor's recording to the voice of the famous actor or actress. In the medical fields, voice conversion can be used to help speech-impaired people. Speech-impaired people suffer from problems in their speech production organs, and their voice sound abnormal to ordinary people. In such cases, a voice conversion system can be used to improve the intelligibility of their voice by converting their voice back to their original voice or converting their voice to a default voice. In speaker verification or speaker recognition, one tends to identify a person based on his/her voice characteristic. Voice conversion can be used for spoofing test to test the reliability of a speaker verification/recognition system.

Converting a speech utterance from one speaker to another in all aspects (timbre, prosody, etc.) is hard to achieve. In this thesis, we focus on converting speaker identity of speech only as most literature do, and other factors of speech will be left unchanged. For the remainder of this thesis, voice conversion would mean voice conversion of speaker identity.

Voice conversion has been extensively studied in the literature. Generally, it can be categorized by the type of datasets used for building the system, namely parallel voice conversion and non-parallel voice conversion. Parallel data means that for each utterance spoken by the source speaker in the dataset, there would be another sentence spoken by the target speaker with the same linguistic content. In other words, a parallel corpus should consist of pairs of utterances uttered by at least two different speakers with the same linguistic content. Gaussian Mixture Models (GMMs) are widely adopted for parallel voice conversion [38, 18]. Recently, neural network based approaches [8, 9, 31] have made significant progress. The state-of-the-art parallel voice conversion approaches are based on sequence-to-sequence model [52, 51, 41].

The main drawback of parallel data setting is that collecting parallel data in practice is costly and time-consuming. As a result, building a voice conversion system for speakers with a small amount of data or adapting existing systems to new speakers are not easy tasks. On the other hand, non-parallel voice conversion refers to training a voice conversion model without any parallel utterance in the dataset. Recently, in light of the success of deep generative models in image field, auto-encoder based approaches [36, 32], variational autoencoder based approaches [13, 19, 15] and generative adversarial network based approaches [20, 22] are proposed for non-parallel voice conversion. The core idea of these methods is to learn disentangled speaker and content representation from training data so that the target speaker representation can substitute the source speaker representation to generate target speech.

Despite the advances brought by deep learning methods, voice conversion systems still exhibit deficiencies in converting speaker identity from one speaker to another especially for non-parallel VC. The most challenging question for non-parallel VC is how to disentangle speaker and content information from speech segments. Our motivation comes from the success of modern text-to-speech (TTS) models. Modern TTS models can already generate highly natural and intelligible speech of any desired speaker's voice. We observe that most recent VC and TTS models adopt similar encoder-decoder structures. Furthermore, the decoders of both VC and TTS models act as an acoustic model. Compared to traditional approaches where TTS models focus on transforming linguistic features to acoustic features and VC models focus on converting acoustic features, we argue that neural network based VC and TTS models both focus on transforming linguistic representations to acoustic features. This suggests that there is room for transfer learning between these two tasks. Recent TTS works have benefited from transfer learning in speaker adaptation [17, 2], but few works have focused on transferring knowledge from TTS to VC yet.

Based on the above motivation, we propose a simple but novel framework for training a non-parallel many-to-many VC model using a multi-speaker text-to-speech corpus. The proposed VC model consists of a speech encoder and an acoustic decoder. Similar to many existing non-parallel VC works [36, 6], we seek to train the speech encoder that generates speaker agnostic content representations. More specifically, we want the speech encoder to generate representations that are as close to the text encoder output of multi-speaker TTS models as possible. We believe that the encoder output of multi-speaker TTS models contains only linguistic information and has little knowledge about the speaker identity. To achieve this, we propose to transfer knowledge from a pre-trained multi-speaker TTS model to our VC model by directly adopting the decoder of the TTS model as our VC decoder. The speech encoder is therefore forced to learn to generate representations that can be recognized by the decoder. In addition, most existing VC works only model speaker identity while other factors of speech (pitch, prosody, emotions, etc.) are solely dataset dependent which result in little controllability of target speech generation. Inspired by the Mellotron model [43], we condition our proposed VC model on both explicit and learnt latent variables to better model different attributes of speech. We demonstrate that the proposed model can be effectively trained with only reconstruction loss.

Our contributions are as follows:

- We propose a training framework for transferring knowledge from TTS to VC model for non-parallel many-to-many VC.

- Our proposed model can convert (seen or unseen) speech between any speaker pairs in the training set.
- The efficacy of conditioning VC models on pitch input is verified.

Organization: The remainder of the thesis is organized as follows. In Chapter 2, we give a brief background of VC and review related works from traditional to state-of-the-art VC. In Chapter 3, we present our proposed method. In Chapter 4, we present our experimental evaluation and the results. We conclude the thesis in Chapter 5.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we provide a brief background review of voice conversion and text-to-speech. We also summarize some related works of both tasks from traditional to state-of-the-art methods. The purpose of this chapter is not to do a thorough literature review of voice conversion systems, but rather a concise one so that one could have enough background knowledge to understand the rest of the thesis.

2.1 Vocoder

Before the review, it's worth mentioning that, in most speech related tasks including voice conversion and text-to-speech, the systems or models may not directly produce the raw speech signal as their output. Instead, they work on more compact and informative acoustic features extracted from raw speech. The reason is that speech signals are particularly highly nonlinear and high dimensional (one second of speech audio recorded with a sampling rate of 16kHz would have a dimension of 16000). The resulted acoustic features will then be used to synthesize the actual speech signal. The model that transforms acoustic features into speech signal is called a vocoder.

Traditional vocoders include linear prediction coding (LPC) [29], harmonic plus noise [39], WORLD [30] and so on. In general, traditional vocoders heavily reply on speech processing techniques. However, typical features used by these vocoders like spectral features are lossy, and there is no perfect inverse transformation. As a result, the reconstructed speech usually sound intelligible but unnatural, and contains artifacts.

The state-of-the-art vocoder is WaveNet [44], and WaveNet-based models [35, 45]. Instead of applying speech processing techniques, WaveNet [44] views speech synthesis as a generative problem. The WaveNet model is trained to maximize the likelihood of raw speech waveform directly. It factorize the joint probability of waveform samples by the product rule as follows:

$$p(x) = \prod_{t=1}^{T} p(x_t|x_1, ..., x_{t-1}) \tag{2.1}$$

x_t represents the value of waveform sample at time step t. The likelihood 2.1 is modeled by a deep auto-regressive dilated convolution network, shown in figure 2.1.

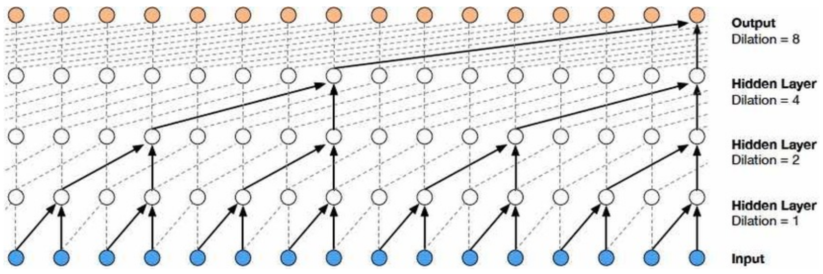


Figure 2.1: Dilated convolution layer of WaveNet

Thanks to the powerful model capacity of deep convolution neural network (CNN) and the availability of hundreds of hours of audio data, WaveNet achieves better speech quality than its traditional counterpart both in intelligibility and naturalness. In [37], it is reported that a WaveNet trained by conditioning on mel-spectrogram can produce natural sounding speech close to human recordings. Although WaveNet achieves the state-of-the-art sound quality, it fails to synthesize speech at real time. To alleviate this problem, recent works [35, 45] manage to synthesize high quality speech at real time by adopting flow-based networks.

2.2 Voice Conversion

As in figure 2.2, a generic voice conversion system consists of two stages, the training stage and conversion stage. In the training stage, acoustic features of both source and target speech would first be extracted by speech analysis tools. A mapping function between source speaker acoustic features and target speaker acoustic features will then be learnt with the training data. Over the years, it is found that voice conversion can be effectively performed by using acoustic features like spectral envelopes and fundamental frequencies. Recently, mel-spectrogram is the most popular acoustic feature. Nevertheless, the key purpose of the mapping function is to estimate the target aforementioned

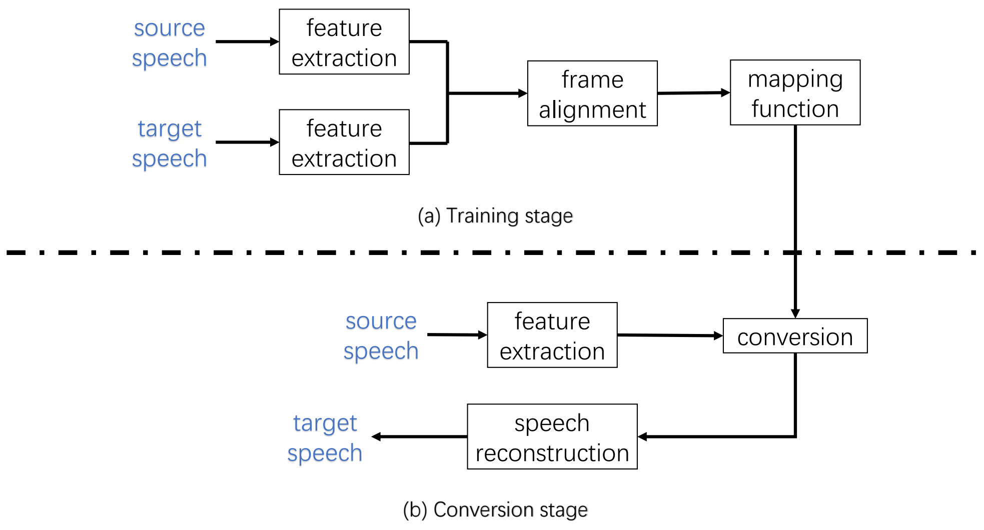


Figure 2.2: Block diagram of generic VC systems

features given source ones. Depending on the nature of the mapping function, the source and target acoustic features may need to be aligned so that the mapping function can handle different lengths of acoustic features during training. In the conversion stage, only the source acoustic features would be extracted and the learnt mapping function is used to estimate the target acoustic features. Once these features are obtained, they would be consumed by a vocoder to generate the final converted target speech.

2.2.1 Parallel VC

Traditional Voice Conversion

Traditional voice conversion focuses on transforming a source speaker's voice to a target speaker's voice where parallel data are available. The simplest way to build a voice conversion system with parallel training data is to create a code book. In [1], Abe et al. propose to apply vector-quantization to create a mapping code book of source and target speakers. During conversion, each target feature frame is calculated as a weighted sum of corresponding entries in the code book. Besides this hard clustering approach, a more common approach is to employ statistical technique to learn a function that maps source features to target features. Gaussian mixture models (GMMs) are one of the most successful functions in the literature. GMMs model acoustic features as a linear combination of a number of multivariate Gaussian distributions as the following,

$$p(\mathbf{x}) = \sum_{c=1}^{M} \alpha_c \mathcal{N}(\mathbf{x}; \mu_c, \Sigma_c) \tag{2.2}$$

where α_c is the weight of each Gaussian distribution such that $\sum_{c=1}^{M} \alpha_c = 1$, and μ_c and Σ_c are the mean and co-variance of each Gaussian distribution. All the parameters of GMMs can be learnt by the EM algorithm. Stylianou et al. [38] propose to model the distribution of source speaker's acoustic features by a GMM. In order to transform the acoustic features to that of the target speaker, a transformation function $\mathcal{F}$ is proposed as,

$$\mathcal{F}(\mathbf{x}) = \sum_{c=1}^{M} p(c|\mathbf{x})[\mathbf{v}_c + \Gamma_c \Sigma_c^{-1}(\mathbf{x} - \mu_c)] \tag{2.3}$$

where $\mathbf{v}_c$ and Γ_c are learnt parameters optimized by least squares optimization (LSO). After the training stage, the target speaker's acoustic features are calculated by 2.3.

Another way to apply GMMs is to jointly model the source and target acoustic feature distributions at the same time. In [18], Kain et al. train a GMM that fits the joint density of source and target acoustic features, which has the following form,

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix} , \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{XY} & \Sigma_{YY} \end{bmatrix} \right) \tag{2.4}$$

X and Y are the source and target acoustic feature variable respectively. During conversion, the target speaker's acoustic features can be calculated as the conditional expected value $\mathbb{E}(Y|x)$ given the GMM parameters. Compared to [38], this work models the relevant information of source and target acoustic features and requires no standalone transformation function.

Although GMM-based approaches make certain progress in voice conversion, the limited modeling capacity has become the bottleneck for these approaches. Recently, deep neural networks (DNN) have shown powerful modeling capacity and achieve state-of-the-art performance in many areas. In [8, 9, 31], DNNs are used to model the statistical relationship between source and target features, which show better performance than GMM-based approaches.

Sequence-to-sequence Based Voice Conversion

Despite the success of GMM- and DNN-based approaches, they have a common problem: the learnt mapping function operates on frame level, and considers no temporal dependencies of acoustic features. Intuitively, the production of speech signal is a highly dynamic process and performance of voice conversion is limited by a frame-by-frame mapping function. To alleviate the aforementioned problem, recent works take advantage of sequence-to-sequence (seq2seq) models [3] to directly learn a mapping function between source and target acoustic feature sequences. Sequence-to-sequence models are a family of models that are widely used in many fields including neural machine translation, speech recognition, text-to-speech and so on. Figure 2.3 shows the components of a generic seq2seq model: The encoder of a seq2seq model would first process the input sequence and generate a sequence of vector representations. These vectors are sometimes called hidden representations, and are supposed to capture the inter-dependencies of each token in the input sequence. After processing the input sequence, the decoder of a seq2seq model generates the corresponding output sequence one token at a time by "decoding" the input sequence. To inform the decoder which part of the input sequence contains most important information, an attention module is used to "summarize" the hidden representation into one context vector for each decoding time step. It should be noted that the seq2seq models can handle input and output sequences of different lengths through the attention module. Compared to traditional approaches, seq2seq models can jointly learn to align and transform input sequences to output sequences.

In [52], Zhang et al. propose to use a seq2seq based model for voice conversion. In their work, mel-spectrogram is used as the acoustic features for source and target speech, and additional bottleneck features from an ASR system are used as input to improve the pronunciation correctness of the converted speech. Compared to vanilla seq2seq models [3], a "post-net" is adopted to refine the converted mel-spectrogram. In addition, Zhang et al. also use WaveNet [44] as the vocoder. In the follow up work of Zhang et al. [51], an auxiliary classification task is added to the hidden representation of the seq2seq model to predict the phonemes of an input utterance. The target phoneme sequence is aligned to the acoustic feature sequence by force-alignment tools during data pre-processing. This auxiliary classification task is claimed to help the model reduce mispronunciations in the

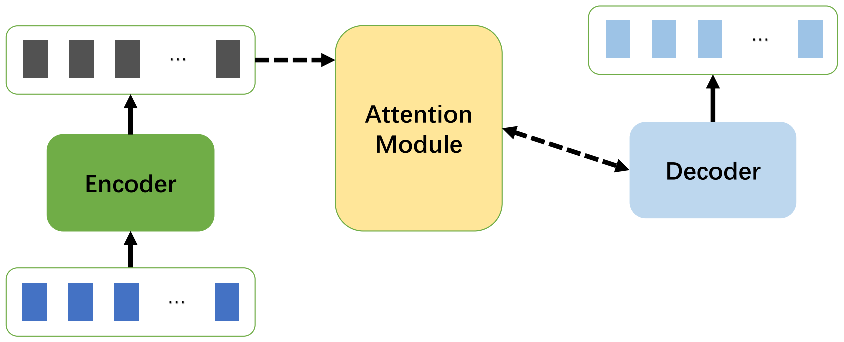


Figure 2.3: A generic sequence-to-sequence model

converted speech. Tanaka et al. [41] also try to improve the seq2seq model's alignment precision. In [41], the authors propose an auxiliary context preservation loss and guided attention loss to stabilize and accelerate the training of the attention module. The context preservation loss is the loss to reconstruct the hidden representation of the seq2seq model back to original feature sequence. Meanwhile, the guided attention loss is a diagonal mask for the alignment result of the attention module, which is the desired alignment shape for voice conversion.

Vanilla seq2seq models usually adopt LSTM or GRU as encoder and decoder, since LSTM and GRU are designed to process sequences and capture temporal relationship among sequences. However, LSTM and GRU are inherently sequential and are unfriendly to modern hardware like GPU, which limits the computational performance of the model. Recently, CNN and self-attention are shown to be effective structures in machine translation task [10, 46]. Following the trend, Kameoka et al. [21] adopt CNN architecture, and Huang et al. [14] use the transformer network [46]. Both works achieve better result than LSTM or GRU based models.

The seq2seq-based voice conversion approaches mentioned above still rely on parallel training data. However, in reality, parallel data could be hard to obtain. The following sections will review some of the voice conversion works that require no parallel data.

2.2.2 Non-parallel VC

Non-parallel voice conversion is a challenging task, since target acoustic features are no longer available. In general machine learning settings, this is an unsupervised learning task.

Auto-encoder Based Voice Conversion

In deep learning field, the most fundamental framework for unsupervised learning is auto-encoder. In the training stage, the auto-encoder usually has identical input and output. The network architecture is designed to have a bottleneck so that the network is forced to find correlations among the input and learn meaningful and succinct representations at the bottleneck layer. In voice conversion, the encoder of auto-encoder is supposed to capture linguistic information of the input speech utterance while removing speaker information so that the decoder can combine the linguistic information with any target speaker to generate the target's acoustic features. Generally, a speaker's identity is modeled by a fixed size vector called speaker embedding.

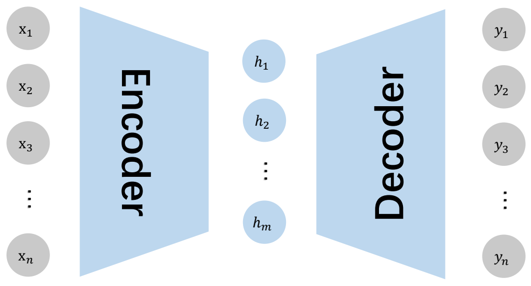


Figure 2.4: A generic auto-encoder model

Ocal et al. [32] propose a multi-path auto-encoder for non-parallel voice conversion. A universal encoder is used to "encode" source acoustic features into hidden bottleneck representations which are assumed to be speaker agnostic. Then the bottleneck representations are "decoded" by a speaker-dependent decoder to reconstruct the acoustic features. To ensure that the encoder output contains as little speaker information as possible, the authors use a classifier that takes as input the bottleneck representations and tries to classify the speaker identity. The training objective is to minimize the reconstruction error and maximize the classification error simultaneously. During conversion, the encoder "encodes" the source utterance and the specific target speaker's decoder is used to "decode" the result.

Recently, Qian et al. [36] show that the dimension of the bottleneck layer of an auto-encoder plays an important role in voice conversion. The authors manage to prove that, by carefully adjusting the bottleneck dimension, an auto-encoder can learn to disentangle linguistic information from speaker information without any auxiliary loss or network. Besides discovering the importance of finding the right bottleneck dimension, Qian et al. also introduce a pre-trained speaker encoder for modeling speaker identity and achieve state-of-the-art zero-shot voice conversion performance.

In [40], Sun et al. use an externally trained speaker-independent automatic speech recognition (ASR) system to obtain the Phonetic PosteriorGrams (PPGs) of any given speech utterance frame by frame, which are then fed to an LSTM network to predict the target speaker's acoustic features. The PPGs are assumed to be speaker-independent and contain rich phonetic information of input speech utterances, which are ideal properties for voice conversion. It should be noted that the model is trained on utterances from one single speaker. Thus, the model can only convert speech of any speaker to one target speaker. In [26], Lu et al. further integrate WaveNet [44] into this pipeline by directly conditioning WaveNet on PPG inputs. From the perspective of an encoder-decoder framework, the external ASR system can be viewed as the encoder and PPGs are the bottleneck features, while the LSTM network works as the decoder.

VAE Based Voice Conversion

Besides auto-encoder, deep generative models like Variational Auto-Encoders (VAEs) [25] are also popular choices for unsupervised learning. Unlike auto-encoder, VAEs try to maximize the log data likelihood ($\log p(x)$) by maximizing the Evidence Lower Bound (ELBO) which has the following form:

$$\log p(x) \geqslant E_{z \sim Q}[\log p(x|z)] - D[Q(z|x)||p(z)] \tag{2.5}$$

where D refers to KL-divergence between two distributions; z is a latent variable that is supposed to capture latent factors within the data samples and $p(z)$ is the prior distribution of z. In general, the posterior distribution $Q(z|x)$ and $p(x|z)$ are modeled by two neural networks and are often called "encoder" and "decoder" respectively. Although the mathematical foundation of VAE is very different from auto-encoder, they share certain spirits in common: VAE-based voice conversion approaches try to find a latent distribution z that contains little speaker information and keeps as much linguistic information as possible so that a VAE's decoder can mix linguistic information with any speaker.

Hsu et al. [13] first adopt VAE framework for converting spectral features of a given speech utterance. However, VAE itself does not guarantee the disentanglement of speaker and linguistic information. In [15], Huang et al. propose a cross-domain VAE (CD-VAE) for voice conversion. The CD-VAE has two pairs of encoder and decoder, and the authors try to disentangle speaker and linguistic information by regularizing the two encoder output to be as close as possible. Kameoka et al. [19] use an auxiliary classifier similar to [32] in order to force the network learn speaker agnostic latent code. Recently, instance normalization is proved to be effective in image style transfer [42, 23]. Inspired by [42, 23], Chou et al. [6] propose to use instance normalization at the last encoder layer to remove speaker information in input speech utterance.

GAN Based Voice Conversion

Apart from VAE, generative adversarial networks (GANs) [12] are the rising star of deep generative models in recently years. Instead of maximizing data likelihood, the generator of GANs learns to directly generate data samples by playing a minimax game with a discriminator network. In GANs-based methods, Cycle-consistent Adversarial Network (CycleGAN) [54] is a popular algorithms for image style transfer. Inspired by [54], Kaneko et al. [22] propose CycleGAN-VC for non-parallel voice conversion and achieve state-of-the-art GAN-based VC performance. However, CycleGAN can only be applied to two domains (speakers). StarGAN [5] is proposed for style transfer between multiple domains. Kameoka et al. [20] propose StarGAN-VC that follows the idea of StarGAN [5] and achieve state-of-the-art performance.

2.3 Text-to-Speech

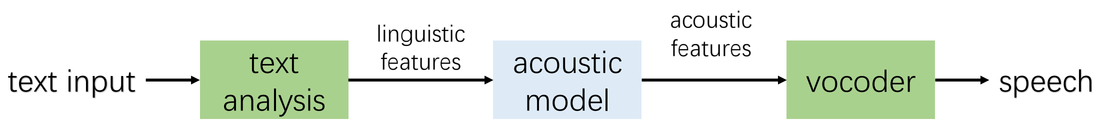


Figure 2.5: Statistical text-to-speech system

Traditional statistical text-to-speech (TTS) systems consist of three major components shown in figure 2.5. The text analysis module is a highly language-dependent module that normalizes the text and extract linguistic features from input text. The acoustic model typically is a HMM-based model that converts linguistic features to acoustic features. Finally, the vocoder synthesize the speech waveform based on the acoustic features. Traditional statistical TTS systems require a lot of domain knowledge. On the other hand, recent neural TTS works [47, 37, 34] adopt seq2seq models that directly model the relationship between text and acoustic features of a voice requiring little domain knowledge. Wang et al. [47] propose the Tacotron model that first maps English characters directly to linear spectrum of the target speech. In [37], Shen et al. propose Tacotron2 that further improves the Tacotron model by using mel-spectrogram as the output acoustic feature and WaveNet as the vocoder. Tacotron2 achieves state-of-the-art speech synthesis performance both in intelligibility and naturalness. Recently, Rafael et al. [43] introduce pitch contour as input to the Tacotron2 model which improves the controllability of the synthesis process.

Although the Tacotron2 [37] model shows promising result in text-to-speech, it can only generate sound of one single person. Arik et al. [11] first propose a multi-speaker TTS system by introducing trainable speaker embeddings to the network. The characteristic of each speaker is modeled by a fixed size vector while the major network is shared by all speakers. In [34], the authors introduce site-specific speaker embeddings and show that speaker embeddings of dimension 512 can effectively model over 2000 different speakers. Deng et al. [7] simplify the design choice by concatenating speaker embeddings only to the encoder output. Trainable speaker embeddings are effective in modeling speaker identity, however it's difficult to generalize the model to unseen speakers with trainable speaker embeddings. A potential solution to this is a speaker encoder that can map any given speech utterance to a fixed size speaker embedding. Ye et al. [17] propose to use a pre-trained speaker verification network as speaker encoder and achieve promising result in zero-shot multi-speaker TTS. In [2], Arik et al. study the problem of voice cloning which aims at generating an unseen speaker's voice with just a few speech samples. Arik et al. propose a speaker adaptation training scheme and a speaker encoder training scheme for getting a unseen speaker's vector embedding, both of which show good performance.

The model architecture and the methods of modeling speaker identity for VC and TTS have a lot in common. Some researchers have tried to train a VC model and TTS model at the same time. In [53], Zhang et al. introduce two encoders and a dual attention-based decoder to jointly learn TTS and VC models together. However, due to the design limitation, their work can only convert speech to one target speaker's voice. Luong et al. [28] propose a unified multi-speaker TTS model that contains a linguistic encoder and an acoustic encoder. Although this work is not designed for VC in the first place, it's later shown that the acoustic encoder can be used for voice conversion task [27]. Recently, Huang et al. [14] propose to transfer knowledge from a transformerTTS model to a transformer based voice conversion network. However, their work requires parallel training data.

CHAPTER 3

PROPOSED METHOD

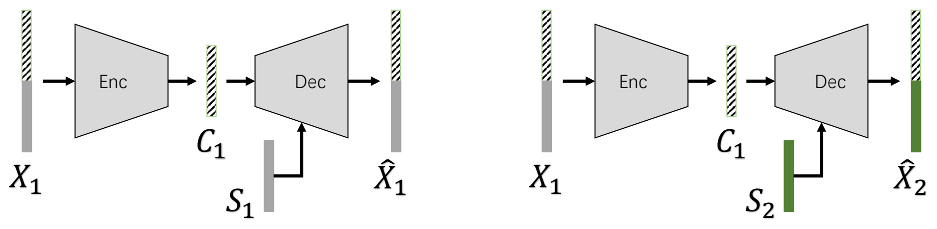
3.1 Overview

In this chapter, we present the design choice and implementation details of our proposed method. Our method consists of three major parts: (1) a TTS model based on Mellotron [43] which explicitly factorizes speech into text, pitches and global style tokens, (2) a VC model that transforms the mel-spectrograms from one speaker to another, (3) a Waveglow vocoder [35] that converts mel-spectrograms into speech audio.

In section 3.2, we first present our baseline model AUTO-VC. In section 3.3, we present the TTS model, Mellotron. In section 3.4, we present our proposed VC model and how we transfer knowledge from the TTS model to our VC model. In section 3.5, we present the Waveglow vocoder. In section 3.6, we describe the procedure of the conversion stage of the VC model.

3.2 Baseline model

We first describe the baseline model AUTO-VC [36]. In AUTO-VC, a speech segment is assumed to contain two types of information: speaker information and content information. Speaker information is produced by a speaker encoder, which can be pre-trained for a speaker verification task to verify if any given speech utterance is uttered by a specific speaker. The speaker embedding produced by the speaker encoder can be assumed to capture a speech utterance's speaker identity but contain no linguistic information. On the other hand, content information is captured by the speech encoder. By carefully choosing the output dimension of the speech encoder, the encoder would only have the capacity to encode linguistic content and "remove" speaker information. As for the decoder, it will take as input the speaker embedding of the target speaker computed by the



(a) Training stage (b) Conversion stage

Figure 3.1: Each speech segment contains two types of information, speaker information (solid) and content information (striped) (a) In training stage, when the bottleneck dimension is just right, content representations contain no speaker information. (b) In conversion stage, content representations are mixed up with target speaker embedding.

speaker encoder, and the linguistic content representation computed by the speech encoder and reconstruct the corresponding mel-spectrograms. Figure 3.1 shows an intuitive explanation of the above process.

Figure 3.2 from [36] shows the detailed architecture of AUTO-VC. Figure 3.2 (a) shows the structure of the speech encoder of AUTO-VC which consists of three "ConvNorm" layers and 2 layers of bi-directional LSTM. Each "ConvNorm" has one convolution layer followed by batch normalization and ReLU activation. The output of encoder is first downsampled and then up sampled as shown in figure 3.2 (e) and (f). In figure 3.2 (b), the speaker encoder of AUTO-VC consists of two LSTM layers followed by a fully connected layer. Figure 3.2 (c) shows the structure of the decoder of AUTO-VC. The decoder mainly consists of 3 "ConvNorm" layers and 3 LSTM layers. A convolution layer with filter width 1 is used to bring down the dimension of LSTM outputs and 5 "ConvNorm" layers learns a residual of the final output. Figure 3.2 (d) indicates the WaveNet vocoder that transforms output mel-spectrograms to speech audio.

In our implementation, we choose to simplify the model by using trainable speaker embeddings instead of a speaker encoder. This is referred as AUTO-VC-one-hot in [36] and has achieved similar performance to AUTO-VC. One may observe that the param-

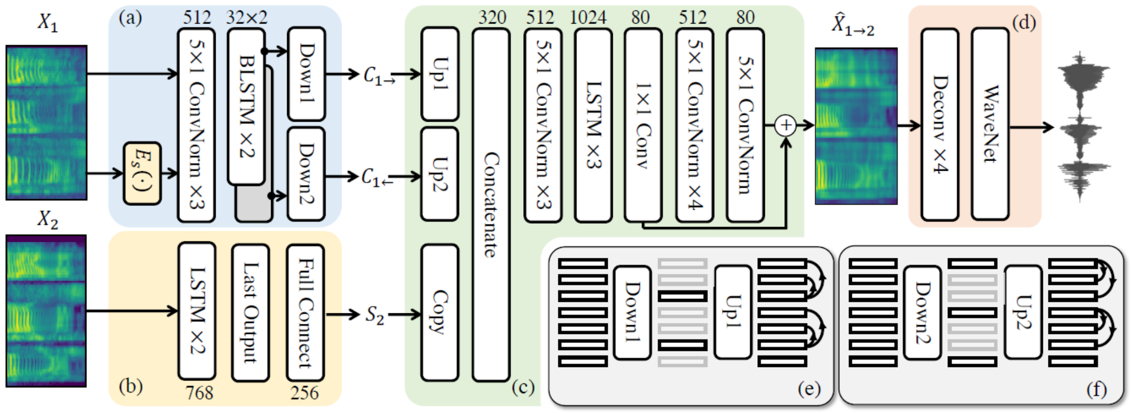


Figure 3.2: The architecture of AUTO-VC.

eters of speaker embeddings are updated only by the loss propagated from one specific speaker's utterances while the remaining model parameters are updated by the loss propagated from the whole dataset. Thus, it is safe to believe that trainable speaker embeddings can also be assumed to contain speaker information only. We use the same model architecture for other parts of the model as [36] and we do not change any hyper-parameters.

The AUTO-VC model looks simple, but the trick is smart: since all the model parameters are trained and optimized as a whole to minimize the reconstruction error, the bottleneck layer is forced to keep "the most important" information which is the linguistic content information. Otherwise, the loss term cannot be properly minimized. However, AUTO-VC has some drawbacks. First, one needs to choose the bottleneck dimension just right which requires a lot of trial-and-error. This can be bothering when training AUTO-VC on another dataset. Second, speech inherently contains information like prosody, pitches besides speaker identity and linguistic content, but AUTO-VC does not take those factors into considerations.

3.3 Text-to-Speech Model

Before we delve into our proposed method, we present the TTS model, Mellotron [43] that we adopt. The backbone of Mellotron is the Tacotron2 model [37], a sequence-to-sequence text-to-speech model.

As shown in figure 3.3, the encoder is a text analysis module which consists of three

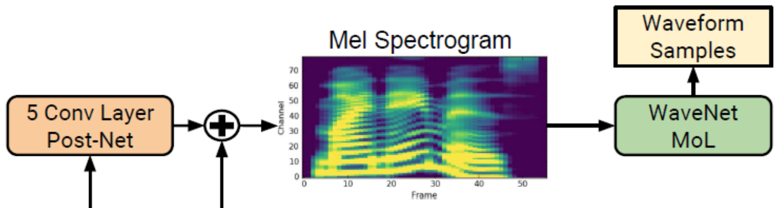


Figure 3.3: Tacotron2 model architecture

major blocks. In our implementation, the "Character Embedding" block is actually an embedding look up table for both characters and phonemes. The embedding lookup table projects one-hot vectors to high dimensional dense vector representations. We will discuss the design choice in chapter 4.2.4. Each layer of the “3 Conv Layers” block consists of a convolutional network followed by batch normalization and ReLU activation. This block is in analogy to a pronunciation module since one word usually consists of more than one phoneme and convolution layers are supposed to capture local relationships between phonemes representations. The Bidirectional LSTM then encodes the forward and backward context information of the whole sequence. The final output of the encoder and the computation procedures are summarized as follows:

$$\{\mathbf{h}_j\}_{j=1}^{T_x} = \mathrm{Enc}(\{\mathbf{x}_j\}_{j=1}^{T_x}) \tag{3.1}$$

where “Enc” represents the three blocks of the encoder, $\{\mathbf{x}_j\}_{j=1}^{T_x}$ is the input sequence of length T_x and $\{\mathbf{h}_j\}_{j=1}^{T_x}$ is the final encoder output.

The decoder is an acoustic model which turns linguistic representations into acoustic features. The “Pre-Net” layer serves the same purpose as the “Character Embedding” layer: to project one frame of mel-spectrogram to high dimensional vector representation. The "2 LSTM Layers" is an attention-based auto-regressive network that predicts one mel-spectrogram frame at a time based on the previous network output. The first LSTM layer, which is sometimes called attention RNN, generates the state $\mathbf{s}_i$ which is used to compute the attention context vector $\mathbf{c}_i$ [3] at every time step i. The second LSTM layer, which is sometimes called decoder RNN, takes as input the context vector $\mathbf{c}_i$ and generates the decoder RNN state $\mathbf{d}_i$. The final decoder output $\mathbf{y}_i$ is computed by linear projection of the decoder RNN state. This procedure is summarized as follows:

$$\mathbf{s}_i = \mathit{attentionRNN}(\mathbf{s}_{i-1}, \mathbf{c}_{i-1}, \mathbf{y}_{i-1}) \tag{3.2}$$

$$\alpha_i = \mathit{Attention}(\mathbf{s}_i, \{\mathbf{h}_j\}_{j=1}^{T_x}, \alpha_{i-1}) \tag{3.3}$$

$$\mathbf{c}_i = \sum_j \alpha_{i,j} \mathbf{h}_j \tag{3.4}$$

$$\mathbf{d}_i = \mathit{decoderRNN}(\mathbf{d}_{i-1}, \mathbf{c}_i, \mathbf{s}_i) \tag{3.5}$$

$$\mathbf{y}_i = \mathit{LinearProjection}(\mathbf{d}_i) \tag{3.6}$$

where α_i is the attention alignment weight [3] at time step i and the context vector $\mathbf{c}_i$ is the weighted sum of encoder outputs. The "Attention" in equation 3.3 refers to the calculation in the "Location Sensitive Attention" block. It is analogous to a duration model as it aligns linguistic representations to mel-spectrogram frames. The whole procedures of "Location Sensitive Attention" is as follows:

$$e_{i,j} = \mathbf{v}^T \mathit{tanh}(\mathbf{W}\mathbf{s}_i + \mathbf{V}\mathbf{h}_j + \mathcal{F}\alpha_{i-1} + \mathbf{b}) \tag{3.7}$$

$$\alpha_i = \mathit{softmax}(\mathbf{e}_i) \tag{3.8}$$

where $\mathcal{F}$ is a convolution filter, $\mathbf{W}$, $\mathbf{V}$ and $\mathbf{v}$ are weight matrices and $\mathbf{b}$ is the bias term, all of which are learnable parameters. One may notice that the previous alignment result α_{i-1} is used as features to calculate the attention alignment which gives the attention module a sense of "location". The initial alignment α_0 is set to a zero vector.

To inform an auto-regressive network to stop predicting, one usually adopts a special token called "stop token". When the "stop token" is generated, an auto-regressive network finishes its job. However, there is no proper "stop token" for mel-spectrogram frames (a frame containing all zeros may be silence of an utterance). Thus, as in figure 3.3, the decoder predicts an additional "stop token" besides predicting mel-spectrogram frames. Since the auto-regressive decoder can only utilize information of previous time steps, a "Post-Net" is introduced to incorporate past and future information to improve the precision of the final prediction.

Mellotron [43] further improves the controllability of speech synthesis by conditioning the network on speaker embeddings, pitch contour and learnt latent variables. Speaker embeddings are common approaches to model speaker identities as in [36, 34]. Pitch contour is the time profile of fundamental frequency of speech which mainly accounts for its expressiveness. Latent variables are widely used in deep learning to unravel the hidden underlying factors of complex high dimensional data. In Mellotron, the learnt latent variable is used to capture the general speech characteristics[48] that are hard to formalize. Figure 3.4 shows a block diagram of Mellotron.

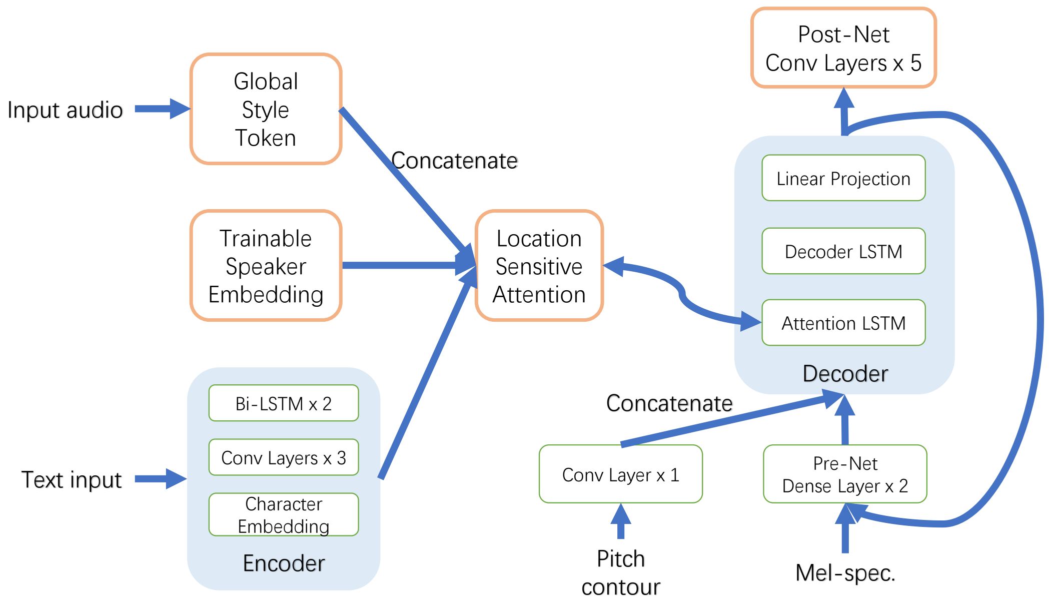


Figure 3.4: Block diagram of Mellotron. The main structure is the same as Tacotron2. Speaker embeddings and GST are concatenated with encoder outputs while pitch contour is concatenated with Pre-Net output.

The "Global Style Token" (GST) block in figure 3.4 consists of a reference encoder and a style token layer [48]. Figure 3.5 [48] presents the GST block in detail. The reference encoder is composed of six 2D convolution layers and a unidirectional GRU. It takes as input the mel-spectrogram of input audio and encodes the audio into one single vector which is then used as input to the style token layer. The style token layer consists of 10 token embeddings. To compute the final style token output, the vector from reference encoder is used as "query" and multi-head attention [46] is performed over the 10 token embeddings. The final output is called global style token because it is shown in [48] to capture general characteristics of speech utterances. As shown in figure 3.4, the global style token and speaker embeddings are channel-wise concatenated with the encoder output. The resulting sequence of vectors is fed to the attention module as a whole for later use. On the other hand, the pitch contour first goes through a convolution layer and is then channel-wise concatenated with the output of the Pre-Net.

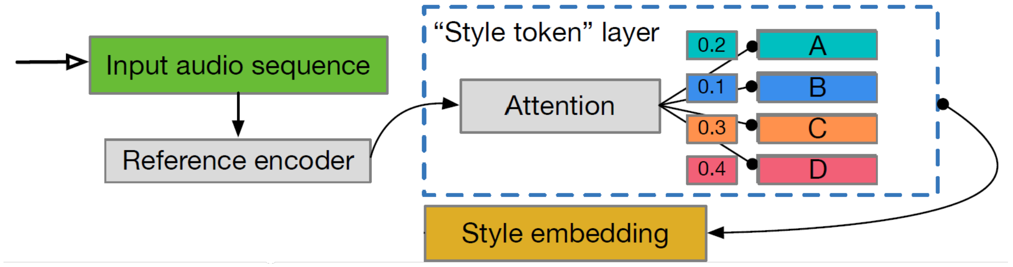


Figure 3.5: The detailed block diagram of the GST block in figure 3.4

By conditioning on these additional inputs, we have a decoder (or an acoustic model) that explicitly factorizes the generation of mel-spectrograms into linguistic representations **T**, speaker identity **s**, pitch contour **P**, alignment **R** and the global style token **z**. The conditional likelihood of mel-spectrogram is as follows:

$$P(\{\mathbf{y}_i\}_{i=1}^{T_y}) = P(\mathbf{y}_i|\mathbf{y}_{<i}, \mathbf{T}, \mathbf{s}, \mathbf{P}, \mathbf{R}, \mathbf{z}) \tag{3.9}$$

where speaker identity **s** is modeled by trainable speaker embeddings, pitch contour **P** is extracted from ground truth audio in dataset during training, alignment **R** is calculated by the attention module, linguistic representations **T** is computed as $\mathbf{T} = \mathrm{Enc}(\{\mathbf{x}_j\}_{j=1}^{T_x})$ and the global style token is computed by feeding the ground truth audio to the GST block. To train the Mellotron model, we minimize the L_2 loss between the generated mel-spectrogram $Y = \{\mathbf{y}_i\}_{i=1}^{T_y}$ and the ground truth mel-spectrogram $MEL = \{\mathbf{mel}_i\}_{i=1}^{T_{mel}}$ and the binary cross entropy loss of the "stop token":

$$\mathbb{J} = \frac{1}{N}\sum_{n=1}^{N}(\|Y^{(n)} - MEL^{(n)}\|_2 - BCE(\mathbf{x}_{stop}^{(n)}, \mathbf{y}_{stop}^{(n)})) \tag{3.10}$$

$$BCE(\mathbf{x}_{stop}^{(n)}, \mathbf{y}_{stop}^{(n)}) = \mathbf{y}_{stop}^{(n)} \cdot \log\sigma(\mathbf{x}_{stop}^{(n)}) + (1 - \mathbf{y}_{stop}^{(n)}) \cdot \log(1 - \sigma(\mathbf{x}_{stop}^{(n)})) \tag{3.11}$$

where $\mathbf{x}_{stop}^{(n)}$ is the logit output of the "stop token" module in figure 3.3 and $\mathbf{y}_{stop}^{(n)}$ is zero everywhere except for the last mel-spectrogram frame.

3.4 Proposed Framework

In this section, we present our proposed framework for training the non-parallel many-to-many VC model. The model consists of a speech encoder and an acoustic decoder.

To demonstrate the effectiveness of the framework, we use the same encoder architecture as the AUTO-VC encoder 3.2, which consists of three "ConvNorm" layers and 2 layers of bi-directional LSTM. The speech encoder takes as input the mel-spectrogram $\{\mathbf{x}_j^{mel}\}_{j=1}^{T_x}$ of speech segments and generates a sequence of hidden representations $\mathbf{h}^{VC}$,

$$\{\mathbf{h}_j^{VC}\}_{j=1}^{T_x} = Enc^{VC}(\{\mathbf{x}_j^{mel}\}_{j=1}^{T_x}) \tag{3.12}$$

The acoustic decoder is identical to the decoder of Mellotron, which takes as input linguistic representation, speaker identity, pitch contour and GST and generates mel-spectrogram frame by frame. We factorize the conditional probability of mel-spectrogram the same way as [43],

$$P(\mathbf{y}_i|\mathbf{y}_{<i}, \mathbf{H}^{VC}, \mathbf{s}, \mathbf{P}, \mathbf{R}, \mathbf{z}) \tag{3.13}$$

where $\mathbf{s}$, $\mathbf{z}$ and $\mathbf{P}$ are computed the same way as in Mellotron. The linguistic representation, however, comes from the output of speech encoder, $\mathbf{H}^{VC} = \{\mathbf{h}_j^{VC}\}_{j=1}^{T_x}$. The last piece of equation 3.13 is the alignment $\mathbf{R}$. Different from text-to-speech, the input of speech encoder and output of decoder are both mel-spectrogram, and it is safe to remove the alignment module which degenerates the model to an auto-encoder. However, keeping the alignment gives the model more flexibility. For this reason, we use the attention module of Mellotron for our VC model.

Same as AUTO-VC [36], the speech encoder is supposed to disentangle linguistic information apart from speaker information in the input mel-spectrogram. The hidden representations $\mathbf{h}^{VC}$ should contain only linguistic information and other information of speech should be purged by the encoder. Compared to Mellotron, the encoder of Mellotron serves as text analysis module and takes as input text while the speech encoder takes as input mel-spectrogram. Although the two encoders encode different input information, their goal are the same. The encoder of Mellotron also aims at capturing pronunciation related linguistic representations. Thus, we argue that the ideal output of the speech encoder should lie in the same representation space as the output of Mellotron encoder.

To achieve this, we propose a training framework that transfers knowledge from pre-trained Mellotron to our VC model. We directly use the parameters of a pre-trained Mellotron decoder for our acoustic decoder and fix the parameters during training. Since the pre-trained decoder is trained to decode representations from the Mellotron encoder, the VC model cannot generate target mel-spectrogram unless the speech encoder generates representations that lie in the same space as the output of Mellotron encoder. Besides, we adopt the speaker embeddings and global style token block of Mellotron and fix their parameters during training. In other words, we only update the parameters of speech encoder during training. In the training stage, we pass the mel-spectrograms of an arbitrary speaker and the speaker embedding of that speaker to the model, and the output of the VC model is supposed to be the reconstruction of the input mel-spectrogram. The training objective is the L_2 loss between the input mel-spectrogram $MEL = \{\mathbf{mel}_i\}_{i=1}^{T_{mel}}$ and the reconstructed mel-spectrogram $Y = \{\mathbf{y}_i\}_{i=1}^{T_y}$:

$$\mathbb{J} = \frac{1}{N}\sum_{n=1}^{N}(\|Y^{(n)} - MEL^{(n)}\|_2) \tag{3.14}$$

In our preliminary experiments, we found that regularizing the VC model training by the L_1 loss between the hidden representations of speech encoder and Mellotron encoder helps stabilize the training process. To calculate this L_1 loss, we pass the text transcript of input mel-spectrogram to the pre-trained Mellotron encoder and get its hidden representations. However, text sequence and mel-spectrogram typically have different length which makes it impractical to directly calculate the L_1 loss of the two hidden representations. To solve this problem, we expand the shape of Mellotron hidden representations with the alignment calculated by the attention module. Formally, the hidden representations of Mellotron encoder $\mathbf{H}$ have shape (T_{text}, dim) where T_{text} is the length of text sequence and dim is the vector dimension of its hidden representations. Whereas the shape of the corresponding mel-spectrogram is (T_{mel}, dim) where T_{mel} is the length of mel-spectrogram. The attention module of Mellotron aligns the aforementioned two sequences by calculating the alignment weight α for each mel-spectrogram frame as in equation 3.3. Hence, α has shape (T_{mel}, T_{text}). To expand the shape of Mellotron hidden representation, we multiply it by α:

$$\mathbf{H}_{expanded} = \alpha \cdot \mathbf{H} \tag{3.15}$$

To get α, we just pass the text sequence to Mellotron model.

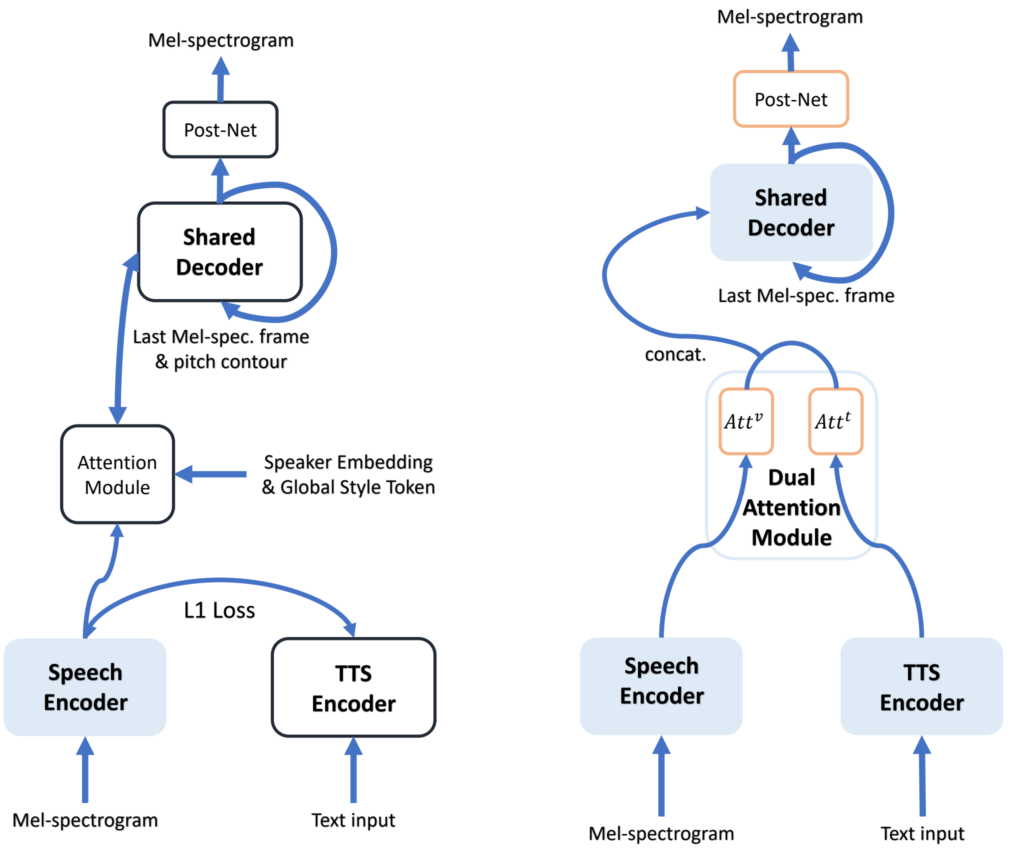
The final training objective thus becomes:

$$\mathbb{J} = \frac{1}{N}\sum_{n=1}^{N}(\|Y^{(n)} - MEL^{(n)}\|_2 + \lambda\|\mathbf{H}^{VC} - \mathbf{H}_{expanded}\|_1) \tag{3.16}$$

where λ is the weight of the regularization term.

Figure 3.6a shows a block diagram of our proposed framework where the blocks with black border indicate the parameters within those blocks are fixed during training. It is worth mentioning that, in [53], Zhang et al. propose a joint training framework for training a VC model and a TTS model simultaneously, which has similar architecture to our proposed framework. Figure 3.6b shows the general architecture of [53]. The speech encoder and TTS encoder encode mel-spectrogram and text, respectively. As shown in figure 3.6b, the dual attention module takes as input the output representation of the two encoders, and generates the context vectors for the two tasks, respectively. The decoder can digest either one of the context vectors and synthesizes the target mel-spectrogram. During training, speech input and text input are randomly selected so that the decoder can learn to decode the two different representations. At inference time, the model can be used as either a TTS model or a VC model depending on the input type. Our proposed method differs from [53]. Firstly, [53] formulates a joint training problem of TTS and VC model while we focus on transferring knowledge from a TTS model to a VC model. Secondly, the model of [53] is based on Tacotron [47] which has no pitch contour and global style token input. Lastly, [53] does not focus on non-parallel VC and can only convert speech to one target speaker's voice.

In [27], Luong et al. also propose to train a speech encoder and text encoder simultaneously. As shown in figure 3.7 [27], "LEnc" is the text encoder and "AEnc" is the speech



(a) The block diagram of our proposed framework

(b) The block diagram of joint training framework [53]

Figure 3.6: A comparison between our proposed framework and the joint training framework

encoder. The model is also trained on TTS task with VAE loss. Our proposed method differs from [27] mainly in that: (1) Our proposed model is trained with reconstruction loss which is much simpler than [27]. (2) Our proposed model takes as input pitch contour which allows fine-grained control of the conversion stage.

In [14], the authors propose a pre-training framework to transfer knowledge from TTS to VC model. Their work shares very similar idea to ours. As shown in 3.8 [14], the authors of [14] first train a standard transformerTTS model on LJSpeech [16] dataset. Second, they pre-train the speech encoder on LJSpeech dataset and directly adopt the TTS decoder. Finally, they fine-tune the speech encoder and the acoustic decoder on a parallel

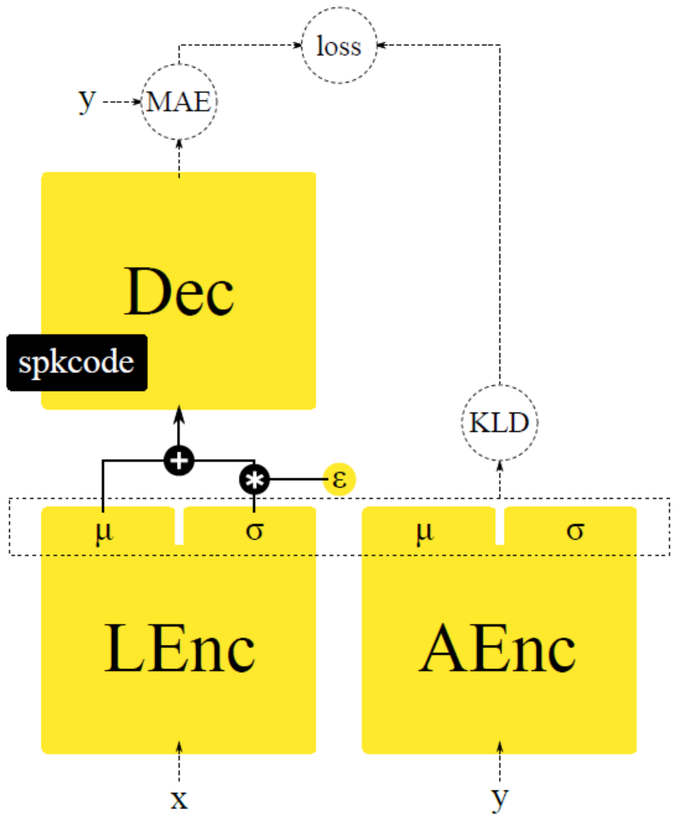


Figure 3.7: The block diagram of [27]

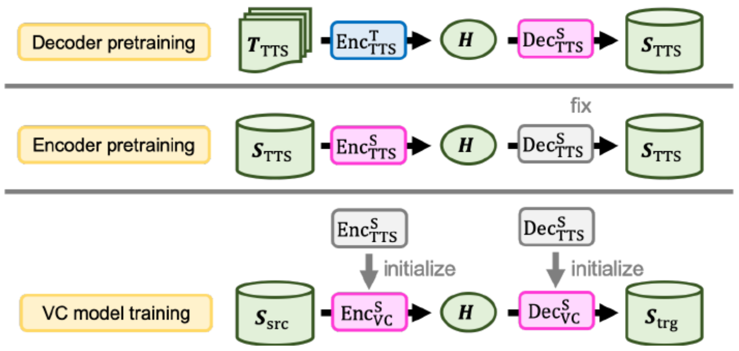


Figure 3.8: The pre-training pipeline of [14]

VC dataset. Our proposed method differs from [14] in that: (1) Our proposed method focuses on non-parallel many-to-many VC while [14] still relies on parallel VC dataset. (2) Our proposed model takes as input pitch contour which allows fine-grained control of the conversion stage.

3.5 Neural Vocoder

In this section, we describe our choice of the vocoder. In the literature, the most common choice for the vocoder is WaveNet [44]. However, due to WaveNet's auto-regressive nature, WaveNet model suffers from slow inference speed. In this thesis, we use Waveglow [35] as our neural vocoder, which is able to synthesize high fidelity speech at real time.

Waveglow is a flow-based generative model that transforms standard Gaussian distribution to target speech distribution through a series of invertible functions. The log-likelihood can be directly calculated as follows:

$$\log p_\theta(x) = \log p_\theta(z) + \sum_{i=1}^{k} \log|\det(\mathcal{J}(f_i^{-1}(x)))| \tag{3.17}$$

$$z = f_k^{-1} \circ f_{k-1}^{-1} \circ ... f_0^{-1}(x) \tag{3.18}$$

where $p_\theta(z)$ follows standard Gaussian distribution, $\det(\mathcal{J}(\cdot))$ stands for the determinant of the Jacobian matrix of a function and f_i is a invertible function. In [35], f_i is an affine transformation. The multiplicative and additive terms of f_i are computed by a WaveNet-like network that only take as input half channels of the input. This not only guarantees the invertibility of the model but also allows the model to model complex non-linear relationship between input and output distributions. In addition, since Waveglow has no auto-regressive operations, the model can fully utilize the parallelism of modern hardware like GPUs which makes it possible to synthesize speech at real time speed. To train Waveglow, we directly maximize the log data likelihood described in equation 3.17.

3.6 Conversion

After training the VC model, we are able to convert (seen or unseen) speech to any target speaker in the training dataset. However, the model cannot generalize to unseen target speakers as we use trainable speaker embeddings to model speaker identities.

To perform conversion, we use the acoustic decoder to predict the mel-spectrogram of target speaker as in equation 3.13. We pass the mel-spectrogram of source speech to the speech encoder and GST module to get $\mathbf{H}^{VC}$ and $\mathbf{z}$ in equation 3.13. To convert speaker identity, we use the target speaker's speaker embedding. It should be noted that there are two ways to compute the alignment **R** in equation 3.13: performing teacher-forcing pass with the source speech or computing the alignment at inference time. In our preliminary experiments, we find that obtaining the alignment by teacher-forcing produces better audio quality. As for the pitch contour **P**, we can extract it from the source speech. However, in our preliminary experiments, we find that directly using the pitch contour of source speech may not give good converted result. The reason is that the pitch of source speech may be out of the vocal range of the target speaker (e.g., cross gender conversion). To alleviate this, we calculate the mean pitch of both source and target speakers using all the utterances in the dataset, and we scale the extracted pitch contour by the ratio between target speaker's mean pitch and source speaker's mean pitch.

To transform predicted mel-spectrogram to speech waveform, we pass the predicted mel-spectrogram to the Waveglow vocoder.

CHAPTER 4

EXPERIMENTAL EVALUATION

In this chapter, we present the dataset we use and illustrate the experiments we conduct. We evaluate the performance of our model by case study and subjective tests.

4.1 Dataset

Two speech corpora are used for training different components in our model. Some of their basic information is shown in table 4.1.

Table 4.1: Summary of LJSpeech and LibriTTS

Dataset	Subset	Sampling Rate	Total Length (hrs)	No. of Speakers
LJSpeech		22,050Hz	24	1
	train-clean-100	24,000Hz	53.78	247
LibriTTS	train-clean-360	24,000Hz	191.29	904
	train-other-500	24,000Hz	310.08	1,160

We train our Waveglow model on LJSpeech dataset [16]. The LJSpeech dataset consists of 13,100 short audio clips of a single female speaker reading passages from 7 non-fiction books. The audio clips are recorded by a Macbook-Pro with a sampling rate of 22,050 Hz. The length of clips varies from 1 second to 10 seconds, and the total length of the dataset is 24 hours. Each audio clip is segmented based on silence, and the text transcripts are matched manually. The dataset is split into three sets: 12,500 audio clips for training, 500 audio clips for validation and 100 audio clips for testing.

For all other components of our model, we train them on a subset of LibriTTS dataset [50]. The LibriTTS dataset is designed for training neural text-to-speech models, and it is derived from the materials of the LibriSpeech dataset. The LibriTTS dataset consists of 585 hours of speech data with a sampling rate of 24kHz from 2,456 speakers and the corresponding text transcripts. The main differences between LibriTTS and LibriSpeech are (1) LibriTTS removes all the audio clips with significant background noise; (2) LibriTTS re-samples original audio material to 24kHz as original material is recorded at a sampling rate of 44.1 or 32kHz; (3) the original audio materials are segmented based on sentence break instead of silence. As shown in table 4.1, the LibriTTS dataset is also split into three subsets like LibriSpeech, namely train-clean-100, train-clean-360 and train-other-500. To conduct our experiments, we only use the train-clean-100 subset of LibriTTS. The train-clean-100 subset is further filtered according to the rules in section 4.2.1.

4.2 Pre-processing

4.2.1 Filter Long Audio clips of LibriTTS

Audio clips in LibriTTS are segmented based on sentence break. This may be useful for research like emotional TTS when one needs to consider contextual features of speech at sentence level. However, due to sentence level break, some audio clips may be too long in length to fit in the GPU memory. As a result, we either use smaller batch size or we filter out long audio clips. In this thesis, we filter out audio clips longer than 10 seconds in length to balance the amount of training data and training batch size. The final training set consists of 19,934 audio clips of 123 speakers. The validation set consists of 180 audio clips and the test set consists of 500 audio clips.

4.2.2 Unify Audio Sampling Rate

Waveglow [35], Mellotron [43] and our proposed model can only work on audio clips with consistent sampling rate. This means that the models only function properly when the audio sampling rate at inference time is the same as the sampling rate at training time. However, the sampling rate of LibriTTS and LJSpeech are different. To cope with this issue, we re-sample all LibriTTS audio clips down to 22050Hz.

4.2.3 Compute Mel-Spectrogram and Extract Fundamental Frequency

We need to compute the mel-spectrogram of input speech and extract fundamental frequency from it, since our models do not directly take as input the raw speech signals

To compute the mel-spectrogram of audio clips, we first compute the linear magnitude spectrogram of audio clips through short-time Fourier transform (STFT). The window length is 1024, the hop length is 256 and the Hanning window function is applied before STFT. The linear spectrogram is then passed to an 80 channel mel filterbank spanning 0Hz to 8000Hz. The result is a mel-spectrogram. For numerical stability, we apply a logarithmic function to the mel-spectrogram.

We use the YIN algorithm [4] to extract the fundamental frequency contour of input audio clips. We use the same window length and hop length as those used in STFT, and set the minimum and maximum frequency to 80Hz and 880Hz, respectively. The harmonic threshold for YIN algorithm is set to 0.25.

4.2.4 Use Phoneme Input

Although modern seq2seq text-to-speech models [47, 37, 34, 43] can directly transform English characters to mel-spectrogram, it is found that using phoneme as input could improve the performance of TTS model. The model that transforms text into phoneme sequence is called grapheme to phoneme model (G2P). However, the performance of TTS model is not the focus of our work, and we do not train a dedicated G2P model. For this reason, we use both phoneme and character input for Mellotron. We use the CMU Pronunciation Dictionary (CMUDict) to map English words to their phonemes whenever the words appear in the dictionary. If a word is polyphonic, we randomly select one of its pronunciations every time we try to map the word. If a word is not in the dictionary (e.g. a rare word or an abbreviation), we use character input instead. Mixing characters and phonemes as input to TTS models is also studied in [34], and is shown to be effective.

We use one of the most popular English phoneme sets, ARPABET, for our models. The ARPABET phoneme set consists of 39 phonemes as shown in table 4.2. However, in English, some phonemes have different stress in different words. For example, the phoneme 'AA' is sometimes written as 'AA1', 'AA2' or 'AA0' that stand for primary, secondary and no stress, respectively. In this thesis, to simplify the model architecture, we use different embeddings for different stress of the same phoneme (i.e. we create 3 entries in the embedding table for 'AA1', 'AA2' and 'AA0').

Table 4.2: The ARPABET phoneme set

Phoneme	**Example**	**Phoneme**	**Example**
AA	odd	L	lee
AE	at	M	me
AH	hut	N	knee
AO	ought	NG	ping
AW	cow	OW	oat
AY	hide	OY	toy
B	be	P	pee
CH	cheese	R	read
D	dee	S	sea
DH	thee	SH	she
EH	Ed	T	tea
ER	hurt	TH	theta
EY	ate	UH	hood
F	fee	UW	two
G	green	V	vee
HH	he	W	we
IH	it	Y	yield
IY	eat	Z	zee
JH	gee	ZH	seizure
K	key		

4.3 Experiments

We implement all our models with the Pytorch framework [33], and train all the models on a single NVIDIA TITAN RTX GPU. Our proposed framework involves training the following models.

Mellotron: We train the Mellotron model on the filtered train-clean-100 subset of LibriTTS. We use the ADAM [24] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-6}$. The initial learning rate is set to $1e-3$ and is exponentially decayed to $1e-5$ every 30,000 iterations. The BCE loss term in equation 3.10 has only one positive sample at the end of each sequence, which may cause problems in learning the stopping criterion. To alleviate this, we impose a positive weight (6.0) on the positive samples of the BCE loss term. In addition, we also apply L_2 regularization with a weight of 10^{-6}. The Mellotron model is trained with a batch size of 64 for 300,000 iterations.

VC model: We then train the VC model following the scheme described in chapter 3.4 on the same dataset as Mellotron. All the parameters except for the speech encoder are fixed during training. The weight λ in equation 3.16 is set to 1. We also use the ADAM [24] optimizer with the same parameters as Mellotron. The initial learning rate is set to $1e-3$, and we halve the learning rate when the learning rate is larger than $1e-5$ and the training loss starts to plateau. The VC model is trained with a batch size of 24 for around 50,000 iterations.

Waveglow: Training a Waveglow model from scratch takes an extremely large amount of time and computing resources. Due to limited available resources, we adopt a published model from NVIDIA [1], and fine-tune it on LJSpeech dataset for another 10,000 iterations using the default hyper-parameters. Although Waveglow is trained on the voice of a single female speaker, it is found that Waveglow can transform mel-spectrogram of unseen speakers, even male speakers, to speech audio. Hence, Waveglow can be regarded as a universal vocoder.

To measure the performance of our proposed framework, we train the following models and compare our proposed model with them.

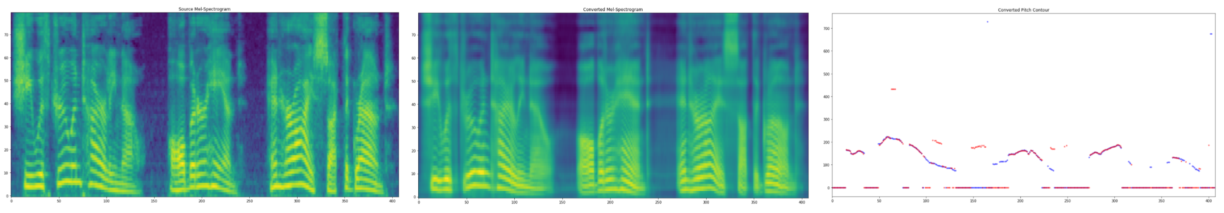
AUTO-VC: AUTO-VC [36] achieves the state-of-the-art performance for non-parallel

[1]https://github.com/NVIDIA/waveglow

voice conversion. Details of the model is described in chapter 3.2. To accommodate the 22050Hz sampling rate of our dataset, we use STFT parameters described in chapter 4.2.3 to compute the mel-spectrogram of speech. Except for the difference of STFT parameters, we use the same parameters and follow the same training procedures described in [36] to train AUTO-VC.

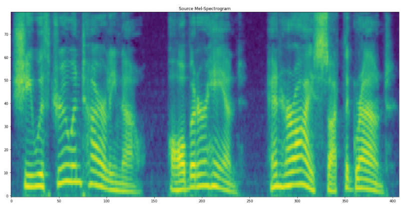
PARA: The major disadvantage of parallel VC is the difficulty of obtaining parallel training data. However, a multi-speaker TTS model like Mellotron can generate speech utterance of any text content, which means that we could generate synthetic parallel utterances of any speaker. Although this thesis focuses on non-parallel VC and our proposed framework transfers knowledge from a TTS model to a VC model, a natural question to ask is: how is the performance of a VC model trained on parallel data generated by a TTS model compared to our proposed model? To answer this question, we train a naive seq2seq parallel data VC model on the same dataset as Mellotron which we call PARA. The PARA model has identical encoder and decoder to our proposed model. The training details are as follows: (1) We sample a batch of text and mel-spectrogram pairs from the dataset, and we randomly generate a batch of speaker IDs. (2) We then pass the text and speaker IDs to the pre-trained Mellotron to synthesize a batch of mel-spectrogram which is our synthetic parallel data. (3) We finally pass the mel-spectrogram sampled from the dataset to PARA along with speaker IDs, and calculated the L2 loss between output of PARA and the synthetic mel-spectrogram. We train PARA with ADAM [24] optimizer and the learning rate is set to $1e-3$. The model is trained with a batch size of 24 for around 100,000 iterations.

4.4 Evaluation

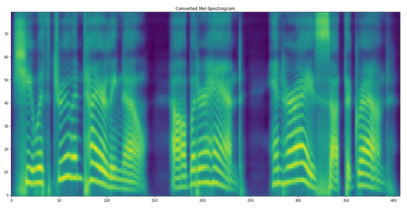


(a) source (b) target (c) pitch contour

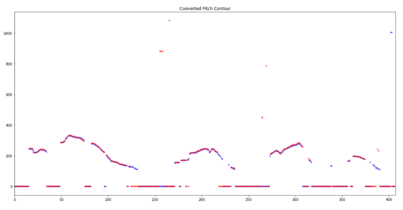
Figure 4.1: Example of male-to-male conversion



(a) source

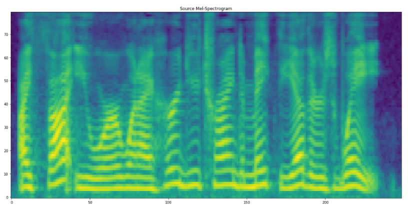


(b) target

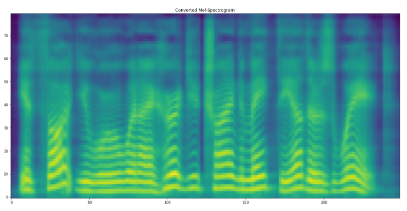


(c) pitch contour

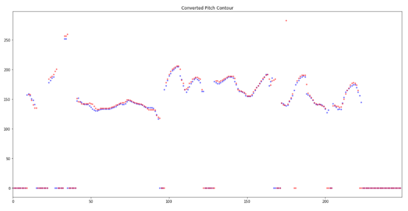
Figure 4.2: Example of male-to-female conversion



(a) source

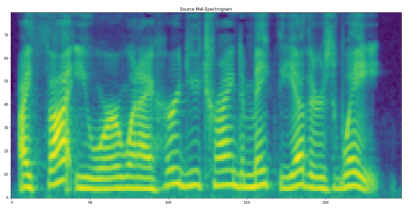


(b) target

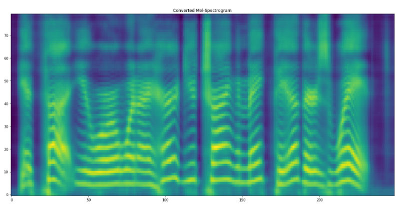


(c) pitch contour

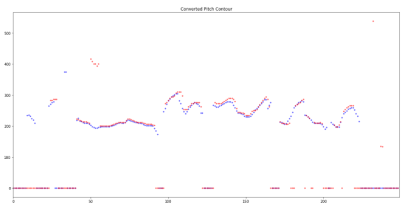
Figure 4.3: Example of female-to-male conversion



(a) source



(b) target



(c) pitch contour

Figure 4.4: Example of female-to-female conversion

4.4.1 Case Study

To investigate the conversion result of our proposed model, we draw several examples of mel-spectrograms of the converted speech. Figure 4.1 to 4.4 are four conversion results that cover four conversion scenarios (male-to-male, male-to-female, female-to-male and female-to-female). Column (a) of the figures are the mel-spectrogram of the source speech and column (b) of the figures are the converted target mel-spectrograms. Figure 4.1 and 4.2 are the conversion results of a male source speaker speaking *"She withdrew entirely now, all but her hand, and her eyes sought the ground."* Figure 4.3 and 4.4 are the conversion results of a female source speaker speaking *"I thought you were when I heard you trying to make the others wait."* It can be seen that the converted mel-spectrograms in column (b) generally follow the envelope of the source mel-spectrograms in column (a). This suggests that the proposed model can keep the linguistic content of the source speech, but some of the high frequency bands in figure 4.1 and figure 4.3 are lost or become indistinct.

In chapter 3.6, we have described how we convert the fundamental frequency from source speaker to target speaker. To show that our proposed model is able to transform fundamental frequency accordingly, we draw the input pitch contour and pitch contour extracted from converted speech in column (c) of figure 4.1 to figure 4.4. The blue points are the conditional input to our proposed model. The red points are pitch contour extracted from converted mel-spectrograms in column (b). As shown in the figures, blue points and red points are mostly aligned which means that the converted speech have the same pitch as the input and our model works as expected. In addition, this result suggests that we have successfully transferred knowledge from Mellotron to our proposed model.

4.4.2 Subjective Evaluation

Traditionally, people measure the performance of voice conversion systems by the mel-cepstral distortion between the converted spectrum and the ground truth target spectrum. Mel cepstral distortion (MCD) is a measure of how different two sequences of mel-cepstrum are. MCD is computed as:

$$\mathrm{MCD[dB]} = \frac{10}{\ln 10}\sqrt{2\sum_{d=1}^{N}(c_d - c_d^{converted})^2} \tag{4.1}$$

where N is the dimension of mel-cepstral features. c_d and $c_d^{converted}$ are the ground truth target and converted features, respectively. However, listening studies have shown that good conversion result does not always relate to small MCD value. Hence, a small MCD is not the necessary condition of a good VC system. Moreover, calculating MCD requires the ground truth target speech which implies the necessity of parallel utterances in the dataset. For these reasons, we do not adopt MCD to evaluate the performance of our VC model. Instead, we follow the common practice [49] to evaluate our models in terms of speaker similarity and naturalness by conducting subjective listening tests.

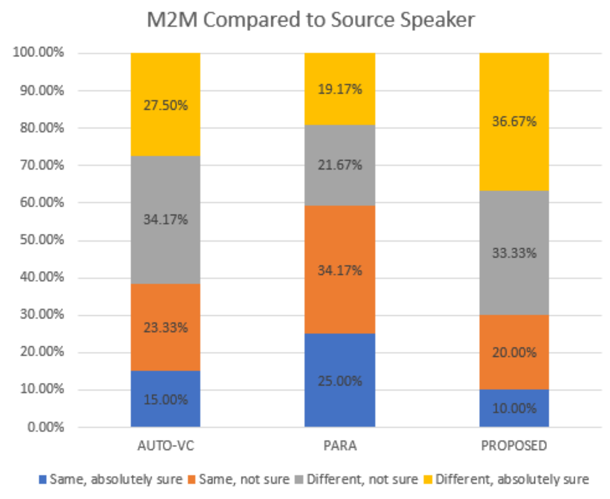
Speaker Similarity Test

The speaker similarity test aims at answering the question: How similar does the converted speech sound compared to the target speaker and to the source speaker?

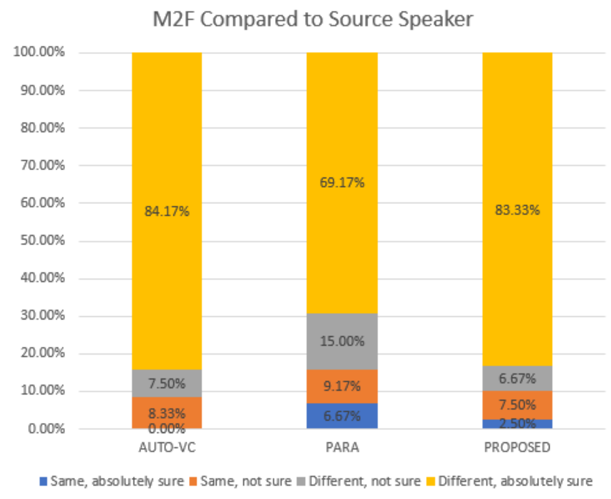
To construct the test, we randomly select 4 speakers (2 male and 2 female) as the source speakers and another 4 speakers (2 male and 2 female) as the target speakers. Each source speaker is paired with the 4 target speakers, and there are 16 pairs in total. For each pair, we select 4 utterances of the source speaker from the test subset and convert them to the target speaker. To control the total length of the listening test, we only select source utterances with length less than 5 seconds. We do the above operations with the same 16 speaker pairs for AUTO-VC, PARA and our proposed model. This results in $16 \times 4 \times 3 = 192$ converted utterances. We then pair each converted utterance with a reference utterance from its source speaker and its target speaker. In total, we have $192 \times 2 = 384$ pairs of utterances, and we call a pair of utterances a test case. We divide test cases into four sets based on the gender of the source and target speakers: male-to-male(M2M), male-to-female(M2F), female-to-male(F2M) and female-to-female(F2F). Each set contains 96 test cases.

To conduct the listening test, we recruit 16 listeners (7 male, 9 female). All the listeners have received higher education and are fluent in speaking English as a second language. We ask the listeners to listen to the test cases and evaluate the speaker similarity of the utterances. Specifically, each listener is given the following instructions: *"Do you think these two samples could have been produced by the same speaker? Some of the samples may sound somewhat degraded/distorted. Please try to listen beyond the distortion and concentrate on identifying the voice. Are the two voices the same or different?"* and asked to judge on the test cases with the following options: *"same, absolutely sure"*, *"same, not sure"*, *"different, not sure"* and *"different, absolutely sure"*. In total, each set is listened for six times. In other words, each test case is evaluated by 6 listeners.

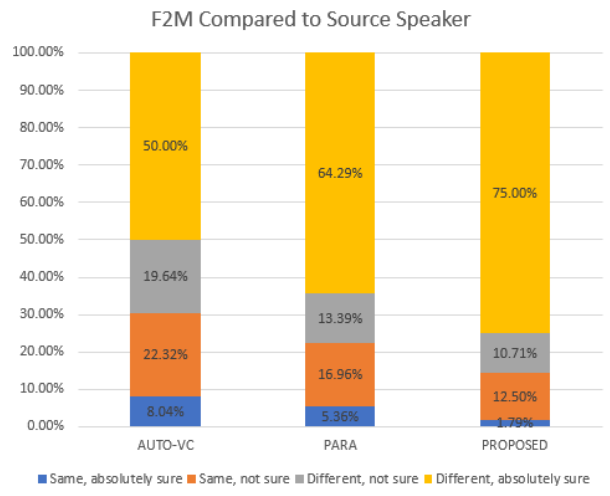
Figure 4.5 presents the similarity test result of the 3 models compared to the source speakers. Ideally, we want a VC model to cast out the source speaker identity, and the converted speech should sound as different from the source speaker as possible. As shown in the figure, in M2F conversion, both AUTO-VC and our proposed model achieve around



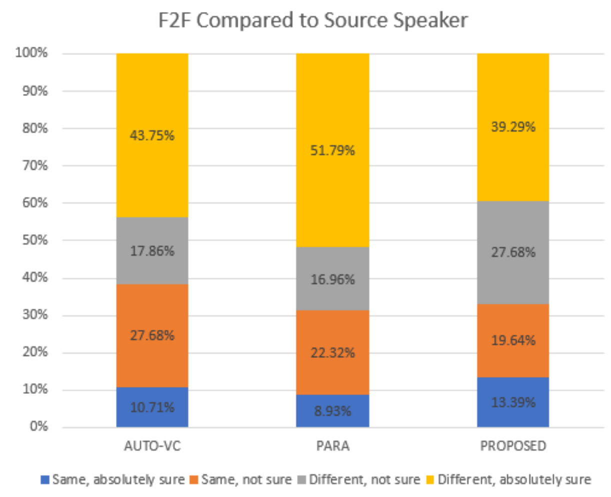
(a) M2M



(b) M2F



(c) F2M

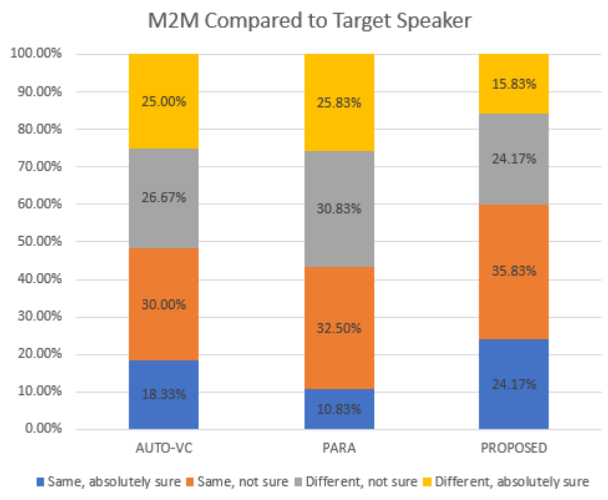


(d) F2F

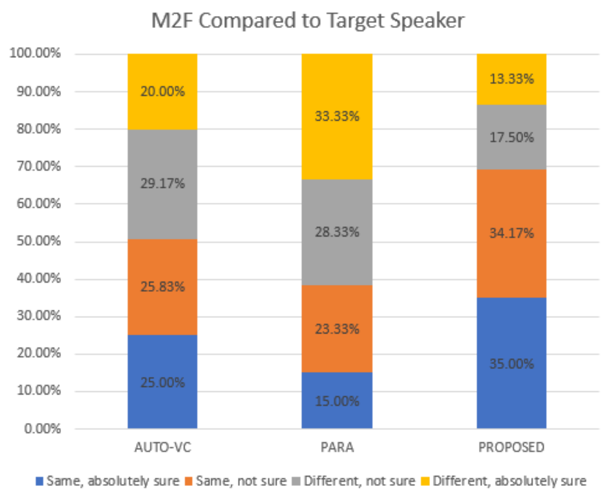
Figure 4.5: Similarity test with the source speaker

90% speaker difference rate. In F2M conversion, nearly 70% of listeners rate the converted speech of AUTO-VC with *"different, not sure"* or *"different, absolutely sure"* and 85% of listeners rate the converted speech of our proposed model with *"different, not sure"* or *"different, absolutely sure"*. On the other hand, in M2M conversion, only about 60% of listeners think that the converted speech of AUTO-VC is uttered by a different speaker from the source speech. Our proposed model out-performs AUTO-VC and achieves 70% speaker difference rate. In F2F conversion, our proposed model performs slightly better than AUTO-VC by achieving 67% speaker difference rate.

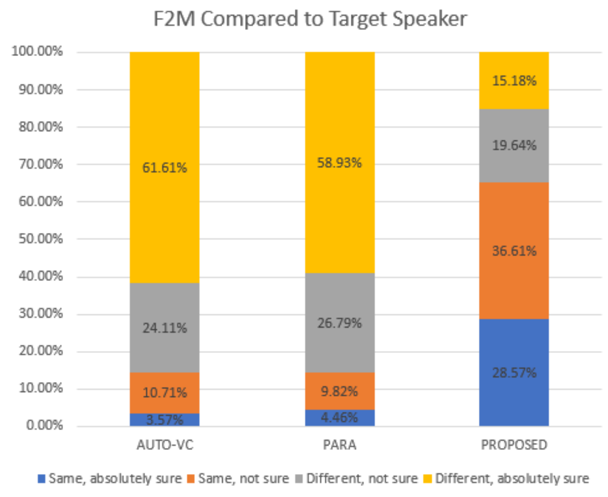
It can be seen that, when performing cross gender voice conversion, all 3 models perform well in casting out source speaker identity. However, when it comes to same gender voice conversion, all 3 models perform worse. PARA even fails to cast out speaker identity in M2M conversion. Our proposed model out-performs AUTO-VC and PARA in all conversion scenarios in casting out the source speaker identity. This result meets our expectation since male and female speakers usually have very different voice characteristics, and the models only need to do little "work" to make the converted voice sound different. Meanwhile speakers with the same gender share a lot of voice characteristics and the models need to be more capable of capturing the voice detailed differences.



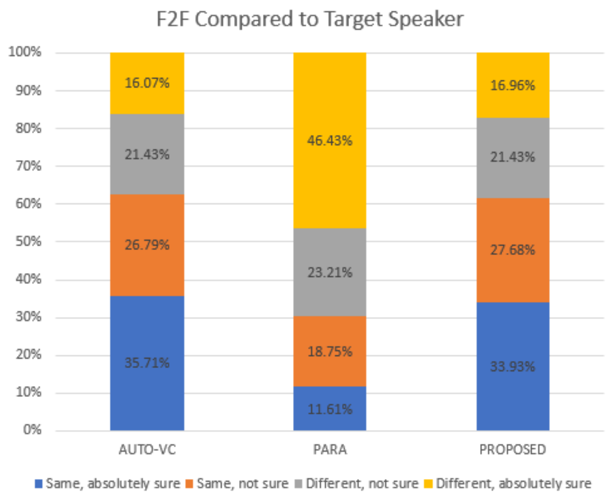
(a) M2M



(b) M2F



(c) F2M



(d) F2F

Figure 4.6: Similarity test with the target speaker

Figure 4.6 presents the similarity test result of the 3 models compared to the target speakers. Not surprisingly, PARA fails to perform a meaningful conversion in most cases, which implies that the naive way of training a parallel VC model with synthetic parallel data does not work.

On the other hand, in M2M conversion, our proposed model gets 60% speaker similarity rate and out-performs AUTO-VC by 10%. In F2F conversion, both AUTO-VC and our proposed model gets 60% speaker similarity rate. In M2F and F2M cross-gender conversion, our proposed model achieves 70% and 64% speaker similarity rate, respectively, which out-performs AUTO-VC by a large margin.

We note that AUTO-VC performs badly in F2M conversion. We suspect that the cause of this is: AUTO-VC cannot properly convert the pitch from source speaker to target speaker since AUTO-VC assumes that a speech segment contains only speaker information and content information. Thus, the converted speech of AUTO-VC still retains the pitch of the source speech. In F2M conversion, this means that the converted speech would have high pitch. Generally, the pitch of female speakers is out of the vocal range of male speakers. As a result, listeners may get confused and rate bad for AUTO-VC. On the other hand, the pitch of male speakers is often within the vocal range of female speakers, which accounts for the better performance of AUTO-VC in M2F conversion.

The above result also suggests that the conversion of fundamental frequency is crucial to the performance of VC models.

Naturalness Test

The naturalness test aims at answering the question: How natural does the converted speech sound?

The most common approach of testing speech naturalness is the Mean Opinion Score (MOS) test. We use the same converted utterances generated in the speaker similarity test for the MOS test. In addition, we randomly select 32 audio recordings from the dataset to evaluate the MOS of ground truth audio recordings for comparison purpose. To construct the MOS test, we make a set of 56 utterances with 16 converted utterances from AUTO-VC, 16 converted utterances from PARA, 16 converted utterances from our proposed model and 8 audio recordings. We repeat this process without replacement, and make 4 test sets.

Similar to the similarity test, we then ask the same 16 listeners to evaluate the naturalness of speech in test sets and rate each utterance on a scale from 1 to 5 with 1 point increments. Each point indicates the naturalness rating of a given utterance with the following levels: *1-bad*, *2-poor*, *3-fair*, *4-good* and *5-excellent*. In total, each listener listens to one set and each set is listened by 4 different listeners.

Table 4.3: Mean Opinion Scores result with 95% confidence intervals computed from normal distribution

Model	**MOS**
AUTO-VC	3.813 ($\pm$0.340)
PARA	3.438 ($\pm$0.273)
Proposed Model	3.719 ($\pm$0.315)
Ground Truth	4.377 ($\pm$0.225)

Table 4.3 shows the MOS result with 95% confidence intervals. We find that PARA has the lowest score while AUTO-VC achieves the highest score. Nevertheless, statistically, we are 95% confident to say that our proposed model and AUTO-VC have no significant difference in their VC performance in terms of speech naturalness. Although it is impractical to compare the absolute score of MOS results with different experimental settings, the relative difference between the MOS result of the ground truth audio recordings and the MOS result of our proposed model is closed to the relative difference of the result in [35]. This implies that our proposed model can generate naturally sounding speech.

In addition to the test, we also ask listeners for feedback. Comments from listeners indicate that our proposed model sometimes makes buzz noise in the converted speech and occasionally makes pronunciation mistakes. For example, our proposed model would convert the word "wait" to the pronunciation in the middle of "wait" and "wet". Since all 3 models use the same Waveglow vocoder, it is very likely that the artifacts come from the VC model instead of the Waveglow vocoder. We have hypothesized several reasons for this. (1) The attention module of the VC model is adopted from Mellotron. However, in Mellotron, the attention module is used to align text representations and mel-spectrogram frames. Hence, The discrepancies between the Mellotron attention module and the VC attention module may cause the issues. (2) Since Mellotron is trained with a mixture of character and phoneme inputs, it may be under-fitting to some pronunciation cases. When we transfer knowledge from Mellotron's decoder, the errors get propagated to our VC model. (3) Despite the success of the proposed framework, L_2 loss only might be too simple for VC task, and we may need additional loss terms (e.g. phoneme classification of the hidden representations) to get better audio quality.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this thesis, we attempt to address the question: "how to disentangle speaker and content information from speech segments for non-parallel many-to-many VC?". The main contribution of our work is that we propose to train a non-parallel many-to-many VC model by directly adopting the decoder and speaker embeddings of a TTS model, Mellotron. In addition, to better disentangle content information from other factors, we propose to condition our proposed model on global style tokens and pitch contour. A simple fundamental frequency conversion trick is proposed for the conversion stage.

To visualize the conversion results, mel-spectrogram figures are presented. To evaluate the performance of our proposed model, we conduct subjective tests, and the result of which implies that our proposed model out-performs the baseline model in terms of speaker similarity. We draw the following conclusions from our experiments. (1) The acoustic decoder of a TTS model can be transferred to a VC model. A well trained TTS decoder can help disentangle speaker and content information for the VC model. (2) The conversion of fundamental frequency is crucial to the performance of VC models. Failure in the conversion of fundamental frequency may result in poor speaker similarity.

Despite the success of our work. There are some improvements that can be made in the future. (1) We may integrate a speaker encoder to our proposed framework so that the model can achieve zero-shot voice conversion. (2) The current trick used for converting fundamental frequency is naive. We may find a more robust way of converting fundamental frequency. (3) We can investigate a better attention mechanism or integrate auxiliary tasks to further improve the audio quality of our proposed model. For example, we may impose a phoneme classification task on the output of speech encoder (4) The efficacy of GST module is to be verified. We may also try to model other factors of speech (prosody, accent, emotion) in a more fine-grained level.

REFERENCES

[1] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara. Voice conversion through vector quantization. In *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pages 655–658 vol.1, 1988.

[2] Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10019–10029. Curran Associates, Inc., 2018.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[4] Alain Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111:1917–30, 05 2002.

[5] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. *CoRR*, abs/1711.09020, 2017.

[6] Ju-Chieh Chou, Cheng-chieh Yeh, and Hung-yi Lee. One-shot voice conversion by separating speaker and content representations with instance normalization. *CoRR*, abs/1904.05742, 2019.

[7] Yan Deng, Lei He, and Frank K. Soong. Modeling multi-speaker latent space to improve neural tts: Quick enrolling new speaker and enhancing premium voice. *ArXiv*, abs/1812.05253, 2018.

[8] S. Desai, E. V. Raghavendra, B. Yegnanarayana, A. W. Black, and K. Prahallad. Voice conversion using artificial neural networks. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3893–3896, 2009.

[9] Srinivas Desai, Alan Black, B. Yegnanarayana, and Kishore Prahallad. Spectral mapping using artificial neural networks for voice conversion. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18:954 – 964, 08 2010.

[10] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[11] Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2962–2970. Curran Associates, Inc., 2017.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[13] Chin-Cheng Hsu, Hsin-Te Hwang, Yi-Chiao Wu, Yu Tsao, and Hsin-Min Wang. Voice conversion from non-parallel corpora using variational auto-encoder. *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–6, 2016.

[14] Wen-Chin Huang, Tomoki Hayashi, Yi-Chiao Wu, Hirokazu Kameoka, and Tomoki Toda. Voice transformer network: Sequence-to-sequence voice conversion using transformer with text-to-speech pretraining. *ArXiv*, abs/1912.06813, 2019.

[15] Wen-Chin Huang, Hsin-Te Hwang, Yu-Huai Peng, Yu Tsao, and Hsin-Min Wang. Voice conversion based on cross-domain features using variational auto encoders. *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 51–55, 2018.

[16] Keith Ito. The LJSpeech dataset. `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[17] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, and Yonghui Wu. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4480–4490. Curran Associates, Inc., 2018.

[18] A. Kain and M. W. Macon. Spectral voice conversion for text-to-speech synthesis. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, volume 1, pages 285–288 vol.1, 1998.

[19] Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. ACVAE-VC: Non-parallel many-to-many voice conversion with auxiliary classifier variational autoencoder. *ArXiv*, abs/1808.05092, 2018.

[20] Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. StarGAN-VC: Non-parallel many-to-many voice conversion with star generative adversarial networks. *CoRR*, abs/1806.02169, 2018.

[21] Hirokazu Kameoka, Kou Tanaka, Takuhiro Kaneko, and Nobukatsu Hojo. ConvS2S-VC: Fully convolutional sequence-to-sequence voice conversion. *CoRR*, abs/1811.01609, 2018.

[22] T. Kaneko and H. Kameoka. CycleGAN-VC: Non-parallel voice conversion using cycle-consistent adversarial networks. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2100–2104, 2018.

[23] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[25] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[26] H. Lu, Z. Wu, R. Li, S. Kang, J. Jia, and H. Meng. A compact framework for voice conversion using WaveNet conditioned on phonetic posteriorgrams. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6810–6814, 2019.

[27] Hieu-Thi Luong and Junichi Yamagishi. Bootstrapping non-parallel voice conversion from speaker-adaptive text-to-speech. *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 2019.

[28] Hieu-Thi Luong and Junichi Yamagishi. A unified speaker adaptation method for speech synthesis using transcribed and untranscribed speech with backpropagation, 2019.

[29] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, 1975.

[30] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. WORLD: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, E99.D:1877–1884, 07 2016.

[31] Toru Nakashika, R. Takashima, T. Takiguchi, and Yasuo Ariki. Voice conversion in high-order eigen space using deep belief nets. *Proc. Interspeech*, pages 369–372, 2013.

[32] O. Ocal, O. H. Elibol, G. Keskin, C. Stephenson, A. Thomas, and K. Ramchandran. Adversarially trained autoencoders for parallel-data-free voice conversion. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2777–2781, 2019.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala.

Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[34] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O. Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep Voice 3: 2000-speaker neural text-to-speech. In *International Conference on Learning Representations*, 2018.

[35] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3617–3621, 2019.

[36] Kaizhi Qian, Yang Zhang, Shiyu Chang, Xuesong Yang, and Mark Hasegawa-Johnson. AutoVC: Zero-shot voice style transfer with only autoencoder loss. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5210–5219, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[37] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, R. J. Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. *CoRR*, abs/1712.05884, 2017.

[38] Y. Stylianou, O. Cappe, and E. Moulines. Continuous probabilistic transform for voice conversion. *IEEE Transactions on Speech and Audio Processing*, 6(2):131–142, 1998.

[39] Yannis Stylianou. Applying the harmonic plus noise model in concatenative speech synthesis. *Speech and Audio Processing, IEEE Transactions on*, 9:21 – 29, 02 2001.

[40] L. Sun, K. Li, H. Wang, S. Kang, and H. Meng. Phonetic posteriorgrams for many-to-one voice conversion without parallel data training. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2016.

[41] K. Tanaka, H. Kameoka, T. Kaneko, and N. Hojo. ATTS2S-VC: Sequence-to-sequence voice conversion with attention and context preservation mechanisms. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6805–6809, 2019.

[42] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[43] Rafael Valle, Jason Li, Ryan Prenger, and Bryan Catanzaro. Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens. *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2020.

[44] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[45] Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Carlos Cobo Rus, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alexander Graves, Helen King, Thomas Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast high-fidelity speech synthesis. Technical report, Google Deepmind, 2017.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[47] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *Interspeech*, 2017.

[48] Yuxuan Wang, Daisy Stanton, Yu Zhang, R. J. Skerry-Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Fei Ren, Ye Jia, and Rif A. Saurous. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. *ArXiv*, abs/1803.09017, 2018.

[49] Mirjam Wester, Zhizheng Wu, and Junichi Yamagishi. Analysis of the voice conversion challenge 2016 evaluation results. In *Interspeech 2016*, pages 1637–1641, 2016.

[50] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. LibriTTS: A corpus derived from librispeech for text-to-speech. *CoRR*, abs/1904.02882, 2019.

[51] Jing-Xuan Zhang, Zhen-Hua Ling, Yuan Jiang, Li-Juan Liu, Chen Liang, and Li-Rong Dai. Improving sequence-to-sequence acoustic modeling by adding text-supervision. *CoRR*, abs/1811.08111, 2018.

[52] Jing-Xuan Zhang, Zhen-Hua Ling, Li-Juan Liu, Yuan Jiang, and Li-Rong Dai. Sequence-to-sequence acoustic modeling for voice conversion. *CoRR*, abs/1810.06865, 2018.

[53] Mingyang Zhang, Xin Wang, Fuming Fang, Haizhou Li, and Junichi Yamagishi. Joint training framework for text-to-speech and voice conversion using multi-source Tacotron and WaveNet. *Interspeech 2019*, Sep 2019.

[54] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.