DEEP GENERATIVE MODELS FOR SPEECH EDITING

BY

KAIZHI QIAN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Doctoral Committee:

Professor Mark A. Hasegawa-Johnson, Chair
Professor Steven E. Levinson
Associate Professor Lav R. Varshney
Doctor Shiyu Chang, MIT-IBM Watson AI Lab

ABSTRACT

Generative models are very useful for generating and modifying natural-sounding speech in various speech processing tasks such as speech synthesis, speech enhancement, and voice conversion. There are two ways that the generative models can help in naturalness for speech processing. The first way is to regularize the speech editing process by defining the sample space of natural speech, and the second way is by permitting the separable modification of components of hierarchical speech generative models to modify specified components of natural speech. In particular, four research projects are introduced, where the first two use WaveNet as the clean speech generative model for single-channel and multi-channel speech enhancement; and the last two projects modify different speaking styles by modeling different speech components using autoencoders.

Multi-channel speech enhancement with ad-hoc sensors has been a challenging task. Speech model guided beamforming algorithms can recover natural-sounding speech, but the speech models tend to be oversimplified to prevent the inference from becoming too complicated. On the other hand, deep learning-based enhancement approaches can learn complicated speech distributions and perform efficient inference, but they are unable to deal with a variable number of input channels. Also, deep learning approaches introduce many errors, particularly in the presence of unseen noise types and settings. Therefore an enhancement framework called DeepBeam is proposed, which combines the two complementary classes of algorithms. DeepBeam introduces a beamforming filter to produce natural-sounding speech, but the filter coefficients are determined with the help of a WaveNet-based monaural speech enhancement model. Experiments on synthetic and real-world data show that DeepBeam can produce clean, dry, and natural-sounding speech, and is robust against unseen noise.

For single-channel speech enhancement, the existing deep learning-based

methods still have two limitations. First, the Bayesian framework is not adopted in many such deep-learning-based algorithms. Second, the majority of the existing methods operate on the frequency domain of the noisy speech, such as the spectrogram and its variations. A Bayesian speech enhancement framework, called BaWN (Bayesian WaveNet) is proposed, which directly operates on raw audio samples. It uses the WaveNet as the prior model to regularize the output to be in the speech space and thus improving the performance. Experiments show that BaWN can recover clean and natural speech.

Non-parallel many-to-many voice conversion, as well as zero-shot voice conversion, remain under-explored areas. Deep style transfer algorithms, such as generative adversarial networks (GAN) and conditional variational autoencoder (CVAE), are popular solutions in this field. However, GAN training is sophisticated and difficult, and there is no strong evidence that its generated speech is of good perceptual quality. On the other hand, CVAE training is simple but does not come with the distribution-matching property as in GAN. A new style transfer scheme that involves only an autoencoder with a carefully designed bottleneck is proposed. This scheme can achieve distribution-matching style transfer by training only on a self-reconstruction loss. Based on this scheme, AutoVC is proposed, which achieves state-of-the-art results in many-to-many voice conversion with non-parallel data, and which is also the first to perform zero-shot voice conversion.

Speech information can be roughly decomposed into four components: language content, timbre, pitch, and rhythm. Obtaining disentangled representations of these components is useful in many speech analysis and generation applications. Recently, state-of-the-art voice conversion systems have led to speech representations that can disentangle speaker-dependent and independent information. However, these systems can only disentangle timbre, while information about pitch, rhythm, and content is still mixed. Further disentangling the remaining speech components is an under-determined problem in the absence of explicit annotations for each component, which are difficult and expensive to obtain. To further explore this problem, we propose SpeechSplit, which can blindly decompose speech into its four components by introducing three carefully designed information bottlenecks. SpeechSplit is among the first algorithms that can separately perform style transfer on timbre, pitch, and rhythm without text labels.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to acknowledge my graduate advisor, Professor Mark Hasegawa-Johnson, who has given me lots of research opportunities, guidance and insights. His broad knowledge and research attitude deeply cultivated me to become an independent, innovative and upright researcher.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

With the development of computers and intelligent systems in almost every aspect of our life, such as laptops, smartphones, and smart homes, etc., human-machine interaction using speech becomes increasingly important. A typical application uses computers to modify or generate speech for human consumption, where naturalness is the most important requirement. In particular, speech enhancement and voice style transfer are the two tasks that involve generating and modifying natural speech respectively.

Speech enhancement in general aims to suppress all interference presented in speech signals. It tries to extract as much as possible useful speech information when the speech is contaminated by noise, reverberation, or even another unwanted speech signal, which can be further divided into speech denoising, speech deverberation, or speech separation. Depending on the problem settings, single-channel enhancement and multi-channel enhancement are the two types of speech enhancement schemes, where the former deals with noisy speech from a single source, and the latter deals with noisy speech from multiple sources. The enhanced speech can either be consumed by computers, such as an automatic speech recognition system, or by human listeners, such as voice communication systems, which require the enhanced speech to sound natural. Zhang et al. [1] show that people prefer noisy but natural speech than clean but unnatural speech.

Voice style transfer aims to modify the voice characteristics, such as timbre, pitch, and rhythm, to sound like another speaker's voice, without altering the linguistic content of the speech. This task can be further divided into timbre conversion, pitch conversion, and rhythm conversion, of which timbre conversion is conventionally referred to as voice conversion. Timbre conversion is relatively more studied than pitch conversion and rhythm conversion. The converted speech is typically intended for human consumption, such as customizing avatar voices and voiceover for videos. These applications again

require the converted speech to sound natural.

Speech enhancement and voice style transfer both aim to modify or generate different components of speech. Here we define such tasks as speech editing. This thesis will focus on speech editing for human consumption, where naturalness is very important. Generative models can be very useful for speech editing because they can characterize the full distribution of natural speech. Therefore, our problem is to determine how generative models help in naturalness for speech editing.

Generative models [2] refer to an important class of models in machine learning, which are able to generate data from the corresponding distribution. Traditional generative models such as the Gaussian mixture model (GMM) [3], hidden Markov model (HMM) [4], and linear predictive coding (LPC) has been extensively used in speech processing. These models all make strong assumptions on speech signals and attempt to fit the parameters of the assumed distributions or models. For example, GMM assumes the data samples are generated from the weighted sum of a finite number of Gaussian distributions, and LPC models the characteristics of the vocal tract based on the source-filter assumption of human speech production [5]. However, the assumed distributions or models may be different from the ground truth, and the bounded number of parameters makes the models difficult to benefit from the unbounded amounts of data in the information era. These traditional models are not suitable for speech editing tasks because they usually suffer from speech artifacts.

In recent years, due to the availability of data and the development of parallel computing, Deep neural networks (DNNs) have gained a large amount of research attention. Given enough data and multi-layer networks with nonlinear activation functions, DNNs are capable of fitting almost any piece-wise smooth functions [6]. Various DNN structures and algorithms have been proposed and achieved state-of-the-art performance in many popular fields such as computer vision and natural language processing. For acoustic modeling, a deep generative model named WaveNet [7] is able to generate high-quality natural-sounding speech that is almost indistinguishable from real human speech, which makes it a very promising candidate model for improving the quality of any speech generation related tasks. Besides modeling the speech sample space as a whole, hierarchical generative models such as autoencoders can model the different characteristics of speech as components and how these

components interact to generate natural speech.

For speech editing tasks, there are two methods in which generative models can be used to ensure speech naturalness. The first method is to regularize the speech editing process by defining the sample space of natural speech, and the second way is permitting the separable modification of components of hierarchical speech generative models in order to modify specified components of natural speech. For the first method, Chapter 3 and Chapter 4 are successful examples of using the WaveNet for regularizing the speech enhancement process. Specifically, in Chapter 3, a beamformer is guided by the output of a WaveNet-based speech model to perform speech beamforming for a multi-channel ad-hoc microphone array. In Chapter 4, a prior model is trained for natural speech using WaveNet and incorporated in a Bayesian framework to regularize a single-channel speech enhancement model. In these applications, the WaveNet-based speech model defines the natural speech sample space and thus can project any unnatural output of the enhancement model into the natural speech sample space to make it natural. In addition, for the second method, in Chapter 5 and Chapter 6, we are able to convert different aspects of the speech by disentangling these aspects using autoencoders with information constraining bottlenecks. Specifically, in Chapter 5, we can modify the speaker identity of the speech by disentangling content and timbre in an unsupervised manner. In Chapter 6, we can convert not only timbre but also pitch and rhythm separately by disentangling content, timbre, pitch, and rhythm without using any text labels. In these applications, since the model only learns to produce natural speech by modeling different components of speech, the output will always be natural speech regardless of the inputs.

The remainder of the thesis is organized as follows. Chapter 2 introduces the background of speech enhancement, voice style transfer, and deep generative models. Chapter 7 discusses the relationships among the four research attempts introduced from Chapter 3 to Chapter 6 and how the thesis contributed to the methods of speech editing. Finally, Chapter 8 states the conclusion of the thesis.

CHAPTER 2

BACKGROUND

2.1 Speech Enhancement

Speech enhancement in general aims to suppress all interference presented in speech signals. It tries to extract as much as possible useful speech information when the speech is contaminated by noise, reverberation, or even another unwanted speech signal, which can be further divided into speech denoising, speech deverberation, or speech separation. Depending on the problem settings, single-channel enhancement and multi-channel enhancement are the two types of speech enhancement schemes, where the former deals with noisy speech from a single source, and the latter deals with noisy speech from multiple sources. This thesis focuses on both single-channel speech enhancement and multi-channel speech enhancement for human consumption, where the naturalness of the enhanced speech is very important.

2.1.1 Multi-Channel Speech Enhancement

The problem of interest is how to perform speech enhancement on an ad-hoc microphone array formed by the microphones on everyone's laptops or cellphones in a conference room setting. This type of problem is traditionally solved using beamforming, which in general tries to optimally combine the microphone signals under some constraints [8]. One classical algorithm is Minimum Variance Distortionless Response (MVDR) [9], which minimizes the output power of the beamformer while preserving the signal coming from the direction of interest. Variants of the MVDR beamformer have been extensively explored for multi-channel speech enhancement [10, 11, 12, 13]. Among these algorithms, the Linear Constraint Minimum Variance (LCMV) [14] beamformer enables more than one direction of interest versus the MVDR.

Later, Griffiths et al. [15] proposed the Generalized Sidelobe Canceler (GSC) structure to separately perform the output power minimization and the application of constraints [16, 17]. However, given this ad-hoc microphone array where both the locations and the frequency responses of the microphones are unknown, traditional beamforming algorithms do not work well because they usually need to calibrate the speaker locations and the interference characteristics to turn the beams toward the speaker while suppressing the interference. For example, the performance of MVDR deteriorates after including far-field microphones [18], and GSC suffers from signal cancellation problem if position calibration is inaccurate [19]. To avoid measuring the speaker locations and the interference characteristics, some research efforts have attempted to find the optimal beamformer by setting and optimizing the features or characteristics that distinguish speech from all other interference. In particular, Gillespie et al. [20] performed dereverberation on ad-hoc microphone arrays by maximizing the kurtosis of the linear prediction residuals based on the observation that the distribution of peaks of the linear prediction residuals of the clean speech is different from that of the reverberated speech [21]. Kim et al. [22] used the prior distributions of the frequency bins of speech as a distinctive feature to separate speech from interference. Similarly, Kumantani et al. [23] assume that the speech distribution should be as non-Gaussian as possible. Later, Zhang et al. [1] tried to fit the glottal residual of the beamformer output to that of the clean speech. The same idea behind these works is to make the output as close as possible to clean speech in terms of some target criterion using prior knowledge on clean speech. Better target criteria may lead to better results. However, their prior knowledge is often too simplified to characterize the true distribution of clean speech. If there exists an ideal target criterion that characterizes the true distribution of clean natural speech, the output will ideally be natural and free of interference.

2.1.2 Single-Channel Speech Enhancement

Single-channel speech enhancement is more challenging than multi-channel speech enhancement. This is because the additional channels may provide complementary information that is missing in other channels, which makes

the inference on clean speech relatively easier. Before deep learning gains more popularity, traditional speech enhancement methods such as Wiener filtering [24, 25, 26], spectral subtraction [27, 28, 29], and subspace methods such as principal component analysis (PCA) and non-negative matrix factorization (NMF) [30, 31, 32] have been explored for many years. Later, deep learning-based methods became the mainstream for speech enhancement due to their strong representation power of characterizing complex noise distributions. These methods are mainly divided into two types. The first type is mapping, which directly predicts clean speech features from noisy speech features [33, 34, 35, 36, 37, 38]. The second type is masking, which predicts masks from noisy speech features [39, 40, 41, 42, 43]. The masks in general indicate the likelihood of the time-frequency bins being speech or noise on the time-frequency representations of noisy speech, such as spectrograms. However, generalization to unseen noise types is a major problem for both the traditional methods and deep learning methods. Traditional methods such as Wiener filtering and spectral subtraction assume statistical properties of speech and noise and require the noise power spectral density (PSD) to work. Meanwhile, most deep learning methods are trained using a finite number of types of noise, but there are typically mismatches between the types of noise in real-world situations and those seen during training, which causes the model performance to degrade in those cases. Although the noise characteristics are difficult to obtain beforehand, the clean speech signal is highly structured and can be modeled beforehand. If the clean speech distribution is known, then any signal that does not follow the clean speech distribution can be removed as noise without knowing the noise distributions or the types of noise. Motivated by this, some works formulate the speech enhancement problem in a Bayesian framework by modeling the clean speech prior distribution. However, these works mostly assume simple models of the distribution of clean speech, such as HMM-GMM [44, 45, 46, 47], or Laplacian models [48, 49, 22, 50], which may not accurately match the true distribution of clean speech. In addition to the generalization problem, speech naturalness also needs improvement for the existing methods. Most deep learning methods including traditional methods such as spectral subtraction and non-negative matrix factorization operate on the amplitude of the time-frequency representations of speech such as magnitude spectrogram. For example, the phase of the noisy spectrogram is directly applied to the enhanced spectrogram to

restore clean speech, which may suffer from phase distortion artifacts. It is possible to estimate the phase of the clean speech to alleviate the phase distortion problem to some extent [51, 52, 53], but it is prone to estimation inaccuracies. Therefore, if there exists a generative model that accurately models the clean speech distribution in raw waveform, it can be used as a clean speech before regularizing the speech enhancement process formulated in a Bayesian framework, which may help to simultaneously improve the generalization to unseen noise and speech naturalness.

2.2 Voice Style Transfer

In general, voice style transfer refers to the task of converting the speaking style of a source speaker into that of a target speaker, while keeping the linguistic contents unchanged. There are different types of voice style transfer. First, in terms of the availability of data, there are parallel conversion [54, 55, 56] and non-parallel conversion [57, 58, 59]. Parallel conversion requires pairs of time-aligned utterances with the same content and different speaking styles for training. This paradigm is impractical because it is very expensive to get pairs of time-aligned utterances with the same content and different speaking styles. In contrast, non-parallel conversion is more practical because it does not require paired utterances for training, but it is technically more challenging. Second, in terms of the number of speakers, there are one-to-one [60], many-to-one [61], one-to-many [62], and many-to-many [63], each of which is more challenging than the previous ones. Finally, in terms of whether the speakers are seen, conventional voice style transfer schemes only convert among the speakers in the training dataset, and none of the existing schemes can convert among unseen speakers.

Different attributes of the speaking style, such as timbre, pitch, and rhythm, can be converted to make it sounds more like the target speaker [64, 65]. Conventionally, timbre conversion is also called voice conversion. However, most of the research attention has been devoted to timbre conversion, where the conversion algorithm focuses on converting the spectral features [66, 67]. More detailed control of the converted voice requires modifications of prosodic features such as pitch and rhythm, which remains under-explored and challenging [68].

For pitch conversion, the widely adopted method is the mean-variance global transformation, which modifies the average level and range of the source pitch contour [69]. It is one of the frame-level methods [70, 71, 72] that keeps the original shape of the contour without matching the detailed intonation of the target speaker. Since pitch contour typically characterizes speech features beyond the level of individual phones, it is more reasonable to convert pitch in various segmental levels. To capture the intonation difference between the source speaker and target speaker, a few contour-based conversion methods were proposed [73, 74, 75]. However, these methods either require parallel data or alignment for creating parallel data to train the model with supervision.

For rhythm conversion, it has been investigated in speech synthesis [76] and voice conversion [75, 77, 78, 79, 80]. Most of these methods use conventional models such as Gaussian mixture models (GMMs) and regular deep neural networks (DNNs), and they require phone boundaries or phone identities from text labels.

Due to limited amounts of parallel data and transcribed data, it is worthwhile to explore the methods of editing speech components without using parallel or transcribed data. The key to success is finding a model that can disentangle the speech components to be converted in an unsupervised manner.

2.3 Deep Generative Models

Traditional generative models such as the Gaussian mixture model (GMM) [3], hidden Markov model (HMM) [4], and linear predictive coding (LPC) have been extensively used in speech processing. In particular, parametric probabilistic generative models such as GMMs are typically used as acoustic models to model the phonetics in speech recognition [81] or the timbre in speaker verification [82]. Besides, signal processing models such as LPC, model the characteristics of the vocal tract based on the classical source-filter assumption of human speech production [5]. These traditional generative models all make strong assumptions on the speech signal and thus are unable to accurately characterize the true distributions of speech or explore latent representations of speech. Thanks to the development of deep learn-

ing, deep generative models are much more flexible and can accurately learn the distributions and the latent representations of speech given enough data. Following, four important deep generative models will be briefly reviewed, among which the WaveNet and vanilla autoencoder are the building blocks of all four research attempts in this thesis.

2.3.1 Autoencoder

The autoencoder is one of the fundamental building blocks of machine learning, from which other systems may be created [83]. In general, a vanilla autoencoder consists of an encoder and a decoder, as shown in Figure 2.1,

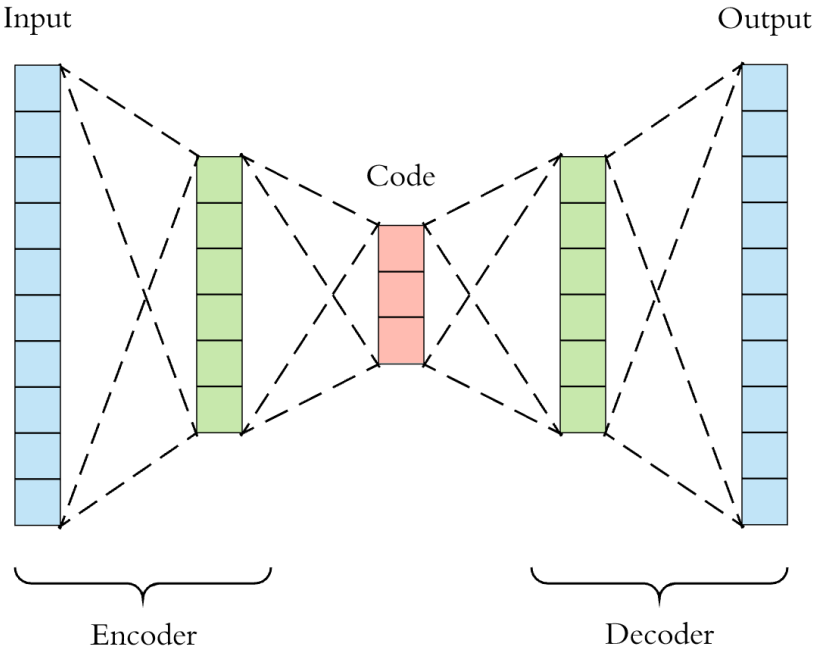


Figure 2.1: Autoencoder structure [84].

where the encoder compresses the input data into a low-dimensional hidden representation and the decoder attempts to reconstruct the input data. The encoder and decoder can use any type of neural network depending on the tasks.

Many variants of vanilla autoencoder have been proposed depending on different applications. For speech processing, denoising autoencoder (DAE) [37, 85, 86] and variational autoencoder (VAE) [87, 88, 89] are two well-known variants of vanilla autoencoder. Denoising autoencoder tries to learn

more informative codes from the input by reconstructing the input from the corrupted input, based on the assumption that the structural level representations are relatively stable to the corruption of the input. The vanilla autoencoder and the denoising autoencoder focus on latent representation discovering and do not explicitly output distributions. In contrast, the variational autoencoder tries to explicitly model the joint distribution of the latent representation and the output, but the output distribution is only an approximation up to a variational lower bound [90]. The sampled outputs from this distribution are well-known to suffer from the over-smoothing effect [91].

The major advantage of autoencoders is that they explicitly discover meaningful latent representations and how the latent representations interact to generate output. This advantage makes autoencoders optimal for learning hierarchical representations of different speech components in an unsupervised manner. When trained using natural speech, a set of properly designed autoencoders may be able to learn disentangled speech components for natural speech editing. Since the autoencoder only learns to reconstruct natural speech, the output speech after editing the components will also be natural speech.

2.3.2 WaveNet

WaveNet is a multi-layer dilated convolutional neural network that is fully probabilistic and autoregressive. The joint probability distribution of the speech samples $\mathbf{X} = \{x_1, \cdots, x_T\}$ is modeled by factoring it into the product of condition distributions as shown in Equation (2.1)

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

where the model predicts the distribution of each sample conditioned on the previous samples. The dilated causal convolution as shown in Figure 2.2 has a very large receptive field size that can capture the long-range temporal dependencies.

WaveNet is simply trained using many hours of speech without making any assumptions about the speech production process. During generation, each

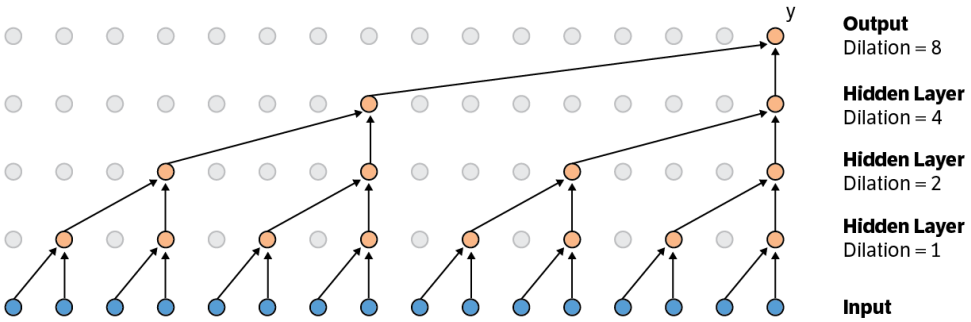


Figure 2.2: Dilated causal convolution neural network [7].

sample is generated by conditioning on the previously generated samples.

If trained only using waveforms, the model generates speech-like babbles smoothly with realistic sounding intonations. If trained by conditioning on other features, the model can generate audio with the required characteristics. There are two ways to condition the model on other variables: global conditioning and local conditioning, where the former influences predictions across all time steps, and the latter guides predictions for each time step. It has been shown that by global conditioning on speaker embeddings [7], the model can generate voice from multiple speakers; and by local conditioning on mel-spectrograms, the model can synthesize the corresponding audio of the mel-spectrograms [92]. One can build a multi-speaker mel-spectrogram based vocoder by training the WaveNet conditioned on mel-spectrograms of multiple speakers.

Although WaveNet can fully characterize the speech waveform distribution, it does not discover the latent representations of speech, which prevents us from editing the output by manipulating the latent representations. In addition, the inference speed of WaveNet is slow because it generates samples autogressively. Nevertheless, the ability to accurately model speech waveform makes WaveNet perfect for speech enhancement. It can be used as a clean speech prior model for single-channel speech enhancement or a clean speech predictor for multi-channel speech enhancement.

2.3.3 Generative Adversarial Network

A generative adversarial network (GAN) [93] is a class of machine learning scheme that trains the generative model using the discriminator in a two-player minmax game. The general framework of a GAN is shown in Figure 2.3

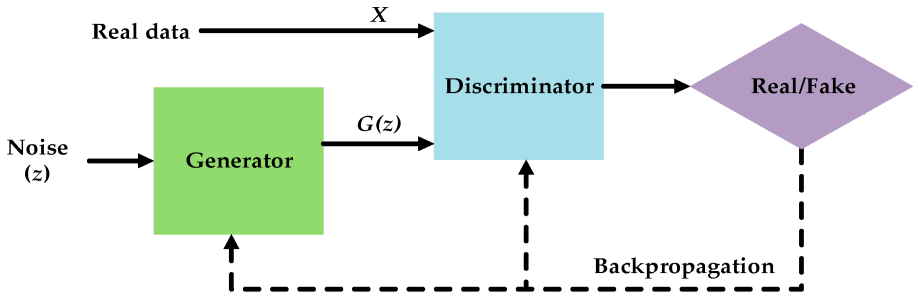


Figure 2.3: GAN framework [94].

where we denote the noise distribution and the data distribution as p_z and p_{data} respectively. The generator maps the input noise to the data. The discriminator outputs the probability of its input coming from real data.

During training, the generator tries to produce data samples that are real enough for the current discriminator. Meanwhile, the discriminator tries to differentiate between the real data and the data produced by the current generator. The generator and the discriminator are updated in turns until the generator is able to produce data samples as if they are drawn from the real data distribution. The training process can be viewed as a two-player minmax game between the generator and the discriminator with value function $V(D, G)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}\,(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (2.2)$$

The generator in a GAN is able to accurately learn the data distribution in an unsupervised manner, which is one of GAN's major advantages because it is labor-intensive to get large amounts of labeled data. However, adversarial training is very unstable and difficult to converge. In addition, the generator may suffer from mode collapse if the optimization between the generator and the discriminator is unbalanced.

GANs have been gaining huge research attention soon after it was pro-

posed, especially in image and computer vision areas. New GAN papers are coming out every week, and thus it is impossible to keep track of all of them. Those research attempts can roughly be categorized into two tracks. The first track is solving existing problems. For example, WGAN [95, 96], LS-GAN [97], and BEGAN [98] try to solve the unstable training problem and the mode collapse problem. The second track is creating new architectures for different applications. For image processing, DCGAN [99] and BigGAN [100] greatly improved the output image quality. StyleGAN [101] and Cycle-GAN [102] promoted the GAN-based image style transfer. StackGAN [103] is able to synthesize images from text. In speech processing, SEGAN [104], StarGAN-VC [58], and WaveGAN [105] used GAN for speech enhancement, voice conversion, and speech synthesis respectively. Although there are many variants of GAN, the fundamental principle behind them remains the same.

2.3.4 Generative Flow

The flow-based generative models [106] model the input distribution by first learning an invertible mapping from the input to a simple auxiliary distribution, such as Gaussian distribution. Then, the input distribution can be directly computed by taking the inverse of the mapping, as shown in Figure 2.4

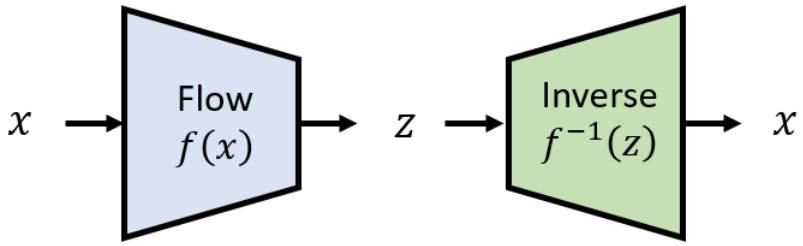


Figure 2.4: Flow based model.

The input distribution is learned once the mapping from the input to the auxiliary distribution is also learned, which saves the computation to learn both. Since the input distribution is typically complex whose exact analytic form is unknown, it is impractical to directly map from the auxiliary distribution to the input distribution. However, the inverse mapping is easier and tractable to learn, which makes it possible to indirectly learn the complex input distribution. Moreover, since the data is not autoregressively learned,

the inference speed of the flow-based models is much faster than that of the WaveNet. One disadvantage is that the invertible layers may limit the representation power of the model. As a result, the output quality of the flow-based models is not as high as that of the state-of-the-art GAN-based models. Nevertheless, flow-based models still achieve promising results on speech synthesis [107].

CHAPTER 3

DEEP LEARNING BASED BEAMFORMING

3.1 Introduction

Multi-channel speech enhancement with ad-hoc sensors has long been a challenging task [108]. As the traditional benchmark in multi-channel enhancement tasks, beamforming algorithms do not work well with ad-hoc microphones. This is because most beamformers need to calibrate the speaker location as well as the interference characteristics, so that it can turn its beam toward the speaker, while suppressing the interference. However, neither of the two vital information can be accurately measured, due to the missing sensor position information and microphone heterogeneity [109].

Another class of beamforming algorithms avoid measuring the speaker position and interference. Instead, they introduce prior knowledge on speech, and find the optimal beamformer by maximizing the "speechness" criteria, such as sample kurtosis [20], negentropy [23], speech prior distributions [22, 50], fitting glottal residual [1] etc. In particular, the GRAB algorithm [1] is able to outperform the closest microphone strategy even in very adverse real-world scenarios. Despite their success, these algorithms are bottlenecked by their oversimplified prior knowledge. For example, GRAB only models glottal energy, resulting in vocal tract ambiguity.

On the other hand, deep learning techniques are well known for their ability to capture complex probability dependencies and efficient inference, and thus have been widely used in single-channel speech enhancement tasks [110, 111, 112, 113, 114, 115, 104]. Unfortunately, directly applying deep enhancement networks to multi-channel enhancement suffers from three difficulties. First, the number of channels is variable, and can be any positive integers. However, designing a network structure that has variable input dimension is rather challenging. Second, deep enhancement techniques of-

ten produce a lot of artifacts and nonlinear distortions [114, 115], which are perceptually undesirable. Third, neural networks often generalize poorly to unseen noise and configurations, whereas in multi-channel speech enhancement with ad-hoc sensors, the degree of such variability is large.

As it turns out, these problems can in turn be resolved by traditional beamforming. Motivated by this observation, we have proposed an enhancement framework called DeepBeam, which combines the two complementary classes of algorithms. DeepBeam introduces a beamforming filter to produce natural sounding speech, but the filter coefficients are iteratively determined with the help of a monaural speech enhancement neural network.

DeepBeam demonstrates the beauty of combining signal processing based beamforming and deep learning. The latter helps the former capture complex speech distribution; while the former can accommodate any number of channels, and eliminate the errors and artifacts made by the latter. It can be shown that despite the error-prone enhancement network, DeepBeam is able to converge approximately to the optimal beamformer under some assumptions. Experiments on both the simulated and real-world data show that DeepBeam is able to produce clean, dry and natural sounding speech, and generalize well to various settings.

3.2 Problem Formulation

To formally define the problem, denote $s[t]$ as the clean speech signal. Suppose there are K channels of observed signals, $y_k[t], k = 1, \cdots, K$, which are represented as

$$y_k[t] = s[t] * i_k[t] + n[t] * j_k[t] \tag{3.1}$$

where $*$ denotes discrete convolution, $n(t)$ denotes additive noise, and $i_k[t]$ and $j_k[t]$ are the impulse responses of the signal reverberation and noise reverberation in the k-th channel, respectively. Our goal is to design a τ-tap beamformer $h_k[t], k = 1, \cdots, K$, whose output is defined as

$$x[t] = \sum_{k=1}^{K} y_k[t] * h_k[t] \tag{3.2}$$

For notational brevity, define

$$\begin{aligned}
&\boldsymbol{s} = [s[1], \cdots, s[T]]^T \quad \boldsymbol{x} = [x[1], \cdots, x[T]]^T \\
&\boldsymbol{y}_k = [y_k[1], \cdots, y_k[T]]^T \quad \boldsymbol{y} = [\boldsymbol{y}_1^T, \cdots, \boldsymbol{y}_K^T]^T \\
&\boldsymbol{h} = [h_1[1], \cdots, h_1[\tau], h_2[1], \cdots, h_K[\tau]]^T
\end{aligned} \tag{3.3}$$

which are all random vectors. Also define convolutional matrices

$$\boldsymbol{Y}_k = \begin{bmatrix}
y_k[1] & & & \\
y_k[2] & y_k[1] & & \\
\vdots & \vdots & \ddots & \\
y_k[\tau] & y_k[\tau-1] & \cdots & y_k[1] \\
\vdots & \vdots & & \vdots \\
y_k[T] & y_k[T-1] & \cdots & y_k[T-\tau+1]
\end{bmatrix} \tag{3.4}$$

and

$$\boldsymbol{Y} = [\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_K] \tag{3.5}$$

With these notations, Equation (3.2) can be simplified as

$$\boldsymbol{x} = \boldsymbol{Y}\boldsymbol{h} \tag{3.6}$$

The target of designing the beamformer is to minimize the weighted mean squared error (MSE):

$$\min_{\boldsymbol{x}=\boldsymbol{Y}\boldsymbol{h}} \mathbb{E}\left[\|\boldsymbol{x} - \boldsymbol{s}\|_{\boldsymbol{W}}^2 | \boldsymbol{y}\right] \tag{3.7}$$

where $\|\boldsymbol{x}\|_{\boldsymbol{W}}^2 = \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x}$; $\boldsymbol{W}$ is a positive definite weight matrix, which, in our case, is a diagonal matrix of $\mathrm{Var}^{-1}(s[t]|\boldsymbol{y})$.

Equation (3.7) is a Wiener filtering problem [116], whose solution is

$$\boldsymbol{x}^* = \boldsymbol{P}\mathbb{E}[\boldsymbol{s}|\boldsymbol{y}] \tag{3.8}$$

where

$$\boldsymbol{P} = \boldsymbol{Y}(\boldsymbol{Y}^T\boldsymbol{W}\boldsymbol{Y})^{-1}\boldsymbol{Y}^T\boldsymbol{W} \tag{3.9}$$

is in fact the *projection matrix* onto the beamforming output space. So by Equation (3.8), $\boldsymbol{x}^*$ is essentially projecting $\mathbb{E}[\boldsymbol{s}|\boldsymbol{y}]$ onto the space that is representable by the beamforming filter.

As shown by Equation (3.8), solving the Wiener filtering problem requires

computing $\mathbb{E}[\boldsymbol{s}|\boldsymbol{y}]$, which, due to the complex probabilistic dependencies, we would like to introduce a deep neural network to learn. However, as discussed, training a neural network to directly predict $\mathbb{E}[\boldsymbol{s}|\boldsymbol{y}]$ from the multi-channel input $\boldsymbol{y}$ suffers from inflexible input dimensions, artifacts and poor generalization. DeepBeam tries to resolve these problems and find an approximate solution.

3.3 The DeepBeam Framework

In this section, we will describe the DeepBeam algorithm. We will first outline the algorithm, and then describe the neural network structure it applies. Finally, a convergence analysis is introduced.

3.3.1 The Algorithm Overview

As mentioned, DeepBeam introduces a deep enhancement network to learn the posterior expectation, while addressing its limitations. First, DeepBeam is regularized by the beamformer to generalize well to unseen noise and microphone configurations. Second, it tolerates the distortions and artifacts generated by the neural network. Formally, the neural network outputs an inaccurate prediction of the posterior expectation $\mathbb{E}[\boldsymbol{s}|\boldsymbol{\xi}]$,

$$f(\boldsymbol{\xi}) = \mathbb{E}[\boldsymbol{s}|\boldsymbol{\xi}] + \boldsymbol{\varepsilon}(\boldsymbol{\xi}) \tag{3.10}$$

where $\boldsymbol{\xi}$ is a *single-channel* noisy observation, and $\boldsymbol{\varepsilon}(\boldsymbol{\xi})$ is the prediction error. The goal of DeepBeam is to approximate the optimal beamformer given the inaccurate enhancement network. Algorithm 1 shows the description of the DeepBeam algorithm. A graph of the DeepBeam framework is shown in Figure 3.1.

Algorithm 1 essentially alternates between the posterior expectation and projection iteratively. It will be shown in Section 3.3.3 that as long as the error term $\boldsymbol{\varepsilon}$ is not too large, this iteration will approximately converge to the optimal beamformer output.

One elegance of DeepBeam is that $\boldsymbol{x}^{(n)}$ can be regarded as a noisy observation, and shares some statistical structures with the true noisy observa-

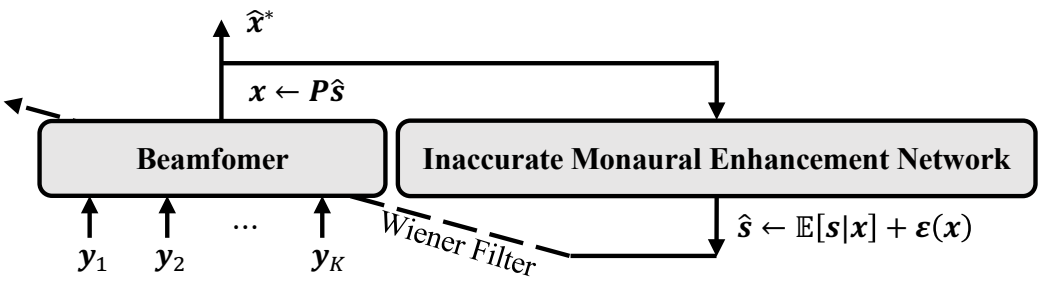


Figure 3.1: DeepBeam framework.

tions, $\boldsymbol{y}_k$. To see this, notice that by Equation (3.12), $\boldsymbol{x}^{(n)}$ is the output of a beamformer on $\boldsymbol{y}$. Therefore, it can be shown that $\boldsymbol{x}^{(n)}$ also takes the form of Equation (3.1), with the same speech and noise source, but with a different impulse response. This justifies the use of one monaural enhancement network to take care of all the $\boldsymbol{x}^{(n)}$.

3.3.2 Enhancement Network Structure

DeepBeam is a general framework, in which the choice of the neural network structure is not fixed. The following network structure is just one of the structures that produce competitive results.

The enhancement network applied here is similar to [115], which is inspired by WaveNet [7]. Formally, denote the *quantized* speech samples as $\tilde{s}[t]$, and the samples of $\boldsymbol{x}^{(n)}$ as $x^{(n)}[t]$. Then the enhancement network predicts the posterior probability mass function (PMF) of $\tilde{s}[t]$:

$$p(\tilde{s}[t]|\boldsymbol{x}^{(n)}) \approx p(\tilde{s}[t]|x^{(n)}[t-\tau_r], \cdots, x^{(n)}[t+\tau_r]) \tag{3.13}$$

Here we have restricted the probabilistic dependency to span τ_r time steps. Cross-entropy is applied as the loss function.

Similar to WaveNet, the enhancement network consists of two modules. The first module, called the dilated convolution module, contains a stack of dilated convolutional layers with residual connections and skip outputs. The second module, called the post-processing module, sums all the skip outputs and feeds them into a stack of fully connected layers before producing the final output.

There are two major differences from the standard WaveNet structure.

Algorithm 1 The DeepBeam algorithm.

Input: Multi-channel noisy speech observations $\boldsymbol{y}$;
A neural network that predicts $f(\boldsymbol{\xi})$ (Equation (3.10)) from any single-channel noisy observation $\boldsymbol{\xi}$.
Output: Beamformer output $\hat{\boldsymbol{x}}^*$.

Initialization:
Find the "cleanest" channel k^* by finding the channel that has the smallest 0.4 quantile of its squared sample points.
Set $\boldsymbol{x}^{(0)} = \boldsymbol{y}_{k^*}$.
Iteration:
for $n = 1$ to maximum number of iterations **do**
Feed $\boldsymbol{x}^{(n-1)}$ to the monaural enhancement network, and obtain its output

$$\hat{\boldsymbol{s}}^{(n)} = f(\boldsymbol{x}^{(n-1)}) = \mathbb{E}[\boldsymbol{s}|\boldsymbol{x}^{(n-1)}] + \boldsymbol{\varepsilon}(\boldsymbol{x}^{(n-1)}) \tag{3.11}$$

Update the beamformer coefficients and output

$$\boldsymbol{x}^{(n)} = \boldsymbol{P}\hat{\boldsymbol{s}}^{(n)} \tag{3.12}$$

end for
return $\hat{\boldsymbol{x}}^* = \boldsymbol{x}^{(N)}$

First, the input to the enhancement network is the noisy observation waveform $\boldsymbol{x}^{(n)}$ instead of the clean speech. Second, to account for the future dependencies, the convolutional layers are noncausal 1×3 instead of the causal 1×2.

After the posterior distribution is predicted, the posterior moments denoted as $\boldsymbol{E}[\boldsymbol{s}|\boldsymbol{x}^{(n)}]$ and $\mathrm{Var}[s[t]|\boldsymbol{y}]$ (for computing $\boldsymbol{W}$), are computed as the moments of the predicted PMF.

3.3.3 Convergence Analysis

In order to analyze the convergence property of DeepBeam, we assume the following bound on the error term:

$$\mathbb{E}[\|\boldsymbol{P}\boldsymbol{\varepsilon}(\boldsymbol{x}^{(n)})\|_{\boldsymbol{W}}^2|\boldsymbol{y}] \leq \rho\mathbb{E}[\|\boldsymbol{x}^{(n)} - \boldsymbol{s}\|_{\boldsymbol{W}}^2|\boldsymbol{y}] \tag{3.14}$$

where $\rho < 0.5$ is some constant. This assumption is actually not quite stringent, because it bounds not the weighted norm of $\boldsymbol{\varepsilon}(\boldsymbol{x}^{(n)})$ itself, but

its projected value $\boldsymbol{P}\boldsymbol{\varepsilon}(\boldsymbol{x}^{(n)})$. In fact, the projection can drastically reduce the weighted norm of the error term. For example, most of the artifacts and nonlinear distortions that the enhancement network introduces cannot possibly be generated by beamforming on $\boldsymbol{y}$, and therefore will be removed by the projection. The only errors that are likely to remain are residual noise and reverberations. This is one advantage of combining beamforming filter and neural network. This assumption is also very intuitive. It means that the projected output error is always smaller than input error.

Then, we have the following theorem.

Theorem 1. *Suppose Equation* (3.14) *holds. Then*

$$\lim_{n\to\infty}\sup\mathbb{E}[\|\boldsymbol{x}^{(n)}-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}]\leq u \tag{3.15}$$

where

$$\begin{aligned} u=&\frac{2\rho}{1-2\rho}\mathbb{E}[\|\boldsymbol{s}-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}]\\ &+\frac{2}{1-2\rho}\sup_n\mathbb{E}[\|\boldsymbol{P}\mathbb{E}[\boldsymbol{s}|\boldsymbol{x}^{(n)}]-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}] \end{aligned} \tag{3.16}$$

Proof. On one hand, from Equations (3.11) and (3.12)

$$\begin{aligned} &\mathbb{E}[\|\boldsymbol{P}\boldsymbol{\varepsilon}(\boldsymbol{x}^{(n)})\|_{\boldsymbol{W}}^2|\boldsymbol{y}]=\mathbb{E}[\|\boldsymbol{x}^{(n+1)}-\boldsymbol{P}\mathbb{E}[\boldsymbol{s}|\boldsymbol{x}^{(n)}]\|_{\boldsymbol{W}}^2|\boldsymbol{y}]\\ \geq&\frac{1}{2}\mathbb{E}[\|\boldsymbol{x}^{(n+1)}-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}]-\mathbb{E}[\|\boldsymbol{P}\mathbb{E}[\boldsymbol{s}|\boldsymbol{x}^{(n)}]-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}] \end{aligned} \tag{3.17}$$

On the other hand, by orthogonality principle

$$\mathbb{E}[\|\boldsymbol{x}^{(n)}-\boldsymbol{s}\|_{\boldsymbol{W}}^2|\boldsymbol{y}]=\mathbb{E}[\|\boldsymbol{x}^{(n)}-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}]+\mathbb{E}[\|\boldsymbol{s}-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}] \tag{3.18}$$

Combining Equations (3.14), (3.17) and (3.18), we have

$$\mathbb{E}[\|\boldsymbol{x}^{(n+1)}-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}]\leq 2\rho\mathbb{E}[\|\boldsymbol{x}^{(n)}-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}]+(1-2\rho)u \tag{3.19}$$

Create an auxiliary sequence

$$a^{(n)}=\mathbb{E}[\|\boldsymbol{x}^{(n)}-\boldsymbol{x}^*\|_{\boldsymbol{W}}^2|\boldsymbol{y}]-u \tag{3.20}$$

Then by Equation (3.19),

$$a^{(n+1)} \leq (2\rho)^n a^{(1)} \tag{3.21}$$

Taking $\limsup_{n\to\infty}$ on both sides of Equation (3.21) concludes the proof. □

If $u = 0$, then Equation (3.15) implies mean square convergence to the optimal beamformer output. In actuality, u is nonzero, but it tends to be very small. The first term of u measures the distance between the optimal beamformer output and the true speech. According to our empirical study, when the number of channels is sufficient, the optimal beamformer is able to recover the true speech very well, so the first term is small. The second term of u measures the distance between two posterior expectations $\boldsymbol{P}\mathbb{E}[\boldsymbol{s}|\boldsymbol{x}^{(n)}]$ and $\boldsymbol{P}\mathbb{E}[\boldsymbol{s}|\boldsymbol{y}]$. The former is conditional on single-channel noisy speech, and the latter on multiple-channel noisy speech. Considering that the speech sample space is highly structured, and that the noisy speech $\boldsymbol{x}^{(n)}$ is relatively clean already, both posterior expectations should be close to the true speech, and thereby close to each other. In a nutshell, with a small u, the DeepBeam prediction is highly accurate. Section 3.4.4 will verify the convergence behavior of DeepBeam empirically.

3.4 Experiments

This section first introduces how the enhancement network is configured and trained, and then presents the results of experiments on both simulated and real-world data.

3.4.1 Enhancement Network Configurations

The enhancement network hyperparameter configurations follow [7]. The network has four blocks of 10 dilated convolution layers. There are two post-processing layers. The hidden node dimension is 32, and the skip node dimension is 256. The clean speech is quantized into 256 levels via μ-law companding, and thus the output dimension is 256. The activation function in the dilated convolutional layers is the gated activation unit; that in

the post-processing layers is the ReLU function. The output activation is softmax.

The enhancement network is trained on simulated data *only*, which is generated in the same way as in [1]. The speech source, noise source and eight microphones are randomly placed into a randomly sized cubic room. The impulse response from each source to each microphone is generated using the image-source method [117, 118]. The noisy observations are generated according to Equation (3.1). The reverberation time is uniformly randomly drawn from [100, 300] ms. The energy ratio between the speech source and noise source, E_r, is uniformly randomly drawn from $[-5, 20]$ dB. The speech content is drawn from VCTK [119], which contains 109 speakers. The noise content contains 90 minutes of audio drawn from [38, 120, 121]. The total duration of the training audio is 8 hours. The enhancement network is trained using ADAM optimizer for 400,000 iterations.

3.4.2 Simulated Data Evaluation

The simulated data for evaluation is generated the same way as the training data, except for two differences. First, the source energy ratio, E_r, is set to four levels, -10 dB, 0 dB, 10 dB, and 20 dB. Second, both the speaker and noise can be either seen or unseen in the training set, leading to four different scenarios to test generalizability. It is worth highlighting that the unseen speaker utterances and unseen noise are both drawn from different corpora from training, TIMIT [122] and FreeSFX [123] respectively. Each utterance is 3 seconds in length. The total length of the dataset is 12 minutes.

DeepBeam is compared with GRAB [1], MVDR [15],[1] IVA [22] and the closest channel (CLOSEST), in terms of two criteria:

- **Signal-to-Noise Ratio (SNR)**: The energy ratio of processed clean speech over processed noise in dB.
- **Direct-to-Reverberant Ratio (DRR)**: the ratio of the energy of direct path speech in the processed output over that of its reverberation in dB. Direct path and reverberation are defined as clean dry speech convolved with the peak portion and tail portion of processed room impulse response. The peak portion is defined as ± 6 ms within the highest peak; the tail portion is

[1] Clean speech is given for voice activity detection.

Table 3.1: Simulated data evaluation results.

E_r =		-10	0	10	20
SNR (dB)	DeepBeam S1	18.5	22.0	26.5	28.4
	DeepBeam S2	17.1	20.3	25.9	27.4
	DeepBeam S3	15.3	19.5	24.1	27.6
	DeepBeam S4	14.1	19.0	23.1	28.5
	GRAB	2.48	12.5	21.6	25.4
	CLOSEST	-5.13	3.38	14.9	24.8
	MVDR	8.41	12.9	22.6	26.7
	IVA	10.3	13.3	16.8	19.2
DRR (dB)	DeepBeam S1	3.45	8.97	11.2	11.5
	DeepBeam S2	7.38	11.9	12.6	11.5
	DeepBeam S3	5.60	4.85	8.43	9.78
	DeepBeam S4	2.11	6.68	7.10	9.31
	GRAB	-0.83	1.70	3.63	3.68
	CLOSEST	8.56	7.32	7.67	8.44
	MVDR	-2.17	-3.47	-3.42	-4.13
	IVA	-8.92	-8.77	-8.81	-8.99

S1: seen speaker, seen noise S2: seen speaker, unseen noise

S3: unseen speaker, seen noise S4: unseen speaker, unseen noise

defined as ±6 ms beyond.

Table 3.1 shows the results. As expected, DeepBeam's performance drops from S1, where both noise and speaker are seen during training, to S4, where neither is seen. However, in terms of SNR, even DeepBeam S4 significantly outperforms MVDR, which is the benchmark in noise suppression. In terms of DRR, DeepBeam matches or surpasses CLOSEST except for -10 dB. GRAB performs worse than in [1], because each utterance is reduced from 10 seconds to 3 seconds, which is more realistic but challenging. In short, of "cleanness" and "dryness", most algorithms can only achieve one, but DeepBeam can achieve *both* with superior performance.

3.4.3 Real-World Data Evaluation

DeepBeam and the baselines are also evaluated on the real-world dataset introduced in [1], which consists of two utterances by two speakers mixed with five types of noise, all recorded in a real conference room using eight randomly positioned microphones. The source energy ratio is set such that

Table 3.2: Real-world Data Evaluation Results.

Noise Type		N1	N2	N3	N4	N5
SNR (dB)	DeepBeam	**20.1**	**20.0**	**16.9**	**19.6**	**18.7**
	GRAB	18.9	17.4	12.4	18.5	17.4
	CLOSEST	10.0	10.0	10.0	10.0	10.0
	MVDR	10.8	16.5	7.72	14.0	13.4
	IVA	11.7	9.74	6.83	12.4	15.9
MOS	DeepBeam	**3.83**	**3.72**	**3.63**	**4.09**	**4.20**
	GRAB	3.10	3.06	2.93	3.71	3.45
	CLOSEST	2.74	2.68	3.02	3.55	3.50
	MVDR	2.05	2.40	2.28	2.71	2.62
	IVA	1.73	2.03	1.75	1.78	2.08

N1: cell phone N2: CombBind machine N3:paper shuffle
N4: door slide N5: footsteps

the SNR for the closest microphone is 10 dB. The utterance in each scenario is around 1 minute long, so the total length of the dataset is 10 minutes.

Besides SNR, a subjective test similar to [1] is performed on Amazon Mechanical Turk. Each utterance is broken into six sentences. In each test unit, called HIT, a subject is presented with one sentence processed by the five algorithms, and asked to assign an MOS [124] to each of them. Each HIT is assigned to 10 subjects.

Table 3.2 shows the results. As can be seen, DeepBeam outperforms the other algorithms by a large margin. In particular, DeepBeam achieves > 4 MOS in some noise types. These results are very impressive because DeepBeam is only trained on simulated data. The real-world data differ significantly from the simulated data in terms of speakers, noise types and recording environment. Furthermore, some microphones are contaminated by strong electric noise, which is not accounted for in Equation (3.1). Still, DeepBeam manages to perform well. The neural network used to be vulnerable to unseen scenarios, but DeepBeam has now made it robust.

3.4.4 Empirical Convergence Analysis

In order to empirically test whether DeepBeam has a good convergence property, 10 sets of eight-channel simulated data are generated with the S1 setting and $E_r = 10$. To study different numbers of channels, in each sub-

test, K channels are randomly drawn from each set of data for DeepBeam prediction, and the resulting SNR convergence curves of the 10 sets are averaged. K runs from 3 to 8.

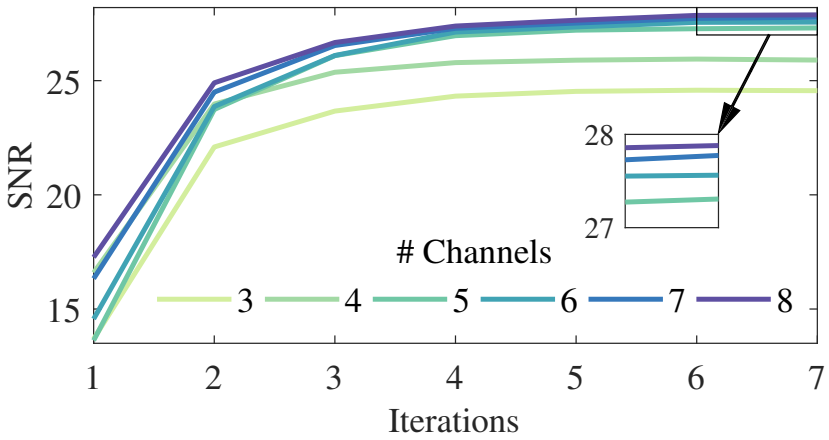


Figure 3.2: SNR convergence curves with different numbers of channels.

Figure 3.2 shows all the averaged convergence curves. As can be seen, DeepBeam converges well in all the sub-tests, which supports our convergence discussions in Section 3.3.3. Also, the more channels DeepBeam has, the higher convergence level it can reach, which shows that DeepBeam is able to accommodate different numbers of channels using only one monaural network. We also see that the marginal benefit of having one more channel diminishes.

3.5 Summary

We have proposed DeepBeam as a solution to multi-channel speech enhancement with ad-hoc sensors. DeepBeam combines the complementary beamforming and deep learning techniques, and has exhibited superior performance and generalizability in terms of noise suppression, reverberation cancellation and perceptual quality. DeepBeam is a step closer toward resolving the longstanding tradeoff of perceptual quality and generalizability in deep enhancement networks, and demonstrates the power of bridging the signal processing and deep learning areas.

CHAPTER 4

SPEECH ENHANCEMENT USING BAYESIAN WAVENET

4.1 Introduction

Deep learning has been widely used in speech enhancement tasks because its strong representation power is capable of characterizing complex noise distributions. For example, some works directly predict output spectrum using deep neural networks (DNN) or denoising auto-encoders [33, 36, 37, 38]. A series of works [110, 111] applied different deep learning architectures to predict ideal ratio masks. In addition, several works performed speech separation using various deep learning architectures [112, 113].

However, these approaches have two major limitations. First, these deep learning algorithms rarely incorporate an explicit prior model for clean speech or a Bayesian framework, which has been shown effective for speech enhancement [125]. While the variability of noise is hardly tractable, the clean speech signal is highly structured, and thus a prior speech model can regularize enhanced speech to become speech-like. Without the speech model, many deep learning algorithms are not generalizable to noise without highly similar characteristics.

On the other hand, existing Bayesian speech enhancement algorithms mostly model speech using simple probability distribution in order to have closed-form solutions. For example, a large body of such works assume HMM-GMM models [44, 45, 46, 47] or Laplacian models [48, 49, 22, 50]. Others make looser assumptions on kurtosis or negentropy of speech distribution [20, 23]. For these algorithms, building a more accurate model for speech becomes a bottleneck, which can potentially be opened by deep learning.

The second limitation regarding the existing deep learning based approach is that most deep learning algorithms operate on amplitude spectrum, such as short-time Fourier transform or cochleargram. The noisy phase spectrum

is directly applied to the enhanced speech without restoring the clean phase spectrum, which may suffer from phase distortion. Also, in some spectral restoration methods, the time domain signal is recovered by overlap-add, which is prone to artifacts and discontinuities. However, applying deep learning directly to speech waveform is difficult because the high sampling rate requires large temporal memory and receptive field size.

Fortunately, the recently announced WaveNet [7] has shown a strong capability in modeling raw audio waveforms. Its receptive field size is significantly boosted by stacking dilated convolution layers with exponentially increasing dilation rates. Experiments have shown that it is able to generate random babbles with high naturalness. Moreover, WaveNet is probabilistic, which naturally fits into the Bayesian framework.

Motivated by these observations, we propose a Bayesian speech enhancement algorithm using deep learning structures inspired by WaveNet, called the Bayesian WaveNet (BaWN). BaWN directly predicts the clean speech audio samples by estimating the prior distribution and the likelihood function of clean speech using WaveNet-like architectures, which are the two major components of the Bayesian network. It promotes a happy marriage between the Bayesian framework and the deep learning techniques: the former broadens the generalizability for the latter, and the latter improves the model accuracy for the former.

4.2 The Model Architecture

The problem is formulated within the Bayesian framework. Denote $X_{0:T-1}$ as the random process of the clean speech, which is quantized into Q levels, $q_{0:Q-1}$, via the μ-law encoding [126], so each X_t is a discrete variable. The subscript $0:T-1$ denotes a set with subscripts running from 0 through $T-1$. Denote $Y_{0:T-1}$ as the random process of the observed noisy signal. In this thesis, only additive noise is considered, but the framework is generalizable to other types of interferences. Our task is to infer the clean speech $\hat{x}_t$ given a set of noisy observations $Y_{0:T} = y_{0:T}$. For notational ease, probability mass functions will be abbreviated, e.g. $p(X_t = x_t|Y_t = y_t)$ as $p(x_t|y_t)$.

We apply a sub-optimal greedy inference scheme for $X_{0:T-1}$. Given inferred values of the past samples $\hat{x}_{0:t-1}$, the inferred value of the current sample,

$\hat{x}_t$, is defined as the posterior expectation

$$\hat{x}_t \triangleq \mathbb{E}\left[X_t | X_{t-\tau_1:t-1} = \hat{x}_{t-\tau_1:t-1}, Y_{t-\tau_2:t+\tau_2} = y_{t-\tau_2:t+\tau_2}\right] \tag{4.1}$$

Here we have made a Markov assumption that the probabilistic dependence of X_t upon variables in the distant past and far future is negligible, when the closer ones, $X_{t-\tau_1:t-1}$ and $Y_{t-\tau_2:t+\tau_2}$, are given. The terms τ_1 and τ_2 denote the range of dependence on $X_{0:T-1}$ and $Y_{0:T-1}$, respectively. Therefore, the following posterior distribution should be evaluated:

$$\begin{aligned} & p(X_t = x_t | X_{t-\tau_1:t-1} = \hat{x}_{t-\tau_1:t-1}, Y_{t-\tau_2:t+\tau_2} = y_{t-\tau_2:t+\tau_2}) \\ \triangleq & p(x_t | \hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2}) \\ \propto & p(x_t | \hat{x}_{t-\tau_1:t-1}) \cdot p(y_{t-\tau_2:t+\tau_2} | \hat{x}_{t-\tau_1:t-1}, x_t) \end{aligned} \tag{4.2}$$

where the $\triangleq$ sign denotes the abbreviation.

Define the likelihood function as

$$L(x_t; \hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2}) \triangleq p(y_{t-\tau_2:t+\tau_2} | \hat{x}_{t-\tau_1:t-1}, x_t) \tag{4.3}$$

Then Equation (4.2) can be rewritten into

$$\begin{aligned} & p(x_t | \hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2}) \\ = & \underbrace{p(x_t | \hat{x}_{t-\tau_1:t-1})}_{\text{prior model}} \cdot \underbrace{L(x_t; \hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2})}_{\text{likelihood model}} \end{aligned} \tag{4.4}$$

The BaWN architecture is based on Equation (4.4). As shown in Figure 4.1(a), it consists of two models. The first model is called the prior model, or the speech model, modeling the prior distribution of clean speech signals. For each time t, it takes $\hat{x}_{t-\tau_1:t-1}$ as input, and outputs a Q-dimensional vector of the log estimated pmf $\log \hat{p}(x_t | \hat{x}_{t-\tau_1:t-1})$ up to an unknown constant.

The second model is called the likelihood model, or the noise model, modeling the likelihood function. It takes as inputs $\hat{x}_{t-\tau_1:t-1}$ and $y_{t-\tau_2:t+\tau_2}$, and outputs a Q-dimensional vector of the estimated log likelihood function $\log \hat{L}(x_t; \hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2})$ up to an unknown constant.

The two outputs are added and then passed through a softmax nonlinearity. Notice that the exponential function in softmax turns addition into multiplication; the normalization step in softmax removes any unknown con-

stant. Therefore it can be easily shown, from Equation (4.4), that the output of the softmax nonlinearity is the $p(x_t|\hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2})$ of interest. Also, the output of the prior model, passing through a softmax nonlinearity alone, becomes the prior distribution $p(x_t|\hat{x}_{t-\tau_1:t-1})$.

The following two subsections introduce the two models respectively.

4.2.1 The Prior Model

The prior model replicates the architecture of WaveNet because it performs a similar task. As shown in Figure 4.1(b), the prior model consists of two modules. The first is the dilated convolution module, which contains a stack of B_1 blocks with L_1 layers for each. The l-th layer in the b-th block is a 1D causal convolution layer through time, with kernel size 2 and dilation rate 2^l. For each time t, it produces two vector outputs—a hidden output $z_t^{(b,l)}$, which is fed into the convolution layer above, and a skip output $s_t^{(b,l)}$, which is directly fed into the second module. The nonlinearity applied is a gated activation unit [127] with residual structure [128]. Formally,

$$f_t^{(b,l)} = \tanh\left(W_{f0}^{(b,l)} i_t^{(b,l)} + W_{f1}^{(b,l)} i_{t-2^l}^{(b,l)} + d_f^{(b,l)}\right) \tag{4.5a}$$

$$g_t^{(b,l)} = \sigma\left(W_{g0}^{(b,l)} i_t^{(b,l)} + W_{g1}^{(b,l)} i_{t-2^l}^{(b,l)} + d_g^{(b,l)}\right) \tag{4.5b}$$

$$r_t^{(b,l)} = f_t^{(b,l)} \odot g_t^{(b,l)} \tag{4.5c}$$

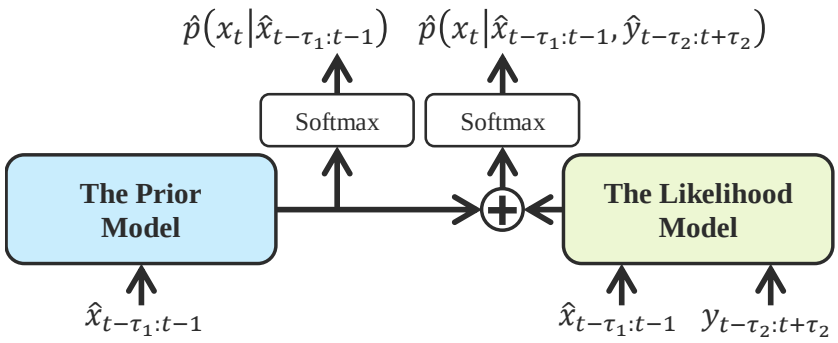
$$z_t^{(b,l)} = i_t^{(b,l)} + W_z^{(b,l)} r_t^{(b,l)} + d_z^{(b,l)} \tag{4.5d}$$

$$s_t^{(b,l)} = i_t^{(b,l)} + W_s^{(b,l)} r_t^{(b,l)} + d_s^{(b,l)} \tag{4.5e}$$

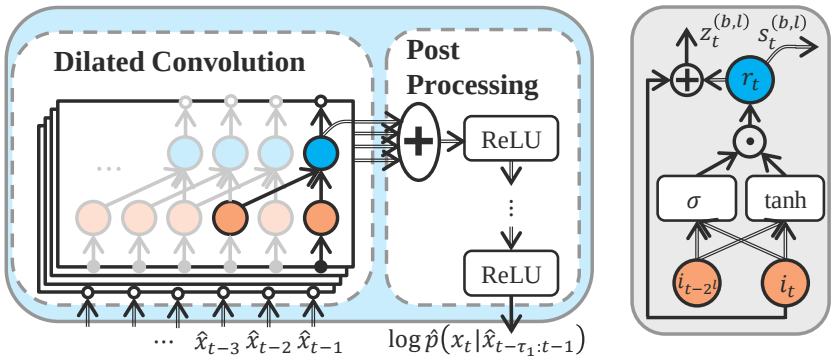
where $\sigma(\cdot)$ denotes the sigmoid function, $\odot$ denotes element-wise multiplication, and $i_t^{(b,l)}$ denotes the input to this layer,

$$i_t^{(b,l)} = \begin{cases} z_t^{(b,l-1)} & \text{if } l > 0 \\ z_t^{(b-1,L_1-1)} & \text{if } l = 0, b > 0 \\ W_i \hat{x}_t & \text{otherwise} \end{cases} \tag{4.6}$$

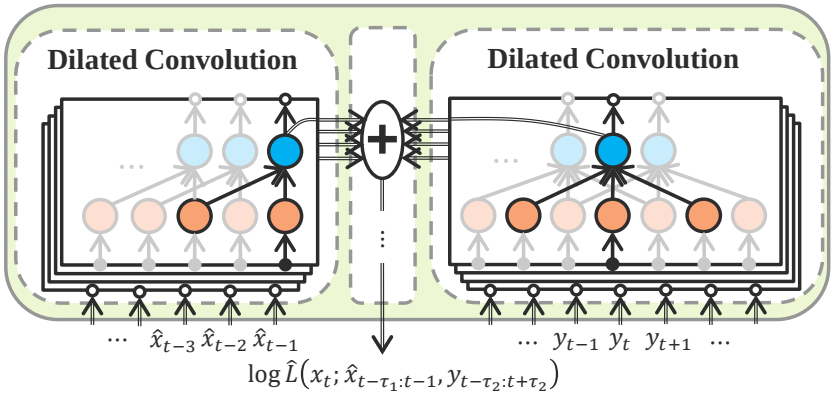
The second module is the post-processing module, which sums all the skip outputs of time t, $s_t^{(0:B_1-1,0:L_1-1)}$, and passes it to a stack of 1×1 convolution (fully connected within time t) layers with ReLU activation. The receptive



(a) The general model framework



(b) The prior model. The right plot gives a detailed view of a basic convolution unit in the left plot (Equation (4.5)).



(c) The likelihood model. The middle module is the post-processing module, whose structure is similar to that in (b).

Figure 4.1: The model architecture. Compound arrows denote that the node is multiplied by a weight matrix before sent to the next unit. Circled add and circled dot denote element-wise addition and multiplication respectively. The data path that generates the current output at time t is highlighted.

field size is shown as

$$\tau_1 = B_1 \left(2^{L_1} - 1\right)$$

4.2.2 The Likelihood Model

The likelihood model is more complex than the prior model. This is because (1) in addition to $\hat{x}_{t-\tau_1:t}$, which is the input to both models, the likelihood model also takes $y_{t-\tau_2:t+\tau_2}$ as input; (2) the prior model is causal, but the likelihood model is non-causal.

To address these complexities, we adapt the original WaveNet structure to that shown in Figure 4.1(c). The likelihood model also has a dilation convolution module and a post-processing module, but the dilation module now contains two parts. The first part deals with the input $\hat{x}_{t-\tau_1:t}$, and has the same structure as in Equations (4.5) and (4.6). The second part deals with the input $y_{t-\tau_2:t+\tau_2}$, and has almost the same structure, except for two differences. First, the number of blocks and layers within each block is changed to B_2 and L_2 respectively, to accommodate τ_2, which can be different from τ_1. Second, instead of a causal convolution with kernel size 2, this part imposes a non-causal convolution with kernel size 3 to account for future dependency. Formally, Equations (4.5a) and (4.5b) are adapted to

$$f_t^{(b,l)} = \tanh\left(W_{f0}^{(b,l)} i_t^{(b,l)} + W_{f1}^{(b,l)} i_{t-2^l}^{(b,l)} + W_{f-1}^{(b,l)} i_{t+2^l}^{(b,l)} + d_f^{(b,k)}\right) \tag{4.7a}$$

$$g_t^{(b,l)} = \sigma\left(W_{g0}^{(b,l)} i_t^{(b,l)} + W_{g1}^{(b,l)} i_{t-2^l}^{(b,l)} + W_{g-1}^{(b,l)} i_{t+2^l}^{(b,l)} + d_g^{(b,l)}\right) \tag{4.7b}$$

The post-processing module in the likelihood model is the same as that in the prior model, except that it sums all the skip outputs from both parts of the dilated convolution module.

4.3 Training the Model

Since the two models in BaWN have their own specific interpretations, the training scheme should be designed carefully to ensure that the models generate the correct outputs.

4.3.1 Training the Prior Model

If we replace the input $\hat{x}_{t-\tau_1:t-1}$ with the true clean samples, denoted as $x^*_{t-\tau_1:t-1}$, then the prior model can be trained on clean speech, following a similar paradigm as in WaveNet. Specifically, for each t, given the previous true clean speech, $x^*_{t-\tau_1:t-1}$ as input, the training scheme minimizes the cross entropy between the estimated prior distribution and the empirical distribution. Formally, the training scheme solves the following optimization problem:

$$\max \sum_{t=0}^{T-1} \sum_{i=0}^{Q-1} \mathbb{1}\left\{x_t^* = q_i\right\} \log \hat{p}(X_t = q_i | x_{t-\tau_1:t-1}) \tag{4.8}$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function, which equals 1 if the statement in its argument is true and 0 otherwise.

So far, we have implemented only the speaker-dependent enhancement task. The generalization to speaker-independent models will be one of our future directions.

4.3.2 Training the Likelihood Model

Once the prior model is trained, the likelihood model can be trained by combining both models to estimate the posterior distribution, as indicated by Equation (4.2). Ideally, we would like to solve

$$\max \sum_{t=0}^{T-1} \sum_{i=0}^{Q-1} \mathbb{1}\left\{x_t^* = q_i\right\} \log \hat{p}(X_t = q_i | \hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2}) \tag{4.9}$$

However, notice that the input of time t contains $\hat{x}_{t-\tau_1:t-1}$, which is a function of the previous time outputs, as shown in Equation (4.1). Therefore, Equation (4.9) introduces time recurrence, which causes gradient explosion in practice. An alternative is to replace $\hat{x}_{t-\tau_1:t-1}$ with the true value $x^*_{t-\tau_1:t-1}$ as in prior model training, but this approximation leads to insufficient training, because the model is given too much oracle information about the clean speech.

Our solution is to replace $\hat{x}_{t-\tau_1:t-1}$ with the inferred clean speech produced by the network trained in the *previous iteration*. Denote the previous inferred

value as $\hat{x}^{(\text{old})}_{t-\tau_1:t-1}$; then the problem in Equation (4.9) is reformulated as

$$\max \sum_{t=0}^{T-1} \sum_{i=0}^{Q-1} \mathbb{1}\left\{x_t^* = q_i\right\} \log \hat{p}(X_t = q_i | \hat{x}^{(\text{old})}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2}) \tag{4.10}$$

The previous inferred value $\hat{x}^{(\text{old})}_{t-\tau_1:t-1}$ can be implemented efficiently using the method in [129].

It should be emphasized that while optimizing for Equation (4.10), the weights of the prior model should be held fixed to prevent deviation from modeling the prior distribution.

4.4 Experiments

This section presents experiments that test the performance of the proposed BaWN model. In particular, we will investigate how the prior model improves the generalizability of BaWN to deal with completely unseen and different noises. The ideal ratio mask (DNN-IRM) based model [110] was also implemented as a baseline.

4.4.1 Configurations

The three dilated convolutional networks of the WaveNet enhancement model all have four blocks of 10 layers, which makes a receptive field size of approximately two to three phones. For each layer, the hidden output has 32 channels and the skip output has 1024 channels. The post-processing modules in both the prior and the likelihood models contain two fully connected layers, each with 1024 hidden nodes. The clean speech is quantized into 256 levels, so the output dimension is 256.

The training dataset consists of a clean training set (for the prior model) and a noisy training set. The clean training set contains a total of 9700 utterances (19 hours) from audio books played by a female speaker [130]. The noisy training set was created by mixing the 9700 clean utterances randomly with 100 environment noises from [38, 120, 121], including train, airport, restaurant and ring tones. The SNR of the noisy training set is set to two levels: 0 dB and -5 dB.

There are two test sets, respectively containing 20 and 100 clean utterances of the same speaker randomly selected from another audio book. For the first test set, called the unseen noise test set, 100 noises were selected from a completely different noise dataset [123] in order to test the generalizablity of BaWN, where the types of noise and recording configurations completely differ from that of the training noise dataset. For investigation purpose, the second test set, called the seen noise test set, contains 20 noises drawn from the training noise dataset.

The input training utterances were first segmented into fixed-length tokens. Then, each clean token was quantized using 256-level μ-law companding and padded with 4092 historical samples based on the receptive field size of the our model. The noisy utterances were not quantized because the model does not make predictions of noisy speech. Each noisy token was padded not only with historical samples but also with the same number of future samples. The target output was a 256-dimensional one-hot vector indicating the quantization level of the desired output sample.

The prior model was trained on all 9700 (19 hours) clean utterances. Due to significantly increased model complexity and the EM-like training procedures, the likelihood model was trained on only 500 (1 hour) utterances from the noisy training set. Though the small amount of training data may lead to an insufficiently trained likelihood model, it actually provides a good opportunity to verify the power of the prior model and test the generalizablity of BaWN. For fair comparison, the DNN-IRM baseline was trained on the complete noisy training set. During testing, each predicted clean sample was fed back as the clean input sample to predict the next clean sample.

The DIRM baseline was constructed according to [111] and trained on the same 9700 noisy utterances. The 64-channel cochleargrams were extracted from the noisy utterances as the input features. The targets were the ideal-ratio-masks (IRMs) at the corresponding frame and channel. The IRM of the current frame is predicted using 23 neighboring frames centered at the current frame. During testing, the IRMs were predicted and applied to the corresponding noisy utterances to recover clean utterances.

Table 4.1: Average SNR, SAR, SDR, STOI of the enhanced utterance using DNN-IRM and BaWN. The first three metrics are measured in decibels (dB), and the STOI is measured in percentage (%). Case indicates the input SNR of the training and testing dataset. Noise indicates whether the noise type is covered by the training set. BaWN stands for Bayesian WaveNet. DIRM stands for DNN-IRM.

Case	Noise	Model	SNR	SAR	SDR	STOI
0dB	seen	BaWN	22.2	8.53	8.83	85.7
		DIRM	15.6	10.3	12.3	86.4
	unseen	BaWN	22.1	8.37	8.75	84.3
		DIRM	11.9	8.58	12.7	84.8
-5dB	seen	BaWN	21.6	7.15	7.37	81.7
		DIRM	12.2	6.45	8.53	79.0
	unseen	BaWN	20.3	6.65	6.92	80.7
		DIRM	9.20	5.25	8.24	76.6

4.4.2 Objective Evaluation Results

The performance was measured by the average of SNR, signal-to-artifacts ratio (SAR), signal-to-distortion ratio (SDR), and short-time objective intelligibility (STOI) of the predicted clean utterances. The first three metrics were computed using the BSS-EVAL toolbox [131].

As seen in Table 4.1, the BaWN model outperforms the DNN-IRM model in terms of much higher SNRs. The performance advantage is more significant in the -5 dB case, where BaWN takes the lead in SAR and STOI as well. Also, our model generalizes better to the completely different unseen noise, as the performance drop is smaller. This is remarkable considering that the likelihood model was trained on only one hour of noisy speech and the parameters of the model were not tuned. The prior model has enough knowledge about the distribution of clean speech samples and tends to make non-speech distributions less likely under unseen noise and low SNR, which helps to make better predictions even if the likelihood model is weak. BaWN achieves slightly lower SDR and, in the 0 dB case, SAR, because the sequential inference would occasionally generate impulse noise. Yet this does not weaken our argument for BaWN, considering the inherent negative correlation between the SNR and SAR/SDR, and the huge performance gain in SNR.

4.5 Entropy Analysis

The effectiveness of the prior model under the Bayesian framework can be further visualized and analyzed by computing the entropies of the estimated prior and posterior distribution of each sample. Specifically

$$\begin{aligned} H_t^{(\mathrm{pr})} &= -\sum_{i=0}^{Q} \hat{p}(X_t = q_i | \hat{x}_{t-\tau_1:t-1}) \\ &\qquad \cdot \log_2 \hat{p}(X_t = q_i | \hat{x}_{t-\tau_1:t-1}) \\ H_t^{(\mathrm{post})} &= -\sum_{i=0}^{Q} \hat{p}(X_t = q_i | \hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2}) \\ &\qquad \cdot \log_2 \hat{p}(X_t = q_i | \hat{x}_{t-\tau_1:t-1}, y_{t-\tau_2:t+\tau_2}) \end{aligned} \tag{4.11}$$

Since the prediction of a sample is more uncertain if the entropy of the corresponding distribution is high, we can conclude that the prior model plays a more important role than the likelihood model at time t if $H_t^{(\mathrm{pr})} < H_t^{(\mathrm{post})}$. Hence we define a prior effectiveness function

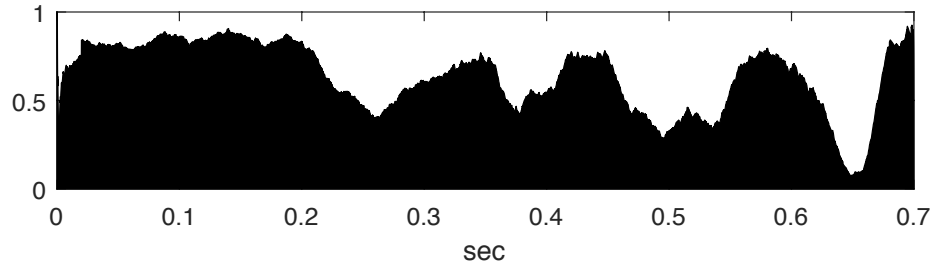
$$e_t = \mathbb{1}\left(H_t^{(\mathrm{pr})} < H_t^{(\mathrm{post})}\right) \tag{4.12}$$

to depict the real-time effectiveness of the prior model. e_t is further smoothed by a 20 ms moving average filter.

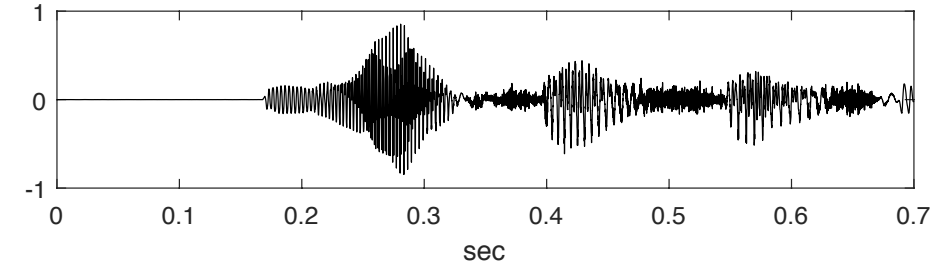
In Figure 4.2a, using the entropies of the predicted distributions for each sample from the prior model and the likelihood model respectively, a 0-1 vector indicating whether the prior model is more certain than the likelihood model about each predicted sample was computed and then smoothed by a rectangular window of 20 ms. For example, a level of 0.8 at some sample point indicates that the prior model is more certain than the likelihood model 80% of the time within 20 ms around this sample point.

Figure 4.2 shows the smoothed e_t of a test speech segment (a), as well as its corresponding clean speech (b) and noise (c) waveforms. There are two important observations. First, the prior model is more effective when the SNR is low, as can be seen from the segment before 0.25 s. This is because when the SNR is high enough, the likelihood model can simply pass noisy observation through, which does not rely much on the prior model.

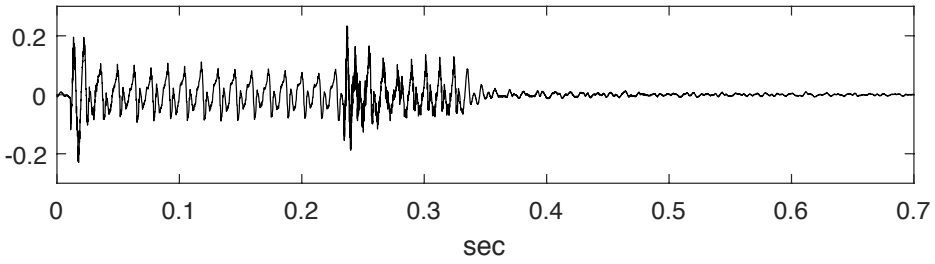
Second, the prior model is more effective after the onset of vowels or voiced



(a) Effectiveness of the prior model, c_t



(b) Clean utterance waveform



(c) Noise waveform

Figure 4.2: The prior effectiveness function (Equation (4.12)) of a speech segment, smoothed by a 20 ms moving average filter, with its corresponding utterance and noise.

consonants. Accordingly, the likelihood model is more effective during unvoiced consonants or at the onset of speech activities, as can be seen from dips in the effectiveness function at around 0.4 s, 0.5 s and 0.65 s. This is because the voiced speech is well structured, so the prior model knows what comes next once it recognizes the phone. On the other hand, the prior model is less certain about the unvoiced phones because they are stochastic and can be easily confused with noise.

4.6 Summary

We proposed a WaveNet enhancement model that directly operates on speech waveforms and exploited its generalizability to completely unseen noise. The results showed that our proposed model is able to produce clean speech and outperforms the DNN-IRM model under small-sized training data in terms of generalizability owing to the effectiveness of the prior model.

CHAPTER 5

ZERO-SHOT VOICE STYLE TRANSFER WITH ONLY AUTOENCODER LOSS

5.1 Introduction

The idea of speaking in someone else's voice never fails to be a fascinating element in action and fiction movies, and it also finds its way to many practical applications, *e.g.* privacy and identity protection, creative industry *etc.* In the speech research community, this task is referred to as the voice conversion problem, which involves modifying a given speech from a source speaker to match its vocal qualities with a target speaker.

Despite the continuing research efforts in voice conversion, three problems remain under-explored. First, most voice conversion systems assume the availability of parallel training data, *i.e.* speech pairs where the two speakers utter the same sentences. Only a few can be trained on non-parallel data. Second, among the few existing algorithms that work on non-parallel data, even fewer can work for many-to-many conversion, *i.e.* converting from multiple source speakers to multiple target speakers. Last but not least, no voice conversion systems are able to perform zero-shot conversion, *i.e.* conversion to the voice of an unseen speaker by looking at only a few of his/her utterances.

With the recent advances in deep style transfer, the traditional voice conversion problem is being recast as a style transfer problem, where the vocal qualities can be regarded as styles, and speakers as domains. There are many style transfer algorithms that do not require parallel data, and are applicable to multiple domains, so they are readily available as new solutions to voice conversion. In particular, generative adversarial network (GAN) [93] and conditional variational autoencoder (CVAE) [132, 133], are gaining popularity in voice conversion.

However, neither of GAN and CVAE is perfect. GAN comes with a nice

theoretical justification that the generated data would match the distribution of the true data, and has achieved state-of-the-art results, particularly in computer vision. However, it is widely acknowledged that GAN is very hard to train, and its convergence property is fragile. Also, although there is an increasing number of works that introduce GAN to speech generation [104, 134, 135], there is no strong evidence that the generated speech ***sounds*** real. Speech that is able to fool the discriminators has yet to fool human ears. On the other hand, CVAE is easier to train. All it needs to do is to perform self-reconstruction and maximize a variational lower bound of the output probability. The intuition is to infer a hypothetical style-independent hidden variable, which is then combined with the new style information to generate the style-transferred output. However, CVAE alone does not guarantee distribution matching, and often suffers from over-smoothing of the conversion output [58].

Due to the lack of a suitable style transfer algorithm, existing voice conversion systems have yet to produce satisfactory results, which naturally leads to the following question. Is there a style transfer algorithm that is also proven to match the distribution as GAN is, and that trains as easily as CVAE, and that works better for speech?

Motivated by this question, we propose a new style transfer scheme, which involves only a ***vanilla*** autoencoder with a carefully designed bottleneck. Similar to CVAE, the proposed scheme only needs to be trained on the self-reconstruction loss, but it has a distribution matching property similar to GAN's. This is because the correctly designed bottleneck will learn to remove the style information from the source and get the style-independent code, which is the goal of CVAE, but which the training scheme of CVAE is unable to guarantee.

Based on this scheme, we propose AutoVC, a many-to-many voice style transfer algorithm without parallel data. AutoVC follows the autoencoder framework and is trained only on autoencoder loss, but it introduces carefully tuned dimension reduction and temporal downsampling to constrain the information flow. As we will show, this simple scheme leads to a significant performance gain. AutoVC achieves superior performance on traditional many-to-many conversion task, where all the speakers are seen in the training set. Also, equipped the speaker embedding trained for speaker verification [136, 137], AutoVC is among the first to perform zero-shot voice conversion

with decent performance. Considering the quality of the results and the simplicity of its training scheme, AutoVC opens a new path toward a simpler and better voice conversion and general style transfer system.

5.2 Related Work

There are several works that perform non-parallel many-to-many voice conversion using VAE and its combination with adversarial training. VAE-VC [57] is a simple voice conversion system using VAE. Afterward, many research studies focus on removing the style information from the VAE code. VAW-GAN [138] introduces a GAN on the VAE output. CDVAE-VC [139] introduces two VAEs on two spectral features and forced the latent codes of the two features to contain similar information. ACVAE-VC [63] introduces an auxiliary classifier on the output to encourage the conversion results to be correctly classified as the target speaker's utterances. Chou et al. [140] introduce a classifier on the code and a GAN on the output. Similarly, StarGAN [141] and CycleGAN [102], which consist of encoder-decoder architectures with GAN, are applied to voice conversion [58, 142]. As another track of effort, phonetic posteriorgrams are introduced to assist the learning of the latent code [143, 60]. Apart from VAE, GAN alone is also applied to voice conversion [144]. However, the conversion quality of these algorithms is still limited.

It is worth mentioning that Atalla et al. [145] conduct preliminary research of style transfer using autoencoder only, and Chou et al. [140] briefly study its ability to disentangle speaker information in voice conversion. However, neither has shown promising results nor unveiled its distribution-matching property by properly designing the bottleneck.

5.3 Style Transfer Autoencoder

In this section, we will discuss how autoencoder can match the data distribution as GAN does. Although our intended application is voice conversion, the discussion in this section is applicable to other style transfer applications as well. As general mathematical notations, uppercase letters, *e.g.* X, de-

note random variables/vectors; lowercase letters, *e.g.* x, denote deterministic values or instances of random variables; $X(1:T)$ denotes a random process, with $(1:T)$ denoting a collection of time indices running from 1 to T. For notational ease, sometimes the time indices are omitted to represent the collection of the random process at all times. $p_X(\cdot|Y)$ denotes the probability mass function (PMF) or probability density function (PDF) of X conditional on Y; $p_X(\cdot|Y=y)$, or sometimes $p_X(\cdot|y)$ without causing confusions, denotes the PMF/PDF of X conditional on Y taking a specific value y.

5.3.1 Problem Formulation

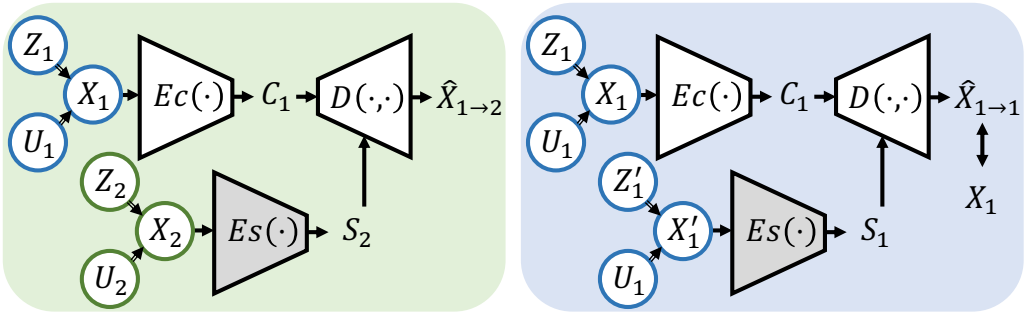
Assume that speech is generated by the following stochastic process. First, a speaker identity U is a random variable drawn from the speaker population $p_U(\cdot)$. Then, a content vector $Z = Z(1:T)$ is a random process drawn from the joint content distribution $p_Z(\cdot)$. Here content refers to the phonetic and prosodic information. Finally, given the speaker identity and content, the speech segment $X = X(1:T)$ is a random process randomly sampled from the speech distribution, *i.e.* $p_X(\cdot|U,Z)$, which characterizes the distribution of the speaker U's speech uttering the content Z. $X(t)$ can represent a sample of speech waveform, or a frame of speech spectrogram. In this chapter, we will be working on speech spectrogram.

Now, assume two sets of variables, (U_1, Z_1, X_1) and (U_2, Z_2, X_2), are independent and identically distributed (i.i.d.) random samples generated from this process. (U_1, Z_1, X_1) belong to the *source speaker* and (U_2, Z_2, X_2) belong to the *target speaker*. Our goal is to design a speech converter that produces the conversion output, $\hat{X}_{1\to 2}$, which preserves the content in X_1, but matches the speaker characteristics of speaker U_2. Formally, an ideal speech converter should have the following desirable property:

$$p_{\hat{X}_{1\to 2}}(\cdot|U_2 = u_2, Z_1 = z_1) = p_X(\cdot|U = u_2, Z = z_1) \tag{5.1}$$

Equation (5.1) means that given the target speaker's identity $U_2 = u_2$ and the content in the source speech $Z_1 = z_1$, the conversion speech should sound as if the target speaker u_2 were uttering z_1.

When U_1 and U_2 are both seen in the training set, the problem is a standard multi-speaker conversion problem, which has been addressed by some existing



(a) Conversion (b) Training

Figure 5.1: The style transfer autoencoder framework. The ovals denote the probabilistic graphical model of the speech generation process. The gray boxes denote pre-trained modules. (a) During conversion, the source speech is fed to the content encoder. An utterance of the target speaker is fed to the speaker encoder. The decoder produces the conversion results. (b) During training, the source speech is fed to the content encoder. Another utterance of the same *source* speaker is fed to the speaker encoder. The content encoder and the decoder minimize the self-reconstruction error.

works. When U_1 or U_2 is not included in the training set, the problem becomes the more challenging zero-shot voice conversion problem, which is also a target task of the proposed AutoVC. This problem formulation can be extended to a general style transfer setting, where U_1 and U_2 can represent two domains and X_1 and X_2 can represent samples from their respective domains.

5.3.2 The Autoencoder Framework

AutoVC solves the voice conversion problem with a very simple autoencoder framework, as shown in Figure 5.1. The framework consists of three modules, a content encoder $E_c(\cdot)$ that produces a content embedding from speech, a speaker encoder $E_s(\cdot)$ that produces a speaker embedding from speech, and a decoder $D(\cdot,\cdot)$ that produces speech from content and speaker embedding. The inputs to these modules are different for conversion and training.

Conversion: As shown in Figure 5.1(a), during the actual conversion, the source speech X_1 is fed into the content encoder to have content information extracted. The target speech is fed into the speaker encoder to provide target speaker information. The decoder produces the converted speech based on

the content information in the source speech and the speaker information in the target speech.

$$C_1 = E_c(X_1), \quad S_2 = E_s(X_2), \quad \hat{X}_{1\to 2} = D(C_1, S_2) \tag{5.2}$$

Here C_1 and $\hat{X}_{1\to 2}$ are both random processes. S_2 is simply a random vector.

Training: Throughout the chapter, we will assume the speaker encoder is already pre-trained to extract some form of speaker dependent embedding, so by training we refer to the training of the content encoder and the decoder. As shown in Figure 5.1(b), since we do not assume the availability of parallel data, only self-reconstruction is needed for training. More specifically, the input to the content encoder is still X_1, but the input to the style encoder becomes another speech drawn from the same speaker U_1 (uttering a different content Z_1'), denoted as X_1'. Then for each input speech X_1, AUTOVC learns to reconstruct itself:

$$C_1 = E_c(X_1), \quad S_1 = E_s(X_1'), \quad \hat{X}_{1\to 1} = D(C_1, S_1) \tag{5.3}$$

The loss function to minimize is simply the weighted combination of the self-reconstruction error and the content code reconstruction error, *i.e.*

$$\min_{E_c(\cdot), D(\cdot,\cdot)} L = L_{\text{recon}} + \lambda L_{\text{content}} \tag{5.4}$$

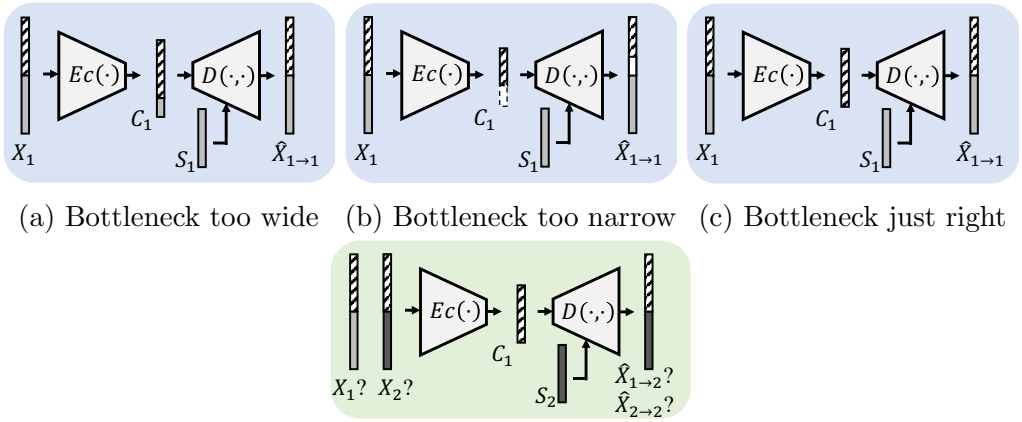
where

$$\begin{aligned} L_{\text{recon}} &= \mathbb{E}[\|\hat{X}_{1\to 1} - X_1\|_2^2] \\ L_{\text{content}} &= \mathbb{E}[\|E_c(\hat{X}_{1\to 1}) - C_1\|_1] \end{aligned} \tag{5.5}$$

As it turns out, this simple training scheme is sufficient to produce the ideal distribution-matching voice conversion, as will be shown in the next section.

5.3.3 Why does it work?

We will show this autoencoder-based training scheme is able to achieve ideal voice conversion (Equation (5.1)). The secret recipe is to have a proper information bottleneck. The basic idea is that the bottleneck dimension of the content encoder needs to be set such that it is just enough to code the speaker independent information.



(a) Bottleneck too wide (b) Bottleneck too narrow (c) Bottleneck just right

(d) Conversion

Figure 5.2: An intuitive explanation of how AutoVC works. The target speaker is the same as the source speaker during training ((a)-(c)), and different during the actual conversion ((d)). Each speech segment contains two types of information: the speaker information (solid) and content information (striped). (a) When the bottleneck is too wide, the content embedding will contain some source speaker information. (b) When the bottleneck is too narrow, the content information is lost, which leads to imperfect reconstruction. (c) When the bottleneck is just right, perfect reconstruction is achievable, *and* the content embedding contains no source speaker information. (d) During the actual conversion, the output should contain no information about the source speaker, so the conversion quality should be as high as if it were doing self-reconstruction.

As shown in Figure 5.2, speech contains two types of information: the speaker information (shown as solid color) and the speaker-independent information (shown as striped), which we will refer to as the content information.[1] Suppose the bottleneck is very wide, as wide as the input speech X_1. The most convenient way to do self-reconstruction is to copy X_1 as is to the content embedding C_1, and this will guarantee a perfect reconstruction. However as the dimension of C_1 decreases, C_1 is forced to lose some information. Since the autoencoder attempts to achieve perfect reconstruction, it will choose to lose speaker information because the speaker information is already supplied in S_1. In this case, perfect reconstruction is still possible, but the C_1 may contain some speaker information, as shown in Figure 5.2(a).

On the other hand, if the bottleneck is very narrow, then the content

[1]The speaker-independent information includes but is not limited to the content information in Z, but for convenience, we will refer to the speaker-independent information as content information.

encoder will be forced to lose so much information that not only the speaker information but also the content information is lost. In this case, the perfect reconstruction is impossible, as shown in Figure 5.2(b).

Therefore, as shown in Figure 5.2(c), when the dimension of C_1 is chosen such that the dimension reduction is just enough to get rid of all the speaker information but no content information is harmed, we have reached our desirable condition, under which two important properties hold:

1. Perfect reconstruction is achieved.

2. The content embedding C_1 does not contain any information of the source speaker U_1, which we refer to as *speaker disentanglement*.

We will now show by contradiction how these two properties imply an ideal conversion. Suppose when AutoVC is performing an actual conversion (source and target speakers are different), the quality is low, or does not sound like the target speaker at all. By property 1, we know that the reconstruction (source and target speakers are the same) quality is high. However, according to Equation (5.2), the output speech $\hat{X}_{1\to 2}$ can only access C_1 and S_2, both of which do not contain any information of the source speaker U_1. In other words, from the conversion output, one can never tell if it is produced by self-reconstruction or conversion, as shown in Figure 5.2(d). If the conversion quality is low, but the reconstruction quality is high, one will be able to distinguish between conversion and reconstruction above chance, which leads to a contradiction.

5.4 AutoVC Architecture

Figure 5.3 shows that AutoVC consists of three major modules: a speaker encoder, a content encoder, and a decoder. AutoVC works on the speech mel-spectrogram of size N-by-T, where N is the number of mel-frequency bins and T is the number of time steps (frames). A spectrogram inverter is introduced to convert the output mel-spectrogram back to the waveform, which will also be detailed in this section.

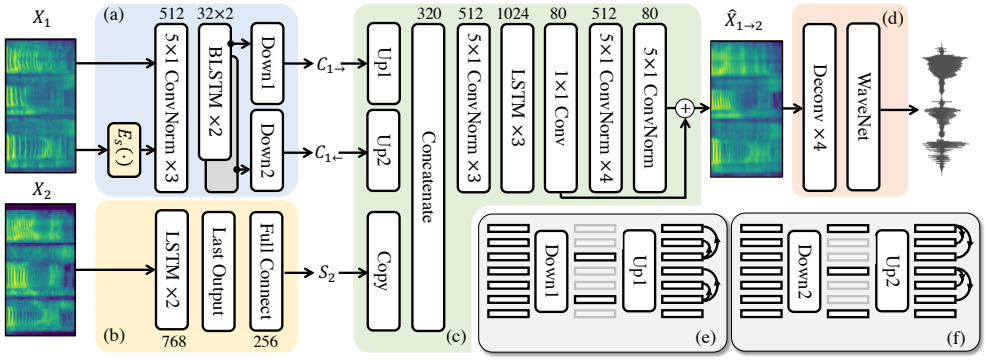


Figure 5.3: AutoVC architecture. The number above each block represents the cell/output dimension of the structure. ConvNorm denotes convolution followed by batch normalization. BLSTM denotes bi-directional LSTM, whose white block denotes forward direction, and gray block denotes backward direction. (a) The content encoder. The $E_s(\cdot)$ module is of the same architecture as in (b). (b) The style encoder. (c) The decoder. (d) The spectrogram inverter. (e) and (f) demonstrate the downsampling and upsampling of the forward and backward outputs of the bi-directional LSTM, using an up/downsampling factor of three as an example. The real up/downsampling factor is 32. The lightened feature denotes that they are removed; the arrows denote copying the feature at the arrow origin to the destination.

5.4.1 The Speaker Encoder

The goal of the speaker encoder is to produce the same embedding for different utterances of the same speaker, and different embeddings for different speakers. For conventional many-to-many voice conversion, the one-hot encoding of speaker identities suffices. However, in order to perform zero-shot conversion, we need to apply an embedding that is generalizable to unseen speakers. Therefore, inspired by Ye et al. [146], we follow the design in [137]. As shown in Figure (5.3)(b), the speaker encoder consists of a stack of two LSTM layers with cell size 768. Only the output of the last time step is selected and projected down to dimension 256 with a fully connected layer. The resulting speaker embedding is a 256-by-1 vector. The speaker encoder is pre-trained on the GE2E loss [137] (the softmax loss version). The GE2E loss attempts to maximize the embedding similarity among different utterances of the same speaker, and minimize the similarity among different speakers.

In our implementation, the speaker encoder is pre-trained on the combi-

nation of VoxCeleb1 [147] and Librispeech [148] corpora, where there are a total of 3549 speakers.

5.4.2 The Content Encoder

As shown in Figure 5.3(a), the input to the content encoder is the mel-spectrogram of X_1 concatenated with the speaker embedding, $E_s(X_1)$, at each time step. The concatenated features are fed into three 5×1 convolutional layers, each followed by batch normalization and ReLU activation. The number of channels is 512. The output then passes to a stack of two bidirectional LSTM layers. Both the forward and backward cell dimensions are 32.

As a key step of constructing the information bottleneck, both the forward and backward outputs of the bidirectional LSTM are downsampled by 32. The downsampling is performed differently for the forward and backward paths. For the forward output, the time steps $\{0, 32, 64, \cdots\}$ are kept; for the backward output, the time steps $\{31, 63, 95, \cdots\}$ are kept. Figures 5.3(e) and (f) also demonstrate how the downsampling is performed (for the ease of demonstration, the downsampling factor is set to three). The resulting content embedding is a set of two 32-by-$T/32$ matrices, which we will denote as $C_{1\rightarrow}$ and $C_{1\leftarrow}$ respectively. The downsampling can be regarded as dimension reduction along the temporal axis, which, together with the dimension reduction along the channel axis, constructs the information bottleneck.

5.4.3 The Decoder

The architecture of the decoder is inspired by [92]; and it is shown in Figure 5.3(c). First, the content and speaker embeddings are both upsampled by copying to restore to the original temporal resolution. Formally, we denote the upsampled features as $U_{\rightarrow}$ and $U_{\leftarrow}$ respectively. Then

$$\begin{aligned} U_{\rightarrow}(:, t) &= C_{1\rightarrow}(:, \lfloor t/32 \rfloor) \\ U_{\leftarrow}(:, t) &= C_{1\leftarrow}(:, \lfloor t/32 \rfloor) \end{aligned} \tag{5.6}$$

where $(:, t)$ denotes indexing the t-th column. Figures 5.3(e) and (f) also demonstrate the copying. The underlying intuition is that each embedding

at each time step should contain both past and future information. For the speaker embedding, simply copy the vector T times.

Then, the upsampled embeddings are concatenated and fed into three 5×1 convolutional layers with 512 channels, each followed by batch normalization and ReLU activation function, and then three LSTM layers with cell dimension 1024. The outputs of the LSTM layer are projected to dimension 80 with a 1×1 convolutional layer. This projection output is the initial estimate of the converted speech, denoted as $\tilde{X}_{1\to 2}$.

In order to construct the fine details of the spectrogram better on top of the initial estimate, we introduce a post-network after the initial estimate, as introduced in [92]. The post-network consists of five 5×1 convolutional layers, where batch normalization and hyperbolic tangent are applied to the first four layers. The channel dimension for the first four layers is 512, and goes down to 80 in the final layer. We will refer to the output of the post-network as the residual signal, denoted as $R_{1\to 2}$. The final conversion result is produced by adding the residual to the initial estimate, *i.e.*

$$\hat{X}_{1\to 2} = \tilde{X}_{1\to 2} + R_{1\to 2} \tag{5.7}$$

During training, reconstruction loss is applied to both the initial and final reconstruction results. Formally, in addition to the loss specified in Equation (5.4), we add an initial reconstruction loss defined as

$$L_{\text{recon0}} = \mathbb{E}[\|\tilde{X}_{1\to 1} - X_1\|_2^2] \tag{5.8}$$

where $\tilde{X}_{1\to 1}$ is the reciprocal of $\tilde{X}_{1\to 2}$ in the reconstruction case, *i.e.* when $U_2 = U_1$. The total loss becomes

$$\min_{E_c(\cdot), D(\cdot,\cdot)} L = L_{\text{recon}} + \mu L_{\text{recon0}} + \lambda L_{\text{content}} \tag{5.9}$$

5.4.4 The Spectrogram Inverter

We apply the WaveNet vocoder as introduced in [149], which consists of four deconvolution layers. In our implementation, the frame rate of the mel-spetrogram is 62.5 Hz and the sampling rate of speech waveform is 16 kHz. So the deconvolution layers will upsample the spectrogram to match the

sampling rate of the speech waveform. Then, a standard 40-layer WaveNet conditioning upon the upsampled spectrogram is applied to generate the speech waveform. We pre-trained the WaveNet vocoder using the method described in [92] on the VCTK corpus.

5.5 Experiments

In this section, we will evaluate AutoVC on many-to-many voice conversion tasks, and empirically validate the assumptions of the AutoVC framework.

5.5.1 Configurations

The evaluation is performed on the VCTK corpus [150], which contains 44 hours of utterances from 109 speakers. Each speaker reads a different set of sentences, except for the rainbow passage[2] and the elicitation paragraph. So the conversion setting is non-parallel. Depending on the conversion tasks, different subsets of speakers were selected. The data of each speaker is then partitioned into training and test sets by 9:1. AutoVC is trained with a batch size of two for 100k steps, using the ADAM optimizer. The speaker embedding is generated by feeding 10 two-second utterances of the same speaker to the speaker encoder and averaging the resulting embeddings. The weights in Equation (5.9) are set to $\lambda = 1$, $\mu = 1$.

We performed two subjective tests on Amazon Mechanical Turk (MTurk).[3] In the first test, called the mean opinion score (MOS) test, the subjects are presented with converted utterances. For each utterance, the subjects are asked to assign a score of 1-5 on the naturalness on the converted speech. In the second test, called the similarity test, the subjects are presented with pairs of utterances. In each pair, there is one converted utterance, and one utterance from the target speaker uttering the same sentence. For each pair, the subjects are asked to assign a score of 1-5 on the voice similarity. We follow the design in [151] to cue the subjects to judge if the speakers are the same, and how confident they are with their judgment. Thus the similarity score of 5 corresponds to the same speaker with high confidence,

[2]http://web.ku.edu/ idea/readings/rainbow.htm
[3]https://www.mturk.com/

and 1 corresponds to different speakers with high confidence. The subjects are explicitly asked to focus on the voice rather than intonation and accent.

5.5.2 Traditional Many-to-Many Conversion

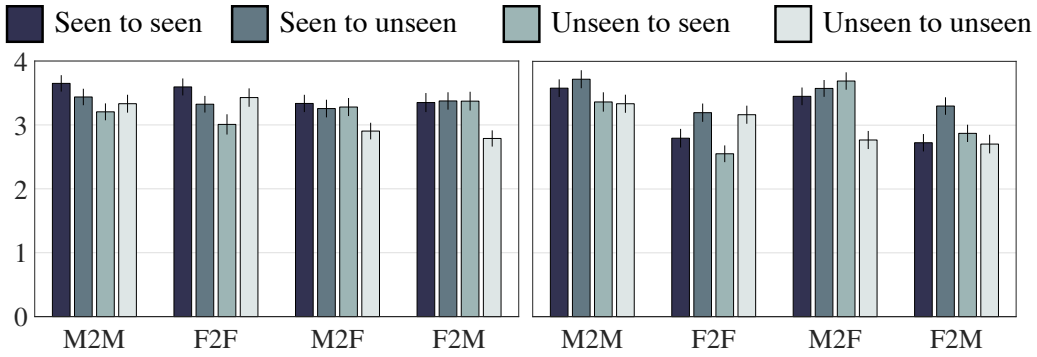
Traditional many-to-many conversion task performs conversion only on speakers seen in the training corpus. Two baselines are compared with AutoVC, which we name StarGAN-VC [58] and Chou et al. [140]. Both baselines are current state-of-the-arts in non-parallel many-to-many voice conversion. For Chou et al., we use the original implementation[4] and its pre-trained model, which is trained on 20 speakers in the VCTK corpus. For fair comparison, the other models are trained on the same 20 speakers. Note that the training/test sets are partitioned differently from the Chou et al.'s pre-trained model, so we are giving the Chou et al. baseline an unfair advantage of seeing part of the test utterance during training. We use the open-source implementation for StarGAN-VC.[5]

AutoVC uses the speaker embeddings produced by the speaker encoder, while the baselines only use the one-hot embeddings of the speakers. To avoid unfair comparison and study if the performance advantage of AutoVC simply comes from the speaker embeddings, we implement another version of AutoVC, called AutoVC-one-hot, that also uses one-hot embeddings of the speakers.

To construct the utterances for the MTurk evaluation, 10 speakers, five male and five female, are randomly chosen from the 20 speakers in the training set. We then produce $10 \times 10 = 100$ conversions by converting a test utterance of each of the 10 speakers to each of the 10 speakers' voice. Each test unit, called HIT, contains conversion results of the same source-target speaker pair of the three algorithms, so there are 100 HITs in total. Each HIT is assigned to 10 subjects.

Figure 5.5(a) presents the MOS scores, and Figure 5.5(b) presents the similarity scores. We are dividing the audio into four gender groups, male to male, male to female, female to male and female to female, and summarize the scores within each gender group. As shown in Figure 5.5(a), the perceptual quality of the speech generated by AutoVC is much better than the

[4]https://github.com/jjery2243542/voice_conversion
[5]https://github.com/liusongxiang/StarGAN-Voice-Conversion



(a) MOS (b) Similarity

Figure 5.4: Subjective evaluation results for zero-shot conversion.

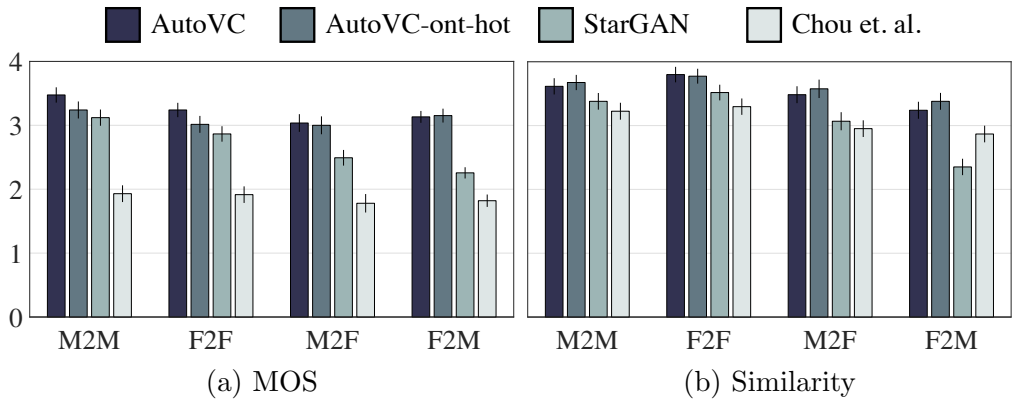
baselines'. The MOS scores of AutoVC are above 3 for all groups, whereas those for the baselines almost all fall below 3. To give readers a better idea of what this means. Notice that the MOS for 16kHz natural speech is around 4.5. The MOS scores of the current state-of-the-art speech synthesizers are between 4 and 4.5 [92, 152]. The highest score in 2016 Voice Conversion Challenge [151] for *parallel* conversion is 3.8 for same-gender conversions, and 3.2 for cross-gender conversion. Therefore, our subjective evaluation results show that AutoVC approaches the performance of parallel conversion systems in terms of naturalness, and is much better than existing non-parallel conversion systems.

In terms of similarity, AutoVC also outperforms the baselines. Note that for the baseline algorithms, there is a significant degrade from same-gender conversion to cross-gender conversion, but AutoVC algorithms do not display such a degrade. Finally, there is no significant difference between AutoVC and AutoVC-one-hot, which implies that the performance gain of AutoVC does not result from the use of the speaker encoder.

5.5.3 Zero-Shot Conversion

Now we are ready to go beyond the traditional conversion task toward zero-shot conversion, where the target speakers are absent in the training set and only a few (20 seconds) utterances of each target speaker are available for reference. Since there are no zero-shot conversion baselines, we will compare the results within AutoVC.

The experiment settings are almost the same as in Section 5.5.2, except



(a) MOS (b) Similarity

Figure 5.5: Subjective evaluation results for traditional conversion.

that the training set is expanded to 40 speakers to improve the generalizability to unseen speakers. Ten seen speakers and 10 unseen speakers are selected for MTurk evaluation, so there are a total of 400 source-target speaker pairs, each producing one conversion utterance. Each HIT contains four utterances, summing up to 100 HITs in total. Each HIT is assigned to 10 subjects.

Figure 5.4 presents the scores. There are three observations. First, for conversions among seen speakers, the performance is comparable to that in Section 5.5.2. Note that in this experiment, AutoVC is trained on 40 speakers, which doubles the number of speakers used in the experiment in Section 5.5.2. Therefore, this comparable performance on seen speakers indicates that AutoVC is scalable to a large number of speakers in the training set.

Second, in terms of MOS score, AutoVC shows good generalizations to unseen speakers, with the MOS score exceeding 3 in most settings. This means, even for unseen speakers, AutoVC is still able to outperform most existing non-parallel conversion algorithms.

Finally, in terms of the similarity score, there is an interesting observation that as long as seen speakers are included in either side of the conversions, the performance is comparable. There is a significant gap between conversions from unseen speakers to unseen speakers and the rest of the paradigms. Nevertheless, even for conversions within unseen speakers, which is the most challenging case, the similarity scores are still very competitive, which demonstrates AutoVC's competence in zero-shot conversion.

Table 5.1: Assessment of the reconstruction quality and speaker disentanglement of AutoVC.

	Narrow	AutoVC	Wide
Recon. Error	34.6	8.59	3.85
Class. Acc.	7.50%	12.0%	70.5%

5.5.4 Bottleneck Dimension Analysis

Our justifications for the proposed style transfer autoencoder lies in the claim that the bottleneck dimension affects perfect reconstruction and disentanglement of content code and source speaker information, and that there exists a desirable bottleneck dimension where both properties hold (Figure 5.2). In this section, we will empirically validate this claim.

We measure AutoVC's reconstruction quality and the degree of disentanglement between the content code and the source speaker information. The reconstruction quality is measured by the ℓ_2-norm of reconstruction error in the training set. Lower reconstruction error means higher reconstruction quality. The disentanglement is measured by training a speaker classifier on the content code and computing the classification accuracy on the training set. Higher classification accuracy means better disentanglement. The speaker classifier consists of 3 fully-connected layers with 2,048, 1,024 and 1,024 hidden nodes respectively in each layer and softplus activation. The output activation is softmax and the training loss is cross entropy. The model architecture and experiment setting follow those in Section 5.5.3, so the speaker classification is on the 40 seen speakers.

As references, we introduce two anchor models. The first model, which we name the "too narrow" model, reduces the dimensions of $C_{1\rightarrow}$ and $C_{1\leftarrow}$ from 32 to 16, and increases the downsampling factor from 32 to 128 (note that higher downsampling factor means lower temporal dimension). The second model, which we name the "too wide" model, increases the dimensions of $C_{1\rightarrow}$ and $C_{1\leftarrow}$ to 256, and decreases the sampling factor to 8, and λ is set to 0. Supposedly, according to Figure 5.2, the "too narrow" model should have low classification accuracy (good disentanglement) but high reconstruction error (poor reconstruction). The "too wide" model should have low reconstruction error (good reconstruction) but high classification accuracy (poor disentanglement). The normal AutoVC model should have both low recon-

struction error (good reconstruction) and low classification accuracy (good disentanglement).

Table 5.1 shows the reconstruction error and speaker classification accuracy for the three models. As expected, as the bottleneck dimension decreases, the reconstruction error increases and the classification accuracy decreases. What is interesting is that the normal AutoVC model does strike a good balance, with reconstruction error almost as low as the "too wide" model and the classification accuracy almost as low as the "too narrow" model. It is worth mentioning that Chou et al. [140] also perform a similar classification experiment to test disentanglement, and the classification accuracy is 45.1% on 20 speakers, after an adversarial training is introduced to force disentanglement. We also perform a classification on the same 20 speakers, and the classification accuracy is 14.2%. Of course, the direct comparison of these two numbers is unfair, but it at least validates that the effect of bottleneck dimension adjustment on speaker disentanglement is no worse, if not better, than that of the more sophisticated adversarial training.

5.6 Summary

We have proposed AutoVC, a non-parallel voice conversion algorithm that significantly outperforms the existing state-of-the-art, and that is the first to perform zero-shot conversions. In sharp contrast to its performance advantage is its simple autoencoder structure that trains only on self-reconstruction, and a bottleneck tuning to balance between reconstruction quality and speaker disentanglement. In an era of building increasingly sophisticated algorithms for style transfer, the success of AutoVC suggests that it is time to return to simplicity, because sometimes an autoencoder with a careful bottleneck design is all you need to make a difference.

CHAPTER 6

UNSUPERVISED SPEECH DECOMPOSITION VIA TRIPLE INFORMATION BOTTLENECK

6.1 Introduction

Human speech conveys a rich stream of information, which can be roughly decomposed into four important components: *content*, *timbre*, *pitch* and *rhythm*. The language content of speech comprises the primary information in speech, which can also be transcribed to text. Timbre carries information about the voice characteristics of a speaker, which is closely connected with the speaker's identity. Pitch and rhythm are the two major components of *prosody*, which expresses the emotion of the speaker. Pitch variation conveys the aspects of the tone of the speaker, and rhythm characterizes how fast the speaker utters each word or syllable.

For decades, speech researchers have sought to obtain disentangled representations of these speech components, which are useful in many speech applications. In speech analysis tasks, the disentanglement of speech components helps to remove interference introduced by irrelevant components. In speech generation tasks, disentanglement is the foundation of many applications, such as voice conversion [153], prosody modification [154], emotional speech synthesis [155], and low bit-rate speech encoding [156], to name a few.

Recently, state-of-the-art voice conversion systems have been able to obtain a speaker-invariant representation of speech, which disentangles the speaker-dependent information [157, 140, 153]. However, these algorithms are only able to disentangle timbre. The remaining aspects, *i.e.*, content, pitch, and rhythm are still lumped together. As a result, the converted speech produced by these algorithms differs from the source speech only in terms of timbre. The pitch contour and rhythm remain largely the same.

From an information-theoretic perspective, the success in timbre disentanglement can be ascribed to the availability of a speaker identity label,

which preserves almost all the information of timbre, such that voice conversion systems can "subtract" such information from speech. For example, AutoVC [157], a voice conversion system, constructs an autoencoder for speech and feeds the speaker identity label to the decoder. As shown in Figure 6.2(a), by constructing an information bottleneck between the encoder and decoder, AutoVC can force the encoder to remove the timbre information, because the equivalent information is supplied to the decoder directly. It is worth noting that although the speaker identity is also correlated with the pitch and timbre information, the information overlap is relatively small, so the speaker identity cannot serve as labels for pitch and rhythm. If we had analogous information-preserving labels for timbre, rhythm or pitch, the disentanglement of these aspects would be straightforward, simply by utilizing these labels the same way voice conversion algorithms use the speaker identity label.

However, obtaining annotations for these other speech components is challenging. First, although language content annotation is effectively provided by text transcriptions, obtaining a large number of text transcriptions is expensive, especially for low-resourced languages. Therefore, here, we will focus on unsupervised methods that do not rely on text transcriptions. Second, the rhythm annotation, which is essentially the length of each syllable, can only be obtained with the help of text transcriptions, which are again unavailable under our unsupervised setting. Finally, for pitch annotation, although the pitch information can be extracted as pitch contour using pitch extraction algorithms, the pitch contour itself is entangled with rhythm information, because it contains the information of how long each speech segment is. Without the information preserving labels, disentangling content, rhythm and pitch becomes an under-determined problem. Hence, here we ask: Is it possible to decompose these remaining speech components without relying on text transcriptions and other information-preserving labels?

In this chapter, we propose SpeechSplit, a speech generative model that can blindly decompose speech into content, timbre, pitch, and rhythm, and generate speech from these disentangled representations. Thus, SpeechSplit is among the first algorithms that can enable flexible conversion of different aspects to different styles without relying on any text transcription. To achieve unsupervised decomposition, SpeechSplit introduces an encoder-decoder structure with three encoder channels, each with a different,

carefully crafted information bottleneck design. The information bottleneck is imposed by two mechanisms: first, a constraint on the physical dimension of the representation, which has been shown effective in AutoVC, and second, the introduction of noise by randomly resampling along the time dimension, which has been shown effective in [158]. We find that subtle differences in the information bottleneck design can force different channels to pass different information, such that one passes language content, one passes rhythm, and one passes pitch information, thereby achieving the blind disentanglement of all speech components.

Besides direct value in speech applications, SpeechSplit also provides insight into a powerful design principle that can be broadly applied to any disentangled representation learning problem: in the presence of an information bottleneck, a neural network will prioritize passing through the information that cannot be provided elsewhere. This observation inspires a generic approach to disentanglement.

6.2 Related Work

The Source-Filter Model Early research on speech generation proposed the source-filter model [5], and many subsequent research efforts try to decompose speech into the source that includes pitch and the filter that includes content, using signal processing approaches, such as linear predictive coding [159], cepstral analysis [160], temporally stable power spectral analysis [161] and probabilistic approaches [162]. However, these approaches do not consider the prosody aspects of speech.

Voice Conversion Inspired by the style transfer and disentanglement techniques in computer vision [163, 141, 164], many approaches based on variational autoencoders (VAEs) and generative adversarial networks (GANs) have been proposed in the field of voice conversion to disentangle the timbre information from the speech. VAE-VC [57] directly applies a VAE for voice conversion, where the encoder produces a speaker-independent content embedding. After that, VAE-GAN [138] replaces the decoder of the VAE with a GAN when generating the converted speech to improve the quality of the conversion results. CDVAE-VC [139] uses two VAEs working on different speech features, one on STRAIGHT spectra [161], and one on mel-cepstral coeffi-

cients (MCCs), and encourages that the latent representation can reconstruct both features well. ACVAE-VC [63] introduces an auxiliary classifier for the conversion outputs, and encourages the converted speech to be correctly classified as the source speaker. Chou et al. [140] introduced a classifier for the latent code, and discouraged the latent code to be correctly classified as the target speaker. Inspired by image style transfer frameworks, [144] and [58] adapted CycleGan [141] and StarGan [164] respectively for voice conversion. Later, CDVAE-VC was extended by directly applying GAN [165] to improve the degree of disentanglement. Chou and Lee [153] used instance normalization to further disentangle speaker from content, and thus can convert to speakers that are not seen during training. StarGan-VC2 [166] refined the adversarial framework by conditioning the generator and discriminator on the source speaker label, in addition to the target speaker label. Recently, Qian et al. [157] proposed AutoVC, a simple autoencoder based method that disentangles the timbre and content using information-constraining bottlenecks. Later, Qian et al. [167] fixed the pitch jump problem of AutoVC by F0 conditioning. Besides, the time-domain deep generative model is gaining more research attention for voice conversion [168, 169, 170]. However, these methods only focus on converting timbre, which is only one of the speech components.

Prosody Disentanglement There have been a few recent text-to-speech (TTS) systems that seek to disentangle the prosody information to generate expressive speech. Skerry-Ryan et al. [171] introduced a Tacotron based speech synthesizer that can disentangle prosody from speech content by having an encoder that can extract the prosody information from the original speech. Mellotron [172] is a speech synthesizer conditional on both explicit prosody labels and latent prosody code to capture and disentangle different aspects of the prosody information. CHiVE [173] introduces a hierarchical encoder-decoder structure that is conditioned on a set of prosodic features and linguistic features. However, these TTS systems all require text transcriptions, which, as discussed, makes the task easier but limits their applications to high-resource language. Besides TTS systems, Parrotron [174] disentangles prosody by encouraging the latent codes to be the same as the corresponding phone representation of the input speech. However, Parrotron still requires text transcriptions to label the phone representation, as well as to generate the synthetic parallel dataset. Polyak and Wolf [158] proposed,

to the best of our knowledge, the only prosody disentanglement algorithm that does not rely on text transcriptions, which attempts to remove the rhythm information by randomly resampling the input speech. However, the effect of their prosody conversion is not very pronounced. In this chapter, our objective is to achieve effective prosody conversion without using text transcriptions, which is more flexible for low-resource languages.

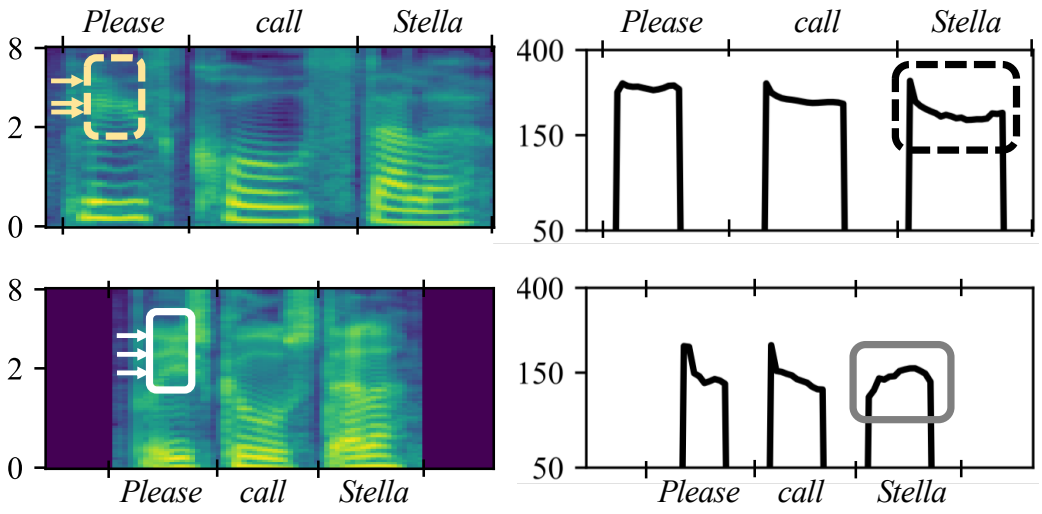


Figure 6.1: Spectrograms (left) and pitch contours (right) of two utterances of the same sentence *"Please call Stella"*. The left rectangle marks highlight the formant structures of the phone *"ea"*. The arrows mark the frequencies of the second, third and fourth formants. The right rectangle marks highlight the pitch tones of the word *"Stella"*.

6.3 Information in Speech

Since this chapter focuses on the decomposition of speech information into rhythm, pitch, timbre, and content, we provide here a brief primer on each of these components. Figure 6.1 shows the spectrograms (left) and pitch contours (right) of utterances of the sentence *"Please call Stella"*. Throughout this chapter, the term "spectrogram" refers to the magnitude spectrogram.

Rhythm Rhythm characterizes how fast the speaker utters each syllable. As shown in Figure 6.1, each spectrogram is divided into segments, which correspond to each word, as marked on the horizontal axis. So the lengths of these segments reflect the rhythm information. In the top spectrogram,

each segment is long, indicating a slow speaker; in the bottom spectrogram, each segment is short, indicating a fast speaker.

Pitch Pitch is an important component of intonation. One popular representation of the pitch information is the *pitch targets* [175], which is defined as the intended pitch, *e.g.* rise or fall, high or low *etc.*, of each syllable. The pitch information, or pitch target information, is contained in the pitch contour, because the pitch contour is generally considered as the result of a constant attempt to hit the pitch targets of each syllable, subject to the physical constraints [175]. However, the pitch contour also entangles other information. First, the pitch contour contains the rhythm information, because each nonzero segment of the pitch contour represents a voiced segment, which typically corresponds to a word or a syllable. So the length of each voiced segment indicates how fast the speaker speaks. Second, the pitch range reflects certain speaker identity information — female speakers tend to have a high pitch range, as shown in the upper panel of Figure 6.1, and male speakers tend to have a low pitch range, as shown in the lower panel of Figure 6.1. Here, we assume that the impact of the speaker identity on the pitch contour is linear. In other words, if we normalize the pitch contour to a common mean and standard deviation, the speaker identity information will be removed. To sum up, the pitch contour entangles the information of speaker identity, rhythm and pitch; the normalized pitch contour only contains the information of the latter two.

Timbre Timbre is perceived as the voice characteristics of a speaker. It is reflected by the *formant frequencies*, which are the resonant frequency components in the vocal tract. In a spectrogram, the formants are shown as the salient frequency components of the spectral envelope. In Figure 6.1, the rectangles and arrows on the spectrogram highlight three formants. As can be seen, the top spectrogram has a higher formant frequency range, indicating a bright voice; the bottom spectrogram has a lower formant frequency range, indicating a deep voice.

Content In English and many other languages, the basic unit of content is the phone. Each phone comes with a particular formant pattern. For example, the three formants highlighted in Figure 6.1 are the second, third and fourth lowest formants of the phone *"ea"* as in *"please"*. Although their formant frequencies have different ranges, which indicates their timbre difference, they have the same pattern — they tend to cluster together and

are far away from the lowest formant (which is at around 100 Hz).

6.4 SpeechSplit

This section introduces SpeechSplit . For notation, uppercased letters, X and $\boldsymbol{X}$, denote random scalars and vectors respectively; lowercased letters, x and $\boldsymbol{x}$, denote deterministic scalars and vectors respectively.

6.4.1 Problem Formulation

Denote $\boldsymbol{S} = \{\boldsymbol{S}_t\}$ as a speech spectrogram, where t is the time index. Denote the speaker's identity as U. We assume that $\boldsymbol{S}$ and U are generated through the following random generative processes

$$\boldsymbol{S} = g_s(\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{F}, \boldsymbol{V}), \quad U = g_u(\boldsymbol{V}) \tag{6.1}$$

where $\boldsymbol{C}$ denotes content; $\boldsymbol{R}$ denotes rhythm; $\boldsymbol{F}$ denotes pitch target; $\boldsymbol{V}$ denotes timbre. $g_s(\cdot)$ and $g_u(\cdot)$ are assumed to be a one-to-one mapping. Note that here we assume $\boldsymbol{C}$ also accounts for the residual information that is not included in rhythm, pitch or timbre.

Our goal is to construct an autoencoder-based generative model for speech, such that the hidden code contains disentangled representations of the speech components. We formally denote the representations as $\boldsymbol{Z}_c$, $\boldsymbol{Z}_r$ and $\boldsymbol{Z}_f$, and these representations should satisfy

$$\boldsymbol{Z}_c = h_c(\boldsymbol{C}), \quad \boldsymbol{Z}_r = h_r(\boldsymbol{R}), \quad \boldsymbol{Z}_f = h_f(\boldsymbol{F}) \tag{6.2}$$

where $h_c(\cdot)$, $h_r(\cdot)$ and $h_f(\cdot)$ are all one-to-one mappings.

6.4.2 AutoVC and Its Limitations

Since SpeechSplit inherits the information bottleneck mechanism proposed in AutoVC, it is necessary to first review its framework and limitations. Figure 6.2(a) shows the framework of AutoVC, which consists of an encoder and a decoder. The encoder has an information bottleneck at the end (shown as the grey tip), which is implemented as a hard constraint on code dimensions. The input to the encoder is speech spectrogram $\boldsymbol{S}$, and the output of

the encoder is called the speech code, denoted as $\boldsymbol{Z}$. The decoder takes $\boldsymbol{Z}$ and the speaker identity label U as its inputs, and produces a speech spectrogram $\hat{\boldsymbol{S}}$ as output. We formally denote the encoder as $E(\cdot)$, and the decoder as $D(\cdot, \cdot)$. The AutoVC pipeline can be expressed as

$$\boldsymbol{Z} = E(\boldsymbol{S}), \quad \hat{\boldsymbol{S}} = D(\boldsymbol{Z}, U) \tag{6.3}$$

During training, the output of the decoder tries to reconstruct the input spectrogram:

$$\min_{\boldsymbol{\theta}} \mathbb{E}[\|\hat{\boldsymbol{S}} - \boldsymbol{S}\|_2^2] \tag{6.4}$$

where $\boldsymbol{\theta}$ denotes all the trainable parameters.

It can be shown that if the information bottleneck is tuned to the right size, this simple scheme can achieve disentanglement of the timbre information as

$$\boldsymbol{Z} = h(\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{F}) \tag{6.5}$$

Figure 6.2(a) provides an intuitive explanation of why this is possible. As can be seen, speech is represented as a concatenation of different blocks, indicating the content, rhythm, pitch and timbre information. Note that speaker identity is represented with the same block style as timbre because it is assumed to preserve equivalent information to timbre according to Equation (6.1). Since the speaker identity is separately fed to the decoder, the decoder can still have access to all the information to perform self-reconstruction even if the encoder does not preserve the timbre information in its output. Therefore, when the information bottleneck is binding, the encoder will remove the timbre information. However, $\boldsymbol{Z}$ still lumps content, rhythm, and pitch together. As a result, AutoVC can only convert timbre.

6.4.3 The SpeechSplit Framework

Figure 6.2(b) illustrates the SpeechSplit framework. SpeechSplit is also an autoencoder with an information bottleneck. However, in order to further decompose the remaining speech components, SpeechSplit introduces three encoders with heterogeneous information bottleneck, which are a *content encoder*, a *rhythm encoder*, and a *pitch encoder*. Following are the details of the encoders and the decoder of SpeechSplit.

The Encoders As shown in Figure 6.2(b), all three encoders are almost the same, but with two subtle differences. First, the input to the content

encoder and rhythm encoder is speech $\boldsymbol{S}$, whereas the input to the pitch encoder is the normalized pitch contour, which we denote as $\boldsymbol{P}$. As discussed in Section 6.3, the normalized pitch contour $\boldsymbol{P}$ refers to the pitch contour that is normalized to have the same mean and variance across all the speakers, so the normalized pitch contour only contains the pitch information, $\boldsymbol{F}$, and rhythm information, $\boldsymbol{R}$, but no speaker ID information, U.

Second, the content encoder and pitch encoder perform a random resampling operation along the time dimension of the input. Random resampling involves two steps of operations. The first step is to divide the input into segments of random lengths. The second step is to randomly stretch or squeeze each segment along the time dimension. Therefore, random resampling can be regarded as an information bottleneck on rhythm. All the encoders have the physical information bottleneck at the output. The final outputs of the encoders are called content code, rhythm code and pitch code, which are denoted as $\boldsymbol{Z}_c$, $\boldsymbol{Z}_r$ and $\boldsymbol{Z}_f$ respectively. Formally, denote the content encoder as $E_c(\cdot)$, rhythm encoder as $E_r(\cdot)$ and pitch encoder as $E_f(\cdot)$, and denote the random resampling operation as $A(\cdot)$. Then we have

$$\boldsymbol{Z}_c = E_c(A(\boldsymbol{S})), \quad \boldsymbol{Z}_r = E_r(\boldsymbol{S}), \quad \boldsymbol{Z}_f = E_f(A(\boldsymbol{P})) \tag{6.6}$$

The Decoder The decoder takes all the speech code and the speaker identity label (or embedding) as its inputs, and produce a speech spectrogram as output, *i.e.*,

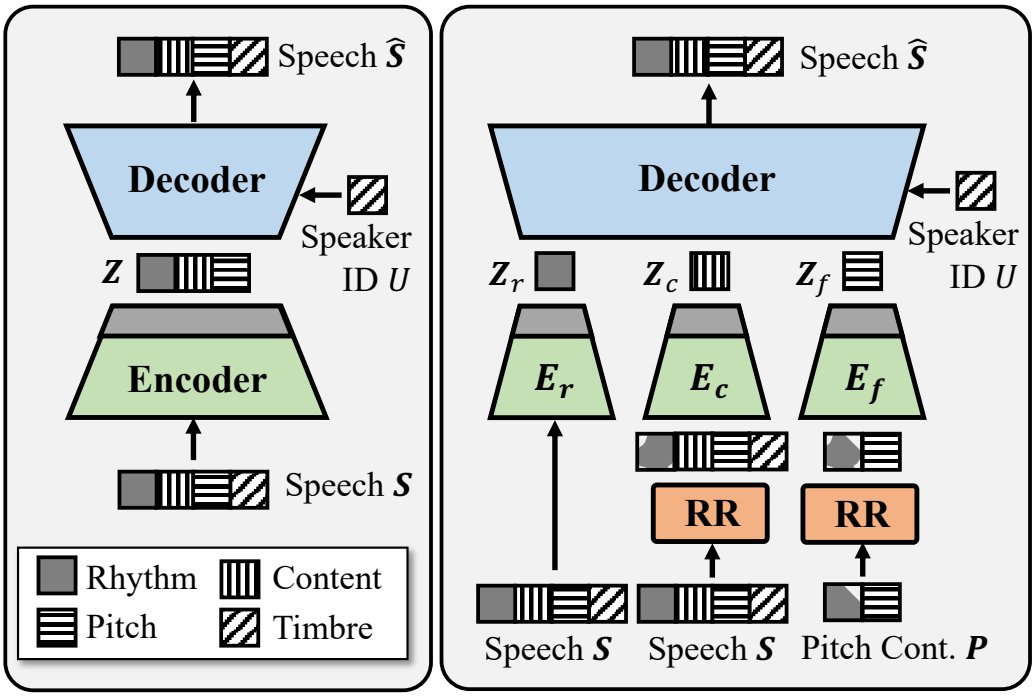
$$\hat{\boldsymbol{S}} = D(\boldsymbol{Z}_c, \boldsymbol{Z}_r, \boldsymbol{Z}_f, U) \tag{6.7}$$

During training, the output of the decoder tries to reconstruct the input spectrogram, which is the same as in Equation (6.4).

Counter-intuitive as it may sound, we claim that when all the information bottlenecks are appropriately set and the network representation power is sufficient, a minimizer of Equation (6.4) will satisfy the disentanglement condition as in Equation (6.2). In what follows, we will explain why such decomposition is possible.

6.4.4 Why Does It Force Speech Decomposition?

Figure 6.2 provides an intuitive illustration of how SpeechSplit achieves speech decomposition, where a few important assumptions are made.



(a) AutoVC (b) SpeechSplit

Figure 6.2: Frameworks and AutoVC and SpeechSplit and illustration of why they can perform disentanglement. Signals are represented as blocks to denote their information components. E_r denotes the rhythm encoder; E_c denotes the content encoder; E_f denotes the pitch encoder. "RR" denotes random resampling. "Pitch Cont." is short for the normalized pitch contour. The grey block at the tip of the encoders denotes the information bottleneck. Some rhythm blocks have some holes in them, which represents that a portion of the rhythm information is lost. The bottlenecks force the encoders to pass only the information that other encoders cannot supply, hence achieving the disentanglement.

Assumption 1: The random resampling operation will contaminate the rhythm information $\boldsymbol{R}$.

Assumption 2: The random resampling operation will not contaminate the other speech components.

Assumption 3: The pitch contour $\boldsymbol{P}$ contains all the pitch information and a portion of rhythm information.

As shown in Figure 6.2(b), speech contains four blocks of information. When it passes through the random resampling operation, a random portion of the rhythm block is wiped (shown as the holes in the rhythm block at the output of the RR module), but the other blocks remain intact. On

the other hand, the normalized pitch contour mainly contains two blocks, the pitch block, and the rhythm block. The rhythm block is missing a corner because the normalized pitch contour does not contain all the rhythm information, and it misses even more when it passes through the random resampling module.

Similar to the AutoVC claim, the timbre information is directly fed to the decoder, so all the encoders do not need to encode the timbre information. Therefore, this section focuses on explaining why SpeechSplit can force the encoders to separately encode the content, pitch, and timbre.

First, the rhythm encoder $E_r(\cdot)$ is the only encoder that has access to the complete rhythm information $\boldsymbol{R}$. The other two encoders only preserve a random portion of $\boldsymbol{R}$, and there is no way for $E_r(\cdot)$ to guess which part is lost and thus only supply the lost part. Therefore, $E_r(\cdot)$ must pass all the rhythm information. Meanwhile, the other aspects are available in the other two encoders. So if $E_r(\cdot)$ is forced to lose some information by its information bottleneck, it will prioritize removing the content, pitch, and timbre.

Second, given that $E_r(\cdot)$ only encodes $\boldsymbol{R}$, then the content encoder $E_c(\cdot)$ becomes the only encoder that can encode all the content information $\boldsymbol{C}$, because the pitch encoder does not have access to $\boldsymbol{C}$. Therefore, $E_c(\cdot)$ must pass all the content information. Meanwhile, the other aspects can be supplied elsewhere, so the rhythm encoder will remove the other aspects if the information bottleneck is binding.

Finally, with $E_r(\cdot)$ encoding only $\boldsymbol{R}$ and $E_c(\cdot)$ encoding only $\boldsymbol{C}$, the pitch encoder $E_f(\cdot)$ must encode the pitch information. All the other aspects are supplied in other channels, so $E_f(\cdot)$ will prioritize removing these aspects if the information bottleneck is binding.

Simply put, if each encoder is only allowed to pass one block, then the arrangement in Figure 6.2 is the only way to ensure full recovery of the speech information.

6.4.5 Network Architecture

Figure 6.3 shows the architecture of SpeechSplit. The left module corresponds to the encoders and the right to the decoder. All three encoders share a similar architecture, which consists of a stack of 5×1 convolutional

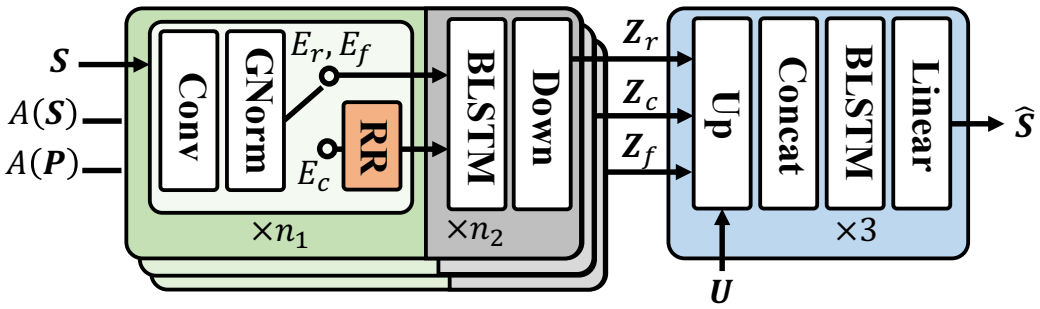


Figure 6.3: The architecture of SpeechSplit. "GNorm" denotes group normalization; "RR" denotes random resampling; "Down" and "Up" denote downsampling and upsampling operations respectively. "Linear" denotes linear projection layer. $\times n$ denotes the module above is repeated n times.

Table 6.1: Hyperparameter settings of the encoders.

	Rhythm	**Content**	**Pitch**
Conv Layers	1	3	3
Conv Dim	128	512	256
Norm Groups	8	32	16
BLSTM Layers	1	2	1
BLSTM Dim	1	8	32
Downsample Factor	8	8	8

layers followed by group normalization [176]. For the content encoder, the output of each convolutional layer is passed to a random resampling module to further contaminate rhythm. The final output of the convolutional layers is fed to a stack of bidirectional-LSTM layers to reduce the feature dimension, which is then passed through a downsampling operation to reduce the temporal dimension, producing the hidden representations. Table 6.1 shows the hyperparameter settings of each encoder.

The decoder first upsamples the hidden representation to restore the original sampling rate. The speaker identity label U, which is a one-hot vector, is also repeated along the time dimension to match the temporal dimension of the other upsampled representations. All the representations are then concatenated along the channel dimension and fed to a stack of three bidirectional-LSTM layers with an output linear layer to produce the final output. The spectrogram is converted back to the speech waveform using the same wavenet-vocoder as in AutoVC.

6.5 Experiments

In this section, we will empirically verify the disentanglement capability of SpeechSplit. We will be visualizing our speech results using spectrogram and pitch contour. The frequency axis units of all the spectrograms are in kHz, and those of the pitch contour plots are in Hz.

6.5.1 Configurations

The experiments are performed on the VCTK dataset [150]. The training set contains 20 speakers where each speaker has about 15 minutes of speech. The test set contains the same 20 speakers but with different utterances, which is the conventional voice conversion setting. SpeechSplit is trained using the ADAM optimizer [177] with a batch size of 16 for 800k steps. Since there are no other algorithms that can perform blind decomposition so far, we will be comparing our result with AutoVC, a conventional voice conversion baseline.

The model selection is performed on the training dataset. Specifically, the physical bottleneck dimensions are tuned based on the criterion: when the input to one of the encoders or the speaker embedding is set to zero, the output reconstruction should not contain the corresponding information. As will be shown in Section 6.5.4, setting the inputs and speaker embedding to zero can measure the degree of disentanglement. From the models that satisfy this criterion, we pick the one with the lowest training error.

6.5.2 Rhythm, Pitch and Timbre Conversions

If SpeechSplit can decompose the speech into different components, then it should be able to separately perform style transfer on each aspect, which is achieved by replacing the input to the respective encoder with that of the target utterance. For example, if we want to convert pitch, we feed the target pitch contour to the pitch encoder. To convert timbre, we feed the target speaker id to the decoder.

We construct parallel speech pairs from the test set, where both the source and target speakers read the same utterances. Please note that we use the parallel pairs *only for testing*. During training, SpeechSplit is trained

without parallel speech data. For each parallel pair, we set one utterance as the source and one as the target, and perform seven different types of conversions, including three single-aspect conversions (rhythm-only, pitch-only and timbre-only), three double-aspect conversions (rhythm+pitch, rhythm+timbre, and pitch+timbre), and one all-aspect conversion.

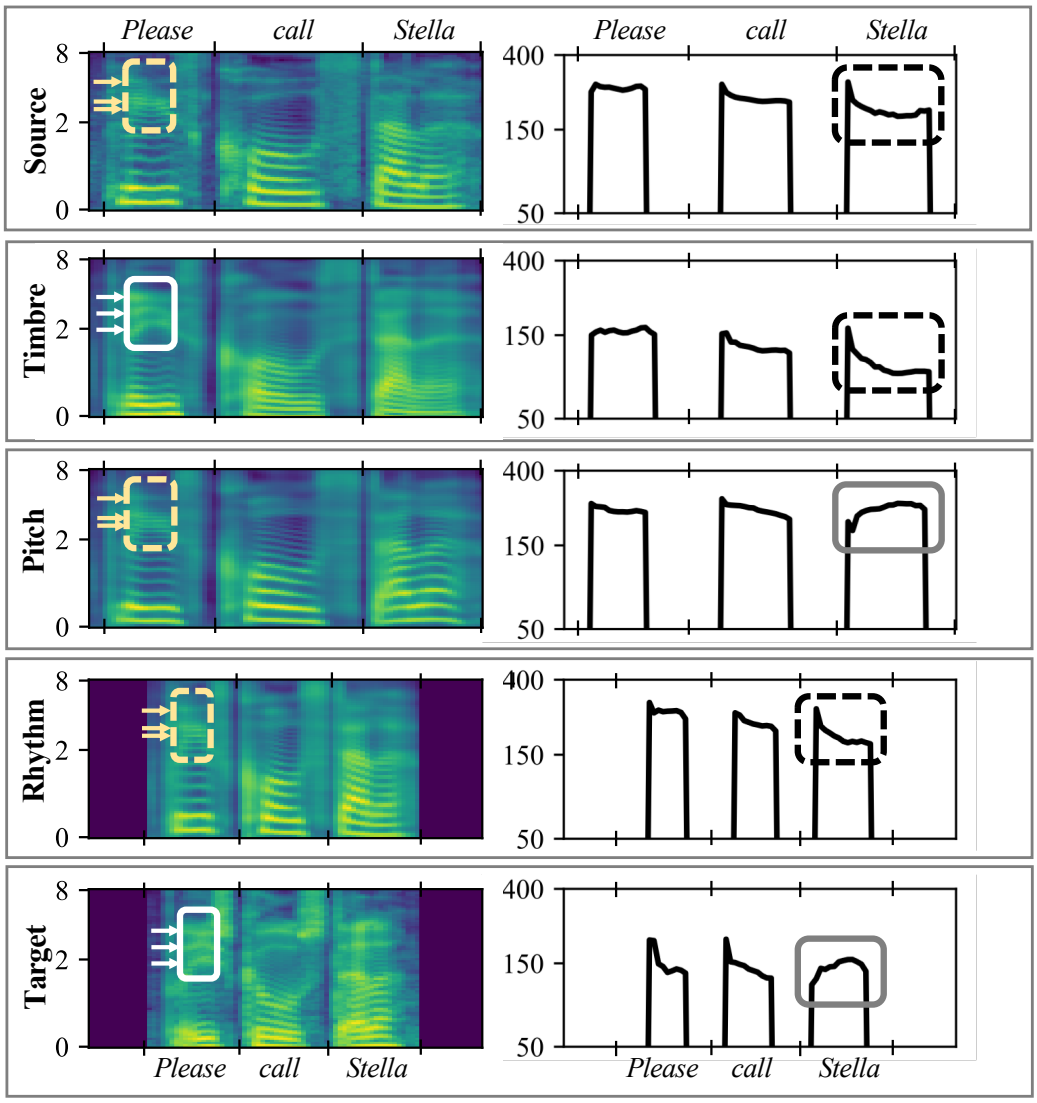


Figure 6.4: Spectrogram (left) and pitch contours (right) of single-aspect conversion results of the utterance *"Please call Stella"*. The left rectangle marks highlight the formant structures of the phone *"ea"*. The arrows mark the frequencies of the second, third and fourth formants. The right rectangle marks highlight the pitch tones of the word *"Stella"*.

Conversion Visualization Figure 6.4 shows the single-aspect conversion results on a speech pair uttering *"Please call Stella"*. The source speaker is a

Table 6.2: MOS of different conversion types/algorithms.

Rhythm Only	**Pitch Only**	**Timbre Only**
3.21	3.79	3.40
Rhythm+Pitch	**Rhythm+Timbre**	**Pitch+Timbre**
3.04	2.73	3.35
All Three	**AutoVC**	**Source**
2.79	3.24	4.65

slow female speaker, and the target speaker is a fast male speaker. As shown in Figure 6.4, SpeechSplit can separately convert each aspect. First, in terms of rhythm, note that the rhythm-only conversion is perfectly aligned with the target utterance in time, whereas the timbre-only and pitch-only conversions are perfectly aligned with the source utterance in time. Second, in terms of pitch, notice that the timbre-only and rhythm-only conversions have a falling tone on the word *"Stella"*, which is the same as the source utterance, as highlighted by the dashed rectangle. The pitch-only conversion has a rising tone on *"Stella"*, which is the same as the target utterance, as highlighted by the solid rectangles. Third, in terms of timbre, as highlighted by the rectangles on the spectrograms, the formants of pitch-only and rhythm-only conversions are as high as those of the source speech, and the formants of timbre-only conversions are as high as those in the target.

Subjective Evaluation We also perform a subjective evaluation on *Amazon Mechanical Turk* on whether the conversion of each aspect is successful. For example, to evaluate whether the different conversions convert pitch, we select 20 speech pairs that are perceptually distinct in pitch, and generate all the seven types of conversions, plus the AutoVC conversion and the source utterance as baselines. Each test is assigned to five subjects. In the test, the subject is presented with two reference utterances, which are the source and target utterances in a random order, and then with one of the nine conversion results. The subject is asked to select which reference utterance has a more similar pitch tone to the converted utterance. We compute the *pitch conversion rate* as the percentage of answers that choose the target utterance. We would expect the utterances with pitch converted to have a high pitch conversion rate; otherwise, the pitch conversion rate should be low. The *rhythm conversion rate* and *timbre conversion rate* are computed in a similar way.

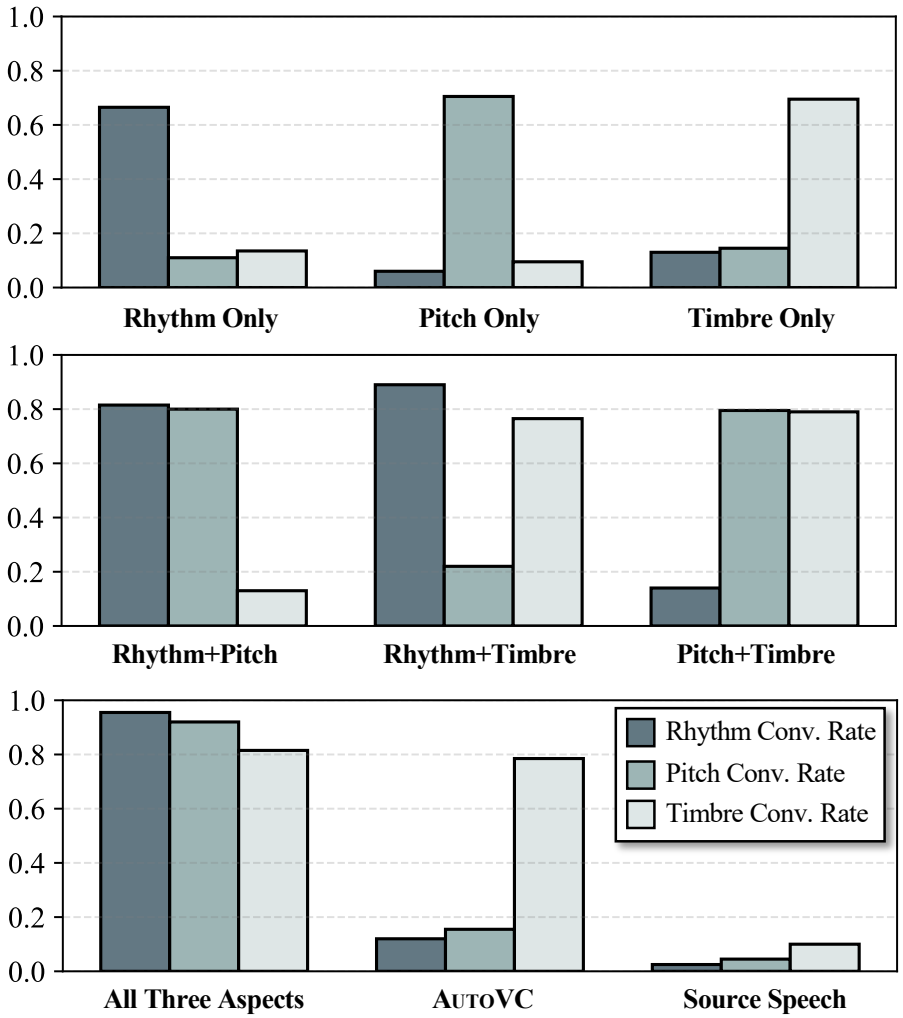


Figure 6.5: Subjective conversion rates of different conversion types. Each bar group corresponds to a conversion type/algorithm. The three bars within each group represent the rhythm, pitch and timbre conversion rates respectively.

Figure 6.5 shows the conversion rates of different types of conversions. As expected, the conversion rate is high when the corresponding aspect is converted, and low otherwise. For example, the pitch-only conversion has a high pitch conversion rate but low rhythm and timbre conversion rates; whereas the rhythm+timbre conversion has a high rhythm and timbre conversion rates but a low pitch conversion rate. It is worth noting that AUTOVC has a high timbre conversion rate, but low in the other, indicating that it only converts timbre. In short, both the visualization results and our subjective evaluation verify that each conversion can successfully convert the intended

aspects, without altering the other aspects, whereas AutoVC only converts timbre.

We also evaluate the MOS (mean opinion score), ranging from one to five, on the quality of the conversion, as shown in Table 6.2. There are a few interesting observations. First, the MOS of pitch conversion is higher than that of timbre and rhythm conversions, which implies that timbre and rhythm conversions are the more challenging tasks. Second, as the number of converted aspects increases, the MOS gets lower, because the conversion task gets more challenging.

Objective Evaluation Due to the lack of explicit labels of the speech components, it is difficult to fully evaluate the disentanglement results using objective metrics. However, we can still objectively evaluate the pitch-only conversion performance by comparing the pitch contour of the converted speech and the target pitch contour. Following [172], we use three metrics for the comparison: Gross Pitch Error (GPE) [178], Voice Decision Error (VDE) [178], and F0 Frame Error (FFE) [179]. SpeechSplit achieves a GPE of 1.04%, a VDE of 8.14%, and an FFE of 8.86%. As a reference, these results are comparable with the results reported in [172], with a slightly higher GPE and lower VDE and FFE. Note that these two sets of results cannot be directly compared, because the datasets are different, but they show the effectiveness of the SpeechSplit in disentangling pitch.

6.5.3 Mismatched Conversion Target

Since utterances with mismatched contents have different numbers of syllables and lengths, we would like to find out how SpeechSplit converts rhythm when the source and target utterances read different content. Figure 6.6 shows the rhythm-only conversion between a long utterance, *'And we will go meet her Wednesday'* (top-left panel), and a short utterance, "*Please call Stella"* (top-right panel).

The short to long conversion is shown in the bottom-left panel. It can be observed that the conversion tries to match the syllable structure of the long utterance by stretching its limited words. In particular, *"please"* is stretched to cover *"and we will"*, *"call"* to cover *"go meet"*, and *"Stella"* to cover *"her Wednesday"*. On the contrary, the long to short conversion, as shown in the

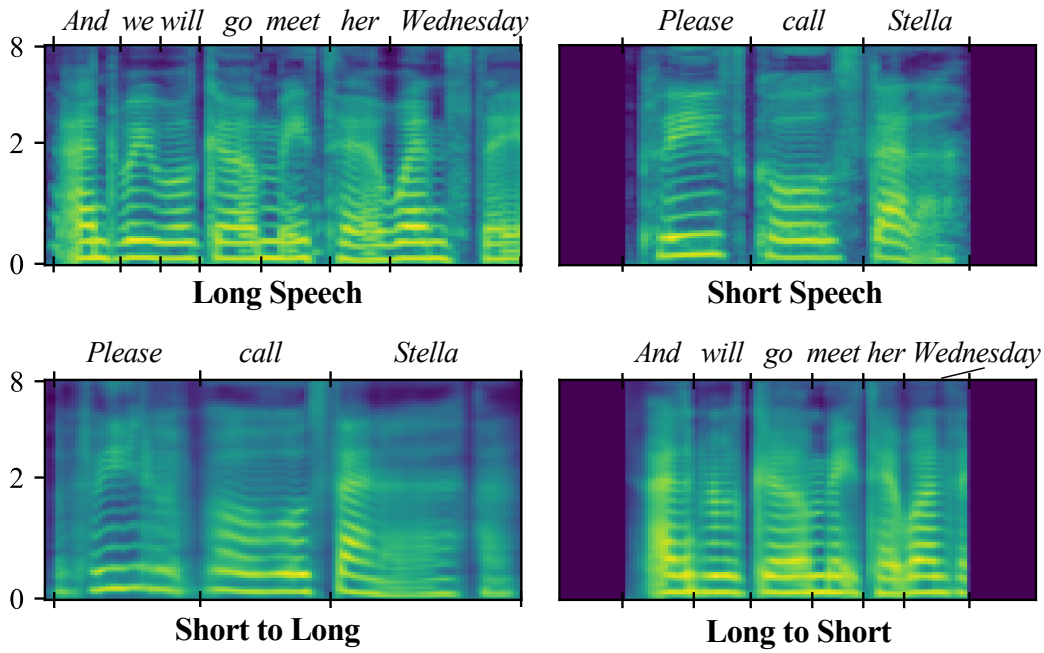


Figure 6.6: Rhythm-only conversion when the source and target speech have mismatched content.

bottom-right panel, tries to squeeze everything to the limited syllable slots in the short utterance. Intriguingly still, the word mapping between the long utterance and the short utterance is *exactly the same* as in the short to long conversion. In both cases, the word boundaries between the converted speech and the target speech are surprisingly aligned.

These observations suggest that SpeechSplit has an intricate *"fill in the blank"* mechanism when combining the rhythm information with content and pitch. The rhythm code provides a number of blanks, and the decoder fills the blanks with the content information and pitch information provided by the respective encoders. Furthermore, there seems to be an anchoring mechanism that associates the content and pitch with the right blank, which functions stably even if the blanks and the content are mismatched.

6.5.4 Removing Speech Components

To further understand the disentanglement mechanism of SpeechSplit, we generate spectrograms with one of the four components removed. To remove rhythm, content or pitch, we respectively set the input to the rhythm encoder, content encoder or pitch encoder to zero. To remove timbre, we

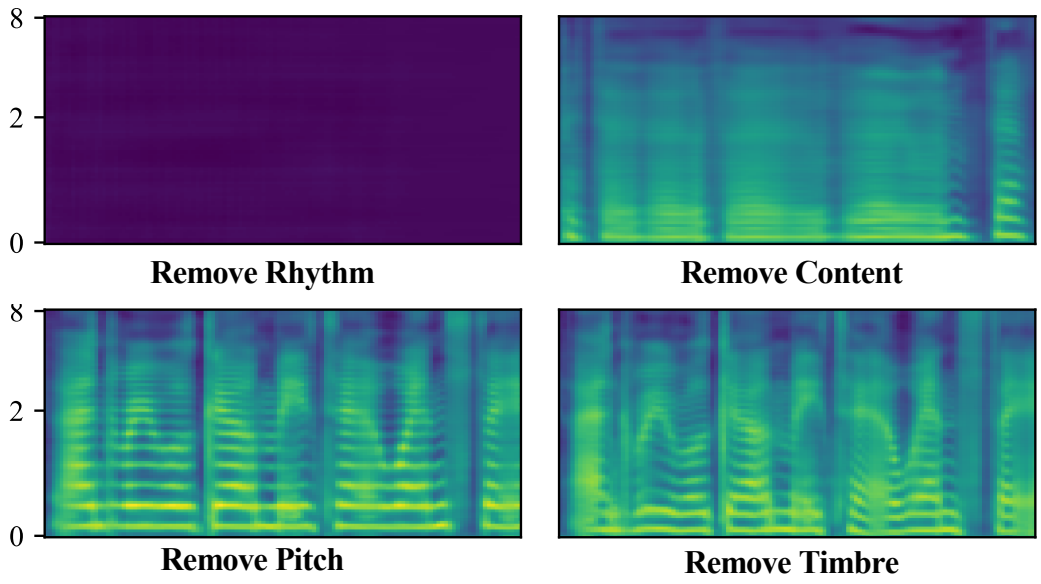


Figure 6.7: Reconstructed speech when one speech component is removed. The ground truth speech is in Figure 6.6 top-left panel.

set the speaker embedding to zero. Figure 6.7 shows the output spectrograms with one component removed. As can be observed, when the rhythm is removed, the output becomes zero, and when the content is removed, the output becomes a set of slots with no informative spectral shape. These findings are consistent with our "fill in the blank" hypothesis in Section 6.5.4. When rhythm code is removed, there is no slot to fill, and hence the output spectrogram is blank. When content is removed, there is nothing to fill in the blanks, resulting in a spectrogram with uninformative blanks. When the pitch is removed, the pitch of the output becomes completely flat, as can be seen from the flat harmonics. Finally, when timbre is removed, the formant positions of the output spectrogram shift, which indicates that the timbre has changed, possibly to an average speaker. These results further verify that SpeechSplit can separately model different speech components.

6.5.5 Varying the Information Bottleneck

In this section, we would like to verify our theoretical explanation in Section 6.4.4 by varying the information bottleneck and see if SpeechSplit will still act as our theory predicts.

According to Figure 6.2, if the physical information bottleneck of the

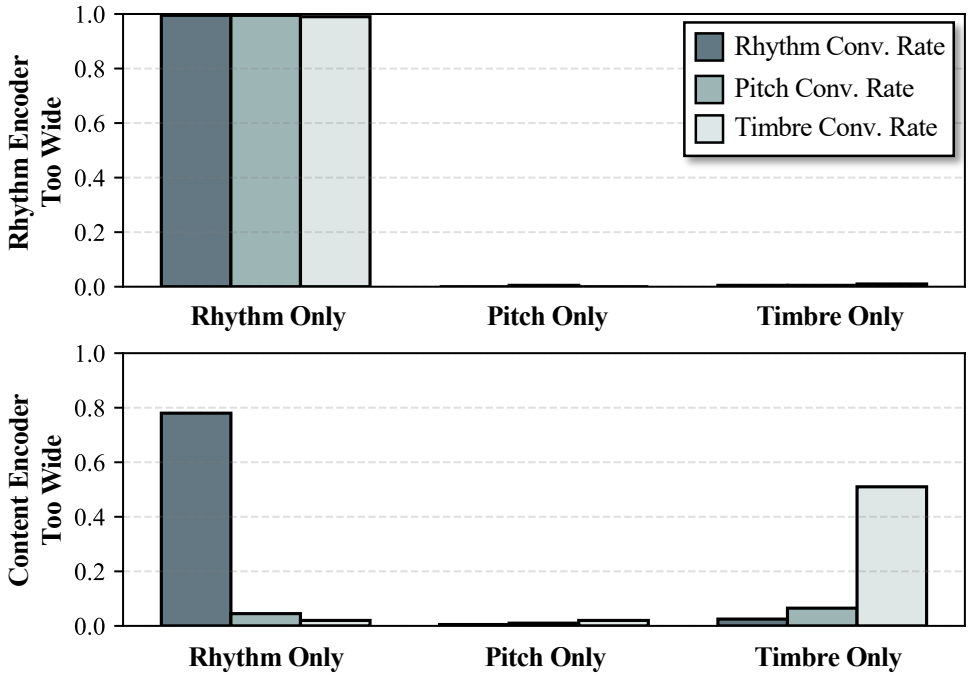


Figure 6.8: Subjective conversion rates of single-aspect conversions of SpeechSplit when the information bottleneck of the rhythm encoder (top panel) or the content encoder (bottom panel) is too wide. Each bar group corresponds to a conversion type/algorithm. The three bars within each group represent the rhythm, pitch and timbre conversion rates respectively.

rhythm encoder is too wide, then the rhythm encoder will pass all the information through, and the content encoder, pitch encoder and speaker identity will be useless. As a result, the rhythm-only conversion will convert all the aspects. On the other hand, the pitch-only and timbre-only conversions will alter nothing. Similarly, if the physical information bottleneck of the content encoder is too wide, but random sampling is still present, then the content encoder will pass almost all the information through, except for the rhythm information, because the random resampling operations still contaminate the rhythm information and SpeechSplit would still rely on the rhythm encoder to recover the rhythm information. As a result, the rhythm-only conversion would still convert rhythm, but the pitch-only and timbre-only conversions would barely alter anything.

Figure 6.8 shows the subjective conversion rates of single-aspect conversions when the physical bottleneck of rhythm encoder or the content encoder is too wide. These results agree with our theoretical predictions. When the

rhythm encoder physical bottleneck is too wide, the rhythm-only conversion converts all the aspects, while other conversions convert nothing. When the content encoder physical bottleneck is too wide, the rhythm-only conversion still converts rhythm. Notably, the timbre-only conversion still converts timbre to some degree, possibly due to the random resampling operation of the content encoder. These results verify our theoretical explanation of SpeechSplit.

6.5.6 Does Random Resampling Remove All Rhythm?

In Figure 6.2 and Section 6.4.4, we assume that the random resampling only contaminates rhythm information, but does not completely remove it. To verify this assumption, we train a single autoencoder for speech, where the encoder and decoder are the SpeechSplit content encoder and decoder respectively. If randomly resampling only removes a portion of the rhythm information, the output reconstruction can still roughly temporally aligned with the ground truth speech. Otherwise, the reconstruction will be completely misaligned.

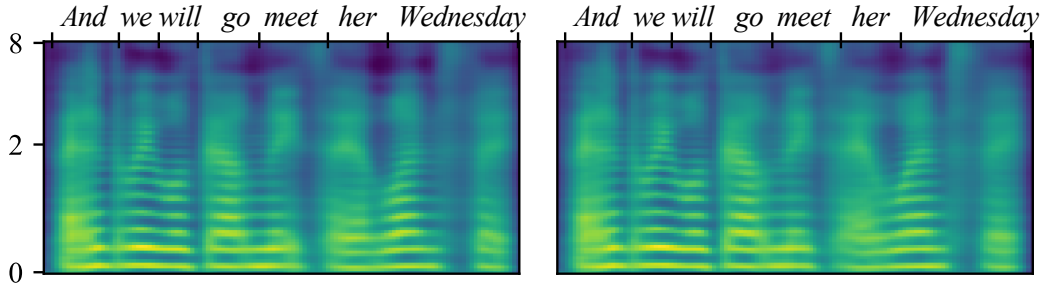


Figure 6.9: Reconstructed speech produced by AutoVC with a random resampling module. The ground truth speech for the left column is in Figure 6.6. The word boundaries and labels are copied from that of the ground truth.

Figure 6.9 shows two reconstruction results with different randomly drawn resampling factors, whose ground truth utterances are both the top-left panel of Figure 6.6. To assist our judgment of the alignment, we directly copy the word boundaries and labels from the ground truth. As can be observed, the two reconstructions are very alike, even though their random resampling factors are different. Furthermore, both reconstructions can recover the ground

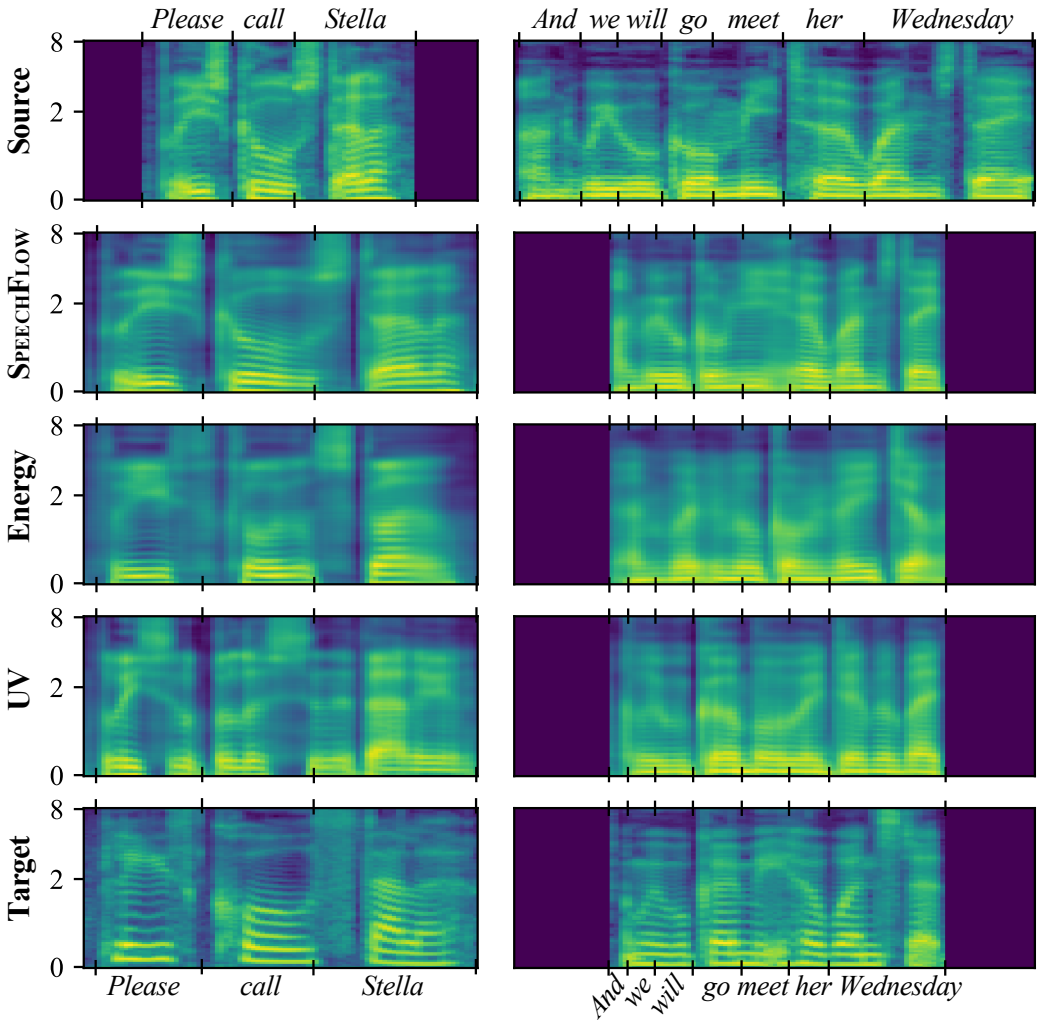


Figure 6.10: Rhythm-only conversion using the rhythm feature in SpeechSplit (second row) compared with that using candidate rhythm features, including short-time energy (third row) and UV label (fourth row).

truth speech decently, only with some minor blurring, which verifies that random resampling performs an *incomplete disentanglement* of rhythm. In other words, SpeechSplit shows that we can build a complete disentanglement mechanism even if we only have a partial disentanglement technique.

6.5.7 Do Rhythm Labels Exist?

In Section 5.1, we have discussed that one motivation for designing SpeechSplit is that rhythm labels are not directly available. If they were, the rhythm aspect could be disentangled in much simpler ways. In this section,

we would like to explore if there exist any rhythm labels.

We have identified two promising candidate rhythm labels, short-time energy and unvoiced-voiced (UV) label. The short-time energy is computed by taking the moving average of the squared waveform. The UV labels are derived from pitch contour, which equals one if the corresponding frame is voiced, and zero otherwise. Both candidates are informative of the syllable boundaries, and neither contains other information such as content and pitch. To test if these candidates are equally effective as the SpeechSplit rhythm encoder, we train two variants of SpeechSplit, one replacing the rhythm code with the short-time energy, and the other with the UV label. We then perform the rhythm-only conversion using SpeechSplit and the two variants, by replacing the rhythm code/label with that of the target speech. If the candidates are effective, the corresponding rhythm-only conversions should be successful.

Figure 6.10 shows the rhythm-only conversion results on two utterances, *"Please call Stella"* and *"And we will go meet her Wednesday"*, produced by these three algorithms. At first glance, all the conversion results are temporally aligned with the target speech, which seems to suggest that the rhythm aspect has been successfully converted. However, a close inspection into the formant structure of the candidate conversion results reveals that the content within each syllable is completely incorrect.

With the "fill in the blank" perspective discussed in Section 6.5.4, we can better understand why the candidate rhythm labels fail. Both candidates can accurately provide the temporal information of the syllable boundaries, and thus the blanks are correctly located in time. However, the candidates fail to provide the anchor information of what to fill in each blank, and that is why the conversion algorithms put the wrong content in the blanks. In summary, obtaining a rhythm label is a nontrivial task, because the rhythm label should contain some anchor information to associate each syllable with the correct content, while excluding excessive content to ensure content disentanglement. SpeechSplit, with a triple information bottleneck design, manages to obtain such an effective rhythm code, which contributes to a successful rhythm conversion.

6.5.8 Additional Conversion Spectrograms

In Figure 6.11, we augment the spectrogram visualization results in Section 6.5.2 (Figure 6.4) with two additional utterances, *"One showing mainly red and yellow"* and *"Six spoons of fresh snow peas"*, and with all the conversion types (not just the single-aspect conversions) displayed. Consistent with the results shown in Section 6.5.2, these additional results show that SpeechSplit can successfully convert the intended aspects to match those of the target speech, while keeping the remaining aspects matching the source speech. Remarkably, when all three aspects are converted, the converted speech becomes very similar to the target speech.

6.6 Summary

We have demonstrated that SpeechSplit has powerful disentanglement capabilities by having multiple intricately designed information bottlenecks. There are three takeaways. First, we have shown that the physical dimension of the hidden representations can effectively limit the information flow. Second, we have verified that when information bottleneck is binding, neural autoencoder will only pass the information that other channels cannot provide. Third, even if we only have a partial disentanglement algorithm, *e.g.* the random resampling, we can still design a complete disentanglement algorithm by having multiple channels with different information bottleneck. These intriguing observations inspire a generic approach to disentanglement.

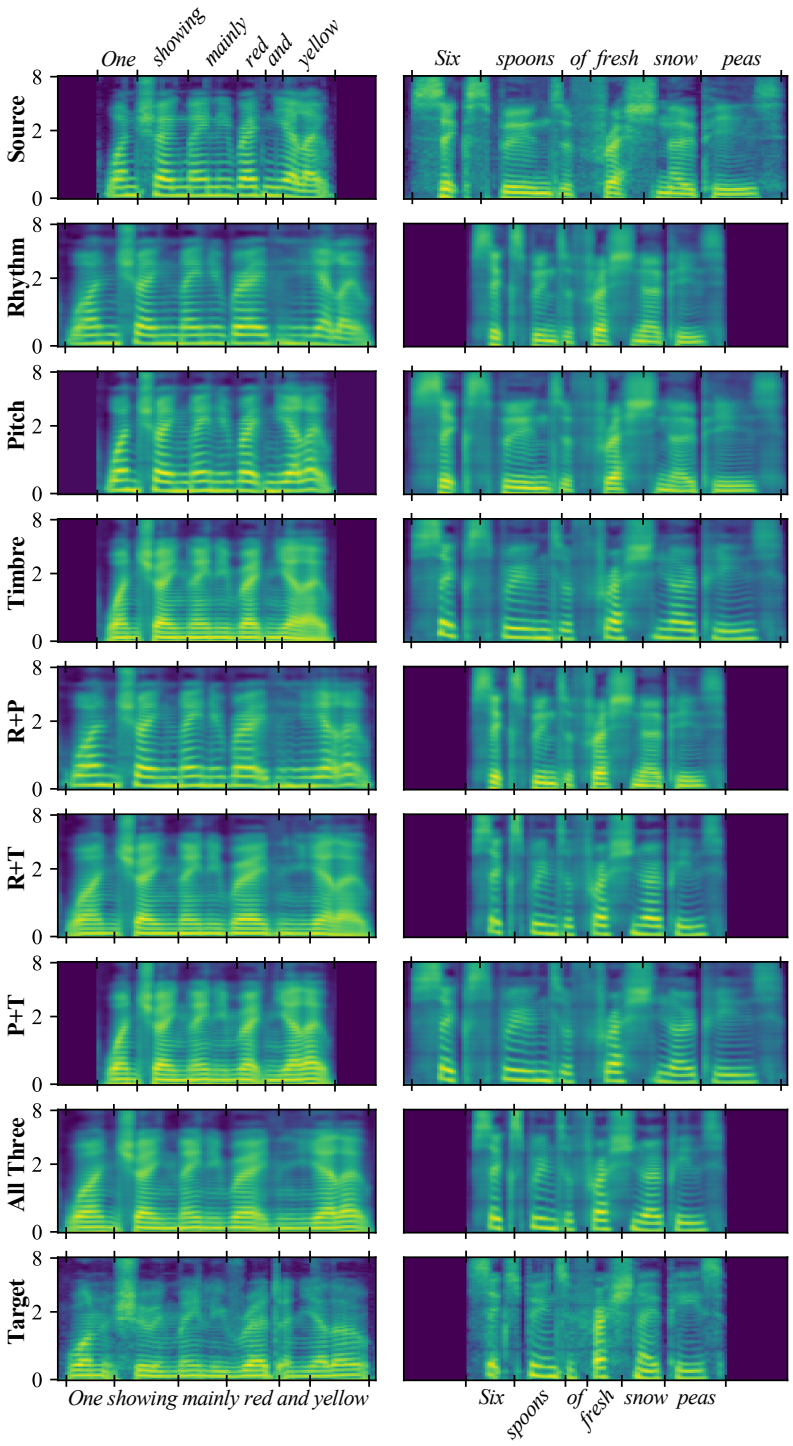


Figure 6.11: Spectorgrams of aspect-specific conversion results on two utterances, *"One showing mainly red and yellow"* (left) and *"Six spoons of fresh snow peas"* (right). R+P denotes rhythm+pitch conversion; R+T denotes rhythm+timbre conversion; P+T denotes pitch+timbre conversion.

CHAPTER 7

DISCUSSION

We have introduced four research attempts that use generative models for speech editing. Now we can reconsider the problem raised in Chapter 1, that is how can generative models help in naturalness for speech editing.

7.1 Regularized Speech Enhancement

The first way is to regularize the speech editing process by defining the sample space of natural speech. In particular, BaWN and DeepBeam essentially regularize the inference on clean speech using generative models of speech. In other words, the generative models make the enhanced speech natural by forcing it to fall in the sample space of natural speech. BaWN and DeepBeam share two major benefits of using generative models. First, they both benefit from using the generative models by focusing on modeling clean speech distribution. As a speech prior used in a Bayesian framework for BaWN, the generative model greatly improves the ability of the enhancement model to generalize to unseen types of noise. Similarly, as a clean speech predictor for DeepBeam, the generative model alleviates the beamformer from the burden of characterizing sensor positions and interference, which are difficult to determine for ad-hoc microphone arrays. Second, they both operate directly on waveform without needing to revert time-frequency representations to waveform, which is prone to phase distortion artifacts. However, the generative models used in BaWN and DeepBeam differ in terms of the distributions being modeled, the function of the model in the framework, and the form of regularization. The generative model used in BaWN only models clean speech waveform and acts as the clean speech prior model in a Bayesian framework. However, the generative model used in DeepBeam models the conditional distribution of clean speech waveform given the noisy

speech waveform, which not only defines the clean speech sample space but also enhances speech by removing noise from its input. The generative model used in BaWN regularizes the enhancement model output by multiplying the prior distribution with the likelihood function. However, the generative model used in DeepBeam predicts clean speech, which is set as the target of the beamformer by minimizing the mean-squared-error between the clean speech target and the beamformer output.

7.2 Disentanglement for Speech Editing

Unlike the first way, which improves the naturalness of edited speech as discussed in Section 7.1, the second way is to directly modify the components of hierarchical generative models. When the model only learns to produce natural speech by modeling different components of speech, the output will always be natural speech regardless of the inputs. In particular, AutoVC learns to generate natural speech from timbre and content, and SpeechSplit learns to generate natural speech from timbre, content, rhythm, and pitch. AutoVC can only convert timbre by modifying the speaker identity label, but SpeechSplit can convert each of different combinations of timbre, pitch, and rhythm. The basic building block of AutoVC and SpeechSplit is autoencoder, where the latter adds two additional autoencoders to disentangle rhythm and pitch respectively. As an extension of AutoVC, SpeechSplit is based on the same principle of learning disentangled representations of speech as AutoVC. The key to achieve speech disentanglement is constraining the information flow using a carefully designed bottleneck that only passes the desired component of speech. Both AutoVC and SpeechSplit achieved unsupervised speech disentanglement by training only on the self-reconstruction loss. Although SpeechSplit is able to convert rhythm, it requires the target utterance to have the same or very similar content as the source utterance, which is not very flexible in practice. This means that the content information and rhythm information are still correlated to some degree. A complete disentanglement of rhythm still remains challenging and needs to be explored in future research.

7.3 WaveNet for Speech Editing

As one of the major breakthroughs in audio waveform modeling in recent years, WaveNet is used in all four research works for the generation of speech. It enables BaWN and DeepBeam to directly operate on speech waveform without needing to convert between waveform and time-frequency representations of speech, which may suffer from phase distortion artifacts. In BaWN, it is used as the basic building block for both the prior model and the likelihood model. The prior model is a causal WaveNet that autoregressively predicts the current sample based on the previously predicted samples. The likelihood model is a non-causal WaveNet that predicts the current sample based on the previous, current, and future input samples. The final output sample distribution is the product of the prior distribution and the likelihood function. Due to the autoregressive nature of the prior model, the current final output sample depends on the previously predicted final output samples during both training and testing. During training, this time dependency significantly slows down the training speed, which needs to be addressed in future research. The non-causal WaveNet is also used in DeepBeam as the enhancement model that predicts the clean speech samples from the noisy speech input samples. For the non-causal WaveNet, the inference on clean speech can be done in a feed-forward fashion because the output sample of the non-causal WaveNet does not depend on the previous outputs. Instead of being the fundamental building block in BaWN and DeepBeam, the WaveNet is used as vocoder in AutoVC and SpeechSplit. The models of these two works directly operate on spectrograms. In order to be able to listen to the results, the spectrograms need to be converted to waveforms, which is a task that WaveNet can accomplish. The same WaveNet vocoder is used in both AutoVC and SpeechSplit, where the vocoder is pre-trained using pairs of spectrogram and waveform from 109 speakers by conditioning the WaveNet on the spectrograms as described in Section 2.3.2. During inference, the WaveNet vocoder autoregressively predicts the current speech sample based on the current frame of the conditioned spectrogram and the previously predicted speech samples. The WaveNet vocoder's autoregressive generation of speech is slow, but we found it generalizes well to unseen speakers and produces natural-sounding speech.

7.4 Methods of Speech Disentanglement

In the second way of using generative models for speech editing, we have seen different methods of speech disentanglement using different ways to construct information-constraining bottlenecks.

We have introduced two methods of speech disentanglement using autoencoders. If there is a label for the information to be disentangled, we can directly supply the label to the decoder and tune the bottleneck dimensions to disentangle that information. In AutoVC, the speaker identity label is used as the label of timbre because there is a one-to-one mapping between speaker identity and timbre. Since the decoder has access to the timbre information, the timbre information is disentangled if the bottleneck dimension is small enough. However, if there is no label for the information to be disentangled, we can corrupt that information and supply the full information to the decoder. In SpeechSplit, since there is no label for the rhythm information, we simultaneously supply the input speech with corrupted rhythm and the original input speech to the decoder through two encoders respectively. In this way, only the encoder with original input speech can provide the rhythm information, and the other information can be provided through the other encoder. Although the corrupted rhythm information still reaches the decoder, the decoder does not use it because it cannot be used to reconstruct the original input speech. In this way, rhythm information is disentangled.

Depending on the two methods of speech disentanglement, there are two ways of constructing the information-constraining bottleneck. In AutoVC, the information flow is constrained by the dimensions of the encoder's output layer. The dimensions of the encoder's output layer in an autoencoder are typically small in order to learn some compact representation of the input. In AutoVC, the output of the encoder is downsampled in both the frequency dimension and time dimension in order to squeeze out the timbre information and only keep the content information. Besides dimension constraints, in SpeechSplit, the information flow is constrained by corruption when it cannot be constrained using dimension constraints. In particular, the rhythm information is corrupted by randomly compressing or stretching the input speech. Depending on the type of information to be disentangled, other methods such as dropout and quantization can also be used to construct information-constraining bottlenecks.

CHAPTER 8

CONCLUSION

This thesis introduces two ways of using generative models for speech editing tasks, where speech naturalness is very important. The first way is to regularize the speech editing process by defining the sample space of natural speech, and the second way is by permitting the separable modification of components of hierarchical speech generative models in order to modify specified components of natural speech. For the first method, Chapter 3 and Chapter 4 are successful examples of using the WaveNet for regularizing the speech enhancement process. Specifically, in Chapter 3, a beamformer is guided by the output of a WaveNet-based speech model to perform speech beamforming for a multi-channel ad-hoc microphone array. In Chapter 4, a prior model is trained for natural speech using WaveNet and incorporated in a Bayesian framework to regularize a single-channel speech enhancement model. In these applications, the WaveNet-based speech model defines the natural speech sample space and thus can project any unnatural output of the enhancement model into the natural speech sample space to make it natural. In addition, for the second method, in Chapter 5 and Chapter 6, we are able to convert different aspects of the speech by disentangling these aspects using autoencoders with information constraining bottlenecks. Specifically, in Chapter 5, we can modify the speaker identity of the speech by disentangling content and timbre in an unsupervised manner. In Chapter 6, we can convert not only timbre but also pitch and rhythm separately by disentangling content, timbre, pitch, and rhythm without using any text labels. In these applications, since the model only learns to produce natural speech by modeling different components of speech, the output will always be natural speech regardless of the inputs.

REFERENCES

[1] Y. Zhang, D. Florêncio, and M. Hasegawa-Johnson, "Glottal model based speech beamforming for ad-hoc microphone arrays," *Interspeech*, pp. 2675–2679, 2017.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] D. A. Reynolds, "Gaussian mixture models," *Encyclopedia of Biometrics*, vol. 741, 2009.

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

[5] T. F. Quatieri, *Discrete-Time Speech Signal Processing: Principles and Practice*. Pearson Education India, 2006.

[6] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" in *International Conference on Learning Representations*, 2017.

[7] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[8] S. Darlington, "Linear least-squares smoothing and prediction, with applications," *Bell System Technical Journal*, vol. 37, no. 5, pp. 1221–1294, 1958.

[9] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408–1418, 1969.

[10] O. L. Frost, "An algorithm for linearly constrained adaptive array processing," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 926–935, 1972.

[11] E. A. Habets, J. Benesty, S. Gannot, and I. Cohen, "The MVDR beamformer for speech enhancement," in *Speech Processing in Modern Communication*. Springer, 2010, pp. 225–254.

[12] U. H. Yapanel and J. H. Hansen, "A new perceptually motivated MVDR-based acoustic front-end (PMVDR) for robust automatic speech recognition," *Speech Communication*, vol. 50, no. 2, pp. 142–152, 2008.

[13] D. E. Ba, D. Florêncio, and C. Zhang, "Enhanced MVDR beamforming for arrays of directional microphones," in *2007 IEEE International Conference on Multimedia and Expo*. IEEE, 2007, pp. 1307–1310.

[14] M. Er and A. Cantoni, "Derivative constraints for broad-band element space antenna array processors," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, no. 6, pp. 1378–1393, 1983.

[15] L. Griffiths and C. Jim, "An alternative approach to linearly constrained adaptive beamforming," *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 1, pp. 27–34, 1982.

[16] S. Affes and Y. Grenier, "A source subspace tracking array of microphones for double talk situations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2. IEEE, 1996, pp. 909–912.

[17] S. Gannot, D. Burshtein, and E. Weinstein, "Signal enhancement using beamforming and nonstationarity with applications to speech," *IEEE Transactions on Signal Processing*, vol. 49, no. 8, pp. 1614–1626, 2001.

[18] I. Himawan, I. McCowan, and S. Sridharan, "Clustered blind beamforming from ad-hoc microphone arrays," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 661–676, 2011.

[19] J. Bitzer, K. U. Simmer, and K.-D. Kammeyer, "Theoretical noise reduction limits of the generalized sidelobe canceller (GSC) for speech enhancement," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5. IEEE, 1999, pp. 2965–2968.

[20] B. W. Gillespie, H. S. Malvar, and D. A. Florêncio, "Speech dereverberation via maximum-kurtosis subband adaptive filtering," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 6, 2001, pp. 3701–3704.

[21] B. Yegnanarayana and P. S. Murthy, "Enhancement of reverberant speech using LP residual signal," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 3, pp. 267–281, 2000.

[22] T. Kim, H. T. Attias, S.-Y. Lee, and T.-W. Lee, "Blind source separation exploiting higher-order frequency dependencies," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 70–79, 2007.

[23] K. Kumatani, J. McDonough, B. Rauch, D. Klakow, P. N. Garner, and W. Li, "Beamforming with a maximum negentropy criterion," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 5, pp. 994–1008, 2009.

[24] T. Sreenivas and P. Kirnapure, "Codebook constrained Wiener filtering for speech enhancement," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 383–389, 1996.

[25] M. El-Fattah, M. Dessouky, A. Abbas, S. Diab, E.-S. El-Rabaie, W. Al-Nuaimy, S. Alshebeili, and F. Abd El-Samie, "Speech enhancement with an adaptive Wiener filter," *International Journal of Speech Technology*, vol. 17, pp. 53–64, 03 2014.

[26] B. Xia and C. Bao, "Wiener filtering based speech enhancement with weighted denoising auto-encoder and noise classification," *Speech Communication*, vol. 60, pp. 13–29, 2014.

[27] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 5, pp. 504–512, 2001.

[28] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 2, pp. 113–120, 1979.

[29] N. Upadhyay and A. Karmakar, "The spectral subtractive-type algorithms for enhancing speech in noisy environments," in *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2012, pp. 841–847.

[30] T. Takiguchi and Y. Ariki, "PCA-based speech enhancement for distorted speech recognition," *Journal of Multimedia*, vol. 2, no. 5, 2007.

[31] K. W. Wilson, B. Raj, P. Smaragdis, and A. Divakaran, "Speech denoising using nonnegative matrix factorization with priors," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2008, pp. 4029–4032.

[32] N. Mohammadiha, P. Smaragdis, and A. Leijon, "Supervised and unsupervised speech enhancement using nonnegative matrix factorization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2140–2151, 2013.

[33] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65–68, 2014.

[34] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.

[35] K. Han, Y. Wang, D. Wang, W. S. Woods, I. Merks, and T. Zhang, "Learning spectral mapping for speech dereverberation and denoising," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 6, pp. 982–992, 2015.

[36] D. Liu, P. Smaragdis, and M. Kim, "Experiments on deep learning for speech denoising," in *Interspeech*, 2014, pp. 2685–2689.

[37] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder," in *Interspeech*, 2013, pp. 436–440.

[38] A. Kumar and D. Florêncio, "Speech enhancement in multiple-noise conditions using deep neural networks," in *Interspeech*, 2016, pp. 3738–3742.

[39] Y. Wang, A. Narayanan, and D. Wang, "On training targets for supervised speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1849–1858, 2014.

[40] K. Han and D. Wang, "A classification based approach to speech segregation," *The Journal of the Acoustical Society of America*, vol. 132, no. 5, pp. 3475–3483, 2012.

[41] A. Narayanan and D. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 7092–7096.

[42] Y. Zhao, D. Wang, I. Merks, and T. Zhang, "DNN-based enhancement of noisy and reverberant speech," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6525–6529.

[43] D. S. Williamson, Y. Wang, and D. Wang, "Complex ratio masking for monaural speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 3, pp. 483–492, 2015.

[44] Y. Ephraim, "A Bayesian estimation approach for speech enhancement using hidden Markov models," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 725–735, 1992.

[45] H. Sameti, H. Sheikhzadeh, L. Deng, and R. L. Brennan, "HMM-based strategies for enhancement of speech signals embedded in nonstationary noise," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 5, pp. 445–455, 1998.

[46] D. Y. Zhao and W. B. Kleijn, "HMM-based gain modeling for enhancement of speech in noise," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 882–892, 2007.

[47] A. Kundu, S. Chatterjee, A. S. Murthy, and T. Sreenivas, "GMM based Bayesian approach to speech enhancement in signal/transform domain," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 4893–4896.

[48] R. Martin and C. Breithaupt, "Speech enhancement in the DFT domain using Laplacian speech priors," in *International Workshop on Acoustic Echo and Noise Control (IWAENC)*, vol. 3, 2003, pp. 87–90.

[49] T. Lotter and P. Vary, "Speech enhancement by MAP spectral amplitude estimation using a super-Gaussian speech model," *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 1110–1126, 2005.

[50] D. Kitamura, N. Ono, H. Sawada, H. Kameoka, and H. Saruwatari, "Determined blind source separation unifying independent vector analysis and nonnegative matrix factorization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1626–1641, 2016.

[51] P. Mowlaee and J. Kulmer, "Harmonic phase estimation in single-channel speech enhancement using phase decomposition and snr information," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 9, pp. 1521–1532, 2015.

[52] J. Kulmer and P. Mowlaee, "Phase estimation in single channel speech enhancement using phase decomposition," *IEEE Signal Processing Letters*, vol. 22, no. 5, pp. 598–602, 2014.

[53] T. Gerkmann, M. Krawczyk, and R. Rehr, "Phase estimation in speech enhancement–Unimportant, important, or impossible?" in *2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel*. IEEE, 2012, pp. 1–5.

[54] A. Kain and M. W. Macon, "Design and evaluation of a voice conversion algorithm based on spectral envelope mapping and residual prediction," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2. IEEE, 2001, pp. 813–816.

[55] A. R. Toth and A. W. Black, "Using articulatory position data in voice transformation," in *SSW*, 2007, pp. 182–187.

[56] T. Toda, A. W. Black, and K. Tokuda, "Mapping from articulatory movements to vocal tract spectrum with gaussian mixture model for articulatory speech synthesis," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[57] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from non-parallel corpora using variational auto-encoder," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2016, pp. 1–6.

[58] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "StarGAN-VC: Non-parallel many-to-many voice conversion using star generative adversarial networks," in *IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 266–273.

[59] A. Mouchtaris, J. Van der Spiegel, and P. Mueller, "Nonparallel training for voice conversion based on a parameter adaptation approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 952–963, 2006.

[60] Y. Saito, Y. Ijima, K. Nishida, and S. Takamichi, "Non-parallel voice conversion using variational autoencoders conditioned by phonetic posteriorgrams and d-vectors," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5274–5278.

[61] 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)*. IEEE, 2016, pp. 1–6.

[62] D. Saito, K. Yamamoto, N. Minematsu, and K. Hirose, "One-to-many voice conversion based on tensor representation of speaker space," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[63] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "ACVAE-VC: Non-parallel voice conversion with auxiliary classifier variational autoencoder," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 9, pp. 1432–1443, 2019.

[64] E. Zetterholm, "Same speaker–different voices. a study of one impersonator and some of his different imitations," in *Proceedings of the 11th Australian International Conference on Speech Science & Technology*, 2006, pp. 70–75.

[65] M. Barlow and M. Wagner, "Prosody as a basis for determining speaker characteristics," in *Proceedings of the Australian International Conference on Speech Science and Technology*, 1988, pp. 80–85.

[66] J. Lorenzo-Trueba, J. Yamagishi, T. Toda, D. Saito, F. Villavicencio, T. Kinnunen, and Z. Ling, "The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 195–202.

[67] T. Toda, L.-H. Chen, D. Saito, F. Villavicencio, M. Wester, Z. Wu, and J. Yamagishi, "The voice conversion challenge 2016," in *Interspeech*, 2016, pp. 1632–1636.

[68] Y. Stylianou, "Voice transformation: A survey," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2009, pp. 3585–3588.

[69] Y. Stylianou, O. Cappé, and E. Moulines, "Continuous probabilistic transform for voice conversion," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, pp. 131–142, 1998.

[70] D. T. Chappell and J. H. Hansen, "Speaker-specific pitch contour modeling and modification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2. IEEE, 1998, pp. 885–888.

[71] B. Gillett and S. King, "Transforming F0 contours," in *Interspeech*, 2003.

[72] H. Ye and S. Young, "High quality voice morphing," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1. IEEE, 2004, pp. I–9.

[73] N. Q. Hy, S. W. Lee, X. Tian, M. Dong, and C. E. Siong, "High quality voice conversion using prosodic and high-resolution spectral features," *Multimedia Tools and Applications*, vol. 75, pp. 5265–5285, 2015.

[74] Z. Wu, T. Kinnunen, C. E. Siong, and H. Li, "Text-independent F0 transformation with non-parallel data for voice conversion," in *Interspeech*, 2010.

[75] G. K. Anumanchipalli, L. C. Oliveira, and A. W. Black, "A style capturing approach to F0 transformation in voice conversion," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6915–6919.

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

[77] B. Şişman, H. Li, and K. C. Tan, "Transformation of prosody in voice conversion," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2017, pp. 1537–1546.

[78] R. Srikanth, B. Bajibabu, and K. Prahallad, "Duration modelling in voice conversion using artificial neural networks," in *2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2012, pp. 556–559.

[79] C.-H. Wu, C.-C. Hsia, T.-H. Liu, and J.-F. Wang, "Voice conversion using duration-embedded bi-hmms for expressive speech synthesis," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1109–1116, 2006.

[80] K. Yutani, Y. Uto, Y. Nankaku, T. Toda, and K. Tokuda, "Simultaneous conversion of duration and spectrum based on statistical models including time-sequence matching," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[81] S. K. Gaikwad, B. W. Gawali, and P. Yannawar, "A review on speech recognition technique," *International Journal of Computer Applications*, vol. 10, no. 3, pp. 16–24, 2010.

[82] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 308–311, 2006.

[83] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[84] M. Stewart, "Comprehensive introduction to autoencoders," Apr 2019. [Online]. Available: https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

[85] X. Feng, Y. Zhang, and J. Glass, "Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1759–1763.

[86] T. Ishii, H. Komiyama, T. Shinozaki, Y. Horiuchi, and S. Kuroiwa, "Reverberant speech recognition based on denoising autoencoder," in *Interspeech*, 2013, pp. 3512–3516.

[87] M. Blaauw and J. Bonada, "Modeling and transforming speech using variational autoencoders," in *Interspeech*, 2016, pp. 1770–1774.

[88] K. Akuzawa, Y. Iwasawa, and Y. Matsuo, "Expressive speech synthesis via modeling expressions with variational autoencoder," in *Interspeech*, 2018, pp. 3067–3071.

[89] W.-N. Hsu, Y. Zhang, and J. Glass, "Learning latent representations for speech generation and transformation," in *Interspeech*, 2017, pp. 1273–1277.

[90] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," in *Advances in Neural Information Processing Systems*, 2016, pp. 2352–2360.

[91] W.-C. Huang, Y.-C. Wu, H.-T. Hwang, P. L. Tobing, T. Hayashi, K. Kobayashi, T. Toda, Y. Tsao, and H.-M. Wang, "Refined wavenet vocoder for variational autoencoder based voice conversion," in *27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[92] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan et al., "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[93] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[94] J. Feng, X. Feng, J. Chen, X. Cao, X. Zhang, L. Jiao, and T. Yu, "Generative adversarial networks based on collaborative learning and attention mechanism for hyperspectral image classification," *Remote Sensing*, vol. 12, no. 7, p. 1149, 2020.

[95] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[96] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, vol. 70, 2017, pp. 214–223.

[97] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.

[98] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[99] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[100] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2018.

[101] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[102] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[103] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5907–5915.

[104] S. Pascual, A. Bonafonte, and J. Serrà, "SEGAN: Speech enhancement generative adversarial network," *Interspeech*, pp. 3642–3646, 2017.

[105] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *International Conference on Learning Representations*, 2018.

[106] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 215–10 224.

[107] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3617–3621.

[108] M. Brandstein and D. Ward, *Microphone Arrays: Signal Processing Techniques and Applications*. Springer Science & Business Media, 2013.

[109] S. Markovich-Golan, A. Bertrand, M. Moonen, and S. Gannot, "Optimal distributed minimum-variance beamforming approaches for speech enhancement in wireless acoustic sensor networks," *Signal Processing*, vol. 107, pp. 4–20, 2015.

[110] J. Chen, Y. Wang, S. E. Yoho, D. Wang, and E. W. Healy, "Large-scale training to increase speech intelligibility for hearing-impaired listeners in novel noises," *The Journal of the Acoustical Society of America*, vol. 139, no. 5, pp. 2604–2612, 2016.

[111] J. Chen and D. Wang, "Long short-term memory for speaker generalization in supervised speech separation," in *Interspeech*, 2016, pp. 3314–3318.

[112] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 1562–1566.

[113] F. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2014, pp. 577–581.

[114] K. Qian, Y. Zhang, S. Chang, X. Yang, D. Florêncio, and M. Hasegawa-Johnson, "Speech enhancement using Bayesian Wavenet," *Interspeech*, pp. 2013–2017, 2017.

[115] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5069–5073.

[116] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press, Cambridge, MA, 1949.

[117] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[118] E. A. Lehmann and A. M. Johansson, "Diffuse reverberation model for efficient image-source simulation of room impulse responses," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1429–1439, 2010.

[119] J. Yamagishi, "English multi-speaker corpus for CSTR voice cloning toolkit," http://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html.

[120] "Freesound," https://freesound.org/, 2015.

[121] G. Hu, "100 nonspeech sounds," http://web.cse.ohio-state.edu/pnl/corpus/HuNonspeech/HuCorpus.html, 2015.

[122] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM. NIST speech disc 1-1.1," *NASA STI/Recon Technical Report N*, vol. 93, 1993.

[123] "FreeSFX," http://www.freesfx.co.uk/, 2017.

[124] F. Ribeiro, D. Florêncio, C. Zhang, and M. Seltzer, "CrowdMOS: An approach for crowdsourcing mean opinion score studies," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 2416–2419.

[125] Z. Ou and Y. Zhang, "Probabilistic acoustic tube: A probabilistic generative model of speech for speech analysis/synthesis," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012, pp. 841–849.

[126] ITU, "Pulse code modulation (PCM) of voice frequencies," in *Recommendation, CCITT*, 1988.

[127] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelCNN decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[128] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[129] T. L. Paine, P. Khorrami, S. Chang, Y. Zhang, P. Ramachandran, M. A. Hasegawa-Johnson, and T. S. Huang, "Fast wavenet generation algorithm," *arXiv preprint arXiv:1611.09482*, 2016.

[130] S. King and V. Karaiskos, "The Blizzard Challenge 2013," *Proc. Blizzard Workshop*, 2013.

[131] C. Févotte, R. Gribonval, and E. Vincent, "BSS_EVAL toolbox user guide–revision 2.0," 2005.

[132] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *Stat*, vol. 1050, p. 10, 2014.

[133] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.

[134] Y. C. Subakan and P. Smaragdis, "Generative adversarial source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 26–30.

[135] Z.-C. Fan, Y.-L. Lai, and J.-S. R. Jang, "SVSGAN: Singing voice separation via generative adversarial network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 726–730.

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

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

[138] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from unaligned corpora using variational autoencoding Wasserstein generative adversarial networks," in *Interspeech*, 2017, pp. 3364–3368.

[139] W.-C. Huang, H.-T. Hwang, Y.-H. Peng, Y. Tsao, and H.-M. Wang, "Voice conversion based on cross-domain features using variational auto encoders," in *11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2018, pp. 51–55.

[140] J.-C. Chou, C.-C. Yeh, H.-Y. Lee, and L.-S. Lee, "Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations," *Interspeech*, pp. 501–505, 2018.

[141] T. Kaneko and H. Kameoka, "Parallel-data-free voice conversion using cycle-consistent adversarial networks," *arXiv preprint arXiv:1711.11293*, 2017.

[142] F. Fang, J. Yamagishi, I. Echizen, and J. Lorenzo-Trueba, "High-quality nonparallel voice conversion based on cycle-consistent adversarial network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5279–5283.

[143] F.-L. Xie, F. K. Soong, and H. Li, "A KL divergence and DNN-based approach to voice conversion without parallel training sentences," in *Interspeech*, 2016, pp. 287–291.

[144] Y. Gao, R. Singh, and B. Raj, "Voice impersonation using generative adversarial networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2506–2510.

[145] C. Atalla, B. Tam, A. Song, and G. Cottrell, "Look ma, no GANs! image transformation with modifAE," 2019. [Online]. Available: https://openreview.net/forum?id=B1ethsR9Ym

[146] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. Lopez Moreno, Y. Wu et al., "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," *Advances in Neural Information Processing Systems*, vol. 31, pp. 4480–4490, 2018.

[147] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Interspeech*, 2017.

[148] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 5206–5210.

[149] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR abs/1609.03499*, 2016.

[150] C. Veaux, J. Yamagishi, K. MacDonald et al., "Superseded-CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit," 2016.

[151] M. Wester, Z. Wu, and J. Yamagishi, "Analysis of the voice conversion challenge 2016 evaluation results," in *Interspeech*, 2016, pp. 1637–1641.

[152] S. Ö. Arık, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman et al., "Deep voice: Real-time neural text-to-speech," in *International Conference on Machine Learning*, 2017, pp. 195–204.

[153] J.-C. Chou and H.-Y. Lee, "One-shot voice conversion by separating speaker and content representations with instance normalization," *Proc. Interspeech 2019*, pp. 664–668, 2019.

[154] S. Shechtman and A. Sorin, "Sequence to sequence neural speech synthesis with prosody modification capabilities," in *Proc. 10th ISCA Speech Synthesis Workshop*, 2019, pp. 275–280.

[155] M. D. Pell, A. Jaywant, L. Monetta, and S. A. Kotz, "Emotional speech processing: Disentangling the effects of prosody and semantic cues," *Cognition & Emotion*, vol. 25, no. 5, pp. 834–853, 2011.

[156] M. Schroeder and B. Atal, "Code-excited linear prediction (CELP): High-quality speech at very low bit rates," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 10. IEEE, 1985, pp. 937–940.

[157] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, "AutoVC: Zero-shot voice style transfer with only autoencoder loss," in *International Conference on Machine Learning*, 2019, pp. 5210–5219.

[158] A. Polyak and L. Wolf, "Attention-based WaveNet autoencoder for universal voice conversion," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6800–6804.

[159] B. Atal and M. Schroeder, "Predictive coding of speech signals and subjective error criteria," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 3, pp. 247–254, 1979.

[160] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern Recognition and Artificial Intelligence*, vol. 116, pp. 374–388, 1976.

[161] H. Kawahara, M. Morise, T. Takahashi, R. Nisimura, T. Irino, and H. Banno, "Tandem-STRAIGHT: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, F0, and aperiodicity estimation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2008, pp. 3933–3936.

[162] Y. Zhang, Z. Ou, and M. Hasegawa-Johnson, "Improvement of probabilistic acoustic tube model for speech decomposition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 7929–7933.

[163] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader networks: Manipulating images by sliding attributes," in *Advances in Neural Information Processing Systems*, 2017, pp. 5967–5976.

[164] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8789–8797.

[165] W.-C. Huang, H. Luo, H.-T. Hwang, C.-C. Lo, Y.-H. Peng, Y. Tsao, and H.-M. Wang, "Unsupervised representation disentanglement using cross domain features and adversarial learning in variational autoencoder based voice conversion," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

[166] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, "StarGAN-VC2: Rethinking conditional methods for StarGAN-based voice conversion," *Proc. Interspeech 2019*, pp. 679–683, 2019.

[167] K. Qian, Z. Jin, M. Hasegawa-Johnson, and G. J. Mysore, "F0-consistent many-to-many non-parallel voice conversion via conditional autoencoder," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6284–6288.

[168] J. Niwa, T. Yoshimura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Statistical voice conversion based on WaveNet," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5289–5293.

[169] E. Nachmani and L. Wolf, "Unsupervised singing voice conversion," *Proc. Interspeech 2019*, pp. 2583–2587, 2019.

[170] J. Serrà, S. Pascual, and C. S. Perales, "Blow: A single-scale hyperconditioned flow for non-parallel raw-audio voice conversion," in *Advances in Neural Information Processing Systems*, 2019, pp. 6790–6800.

[171] R. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. Weiss, R. Clark, and R. A. Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with Tacotron," in *International Conference on Machine Learning*, 2018, pp. 4693–4702.

[172] R. Valle, J. Li, R. Prenger, and B. Catanzaro, "Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6189–6193.

[173] T. Kenter, V. Wan, C.-A. Chan, R. Clark, and J. Vit, "CHiVE: Varying prosody in speech synthesis with a linguistically driven dynamic hierarchical conditional variational network," in *International Conference on Machine Learning*, 2019, pp. 3331–3340.

[174] F. Biadsy, R. J. Weiss, P. J. Moreno, D. Kanvesky, and Y. Jia, "Parrotron: An end-to-end speech-to-speech conversion model and its applications to hearing-impaired speech and speech separation," *Proc. Interspeech 2019*, pp. 4115–4119, 2019.

[175] Y. Xu and Q. E. Wang, "Pitch targets and their realization: Evidence from Mandarin Chinese," *Speech Communication*, vol. 33, no. 4, pp. 319–337, 2001.

[176] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

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

[178] T. Nakatani, S. Amano, T. Irino, K. Ishizuka, and T. Kondo, "A method for fundamental frequency estimation and voicing decision: Application to infant utterances recorded in real acoustical environments," *Speech Communication*, vol. 50, no. 3, pp. 203–214, 2008.

[179] W. Chu and A. Alwan, "Reducing F0 frame error of F0 tracking algorithms under noisy conditions with an unvoiced/voiced classification frontend," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2009, pp. 3969–3972.