ADVERSARIAL ONE-SHOT VOICE CONVERSION USING DISENTANGLED REPRESENTATIONS

by

Ali Yeşilkanat

B.S., Computer Engineering, Ege University, 2016

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

Graduate Program in Computer Engineering

Boğaziçi University

2020

ADVERSARIAL ONE-SHOT VOICE CONVERSION USING DISENTANGLED REPRESENTATIONS

APPROVED BY:

Prof. Sadık Fikret Gürgen (Thesis Supervisor)	.....
Prof. Murat Saraçlar	.....
Assist. Prof. Tevfik Aytekin	.....

DATE OF APPROVAL: 10.03.2020

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Prof. Sadık Fikret Gürgen, who has given me the opportunity to pursue my master's education and supervised my research. I would also like to express my gratefulness to Assist. Prof. Heysem Kaya, for his expertise and guidance during my studies.

I would also like to thank Prof. Murat Saraçlar and Assist. Prof. Tevfik Aytekin for their valuable contribution and comments on my thesis.

I owe my deepest gratitude to my family, especially my mother, for their endless love and support throughout my life.

Lastly, I would especially like to thank Deniz Engin for motivating me during my hard moments throughout my research. Without her support, I could not have been accomplished this work.

ABSTRACT

ADVERSARIAL ONE-SHOT VOICE CONVERSION USING DISENTANGLED REPRESENTATIONS

In this thesis, a new adversarial one-shot voice conversion (VC) method is introduced by enhancing one of the latest variational autoencoder based one-shot VC methods. The proposed method utilizes acoustic features as Mel-spectrograms and relies on disentangled representations by separating speaker and content representations of the spoken content. An adversarial loss and perceptual loss are combined in order to increase the quality of generated Mel-spectrograms. We train a speaker classifier by utilizing the architecture of a well-known model in the computer vision area, to be able to adapt perceptual loss during the training of the VC model. We conduct experiments on the Voice Cloning Toolkit dataset and evaluate the proposed approach in terms of Global Variance and MOSNet, a humanoid opinion score simulator. Experimental results indicate that our approach improves VC quality remarkably.

ÖZET

ÇÖZÜLMÜŞ GÖSTERİMLERİ KULLARARAK TEK ÖRNEKLE ÇEKİŞMELİ SES DÖNÜŞÜMÜ

Bu tezde, en yeni varyasyonel özkodlayıcı tabanlı tek örnekli ses dönüşümü yöntemlerinden biri geliştirilerek yeni bir ses dönüştürme yöntemi tanıtılmıştır. Önerilen yöntem, akustik öznitelikler olarak Mel-spektrogramları kullanmakta ve konuşulan içeriğin konuşmacı ve içerik gösterimlerini ayırarak çözülmüş gösterimler oluşturmaktadır. Üretilen Mel-spektrogramlarının kalitesini arttırmak için çekişmeli ve algısal kayıplar kullanılmıştır. Ses çevrim modelinin eğitimi sırasında algısal kaybı uyarlayabilmek için bilgisayarlı görme alanında iyi bilinen bir modelin mimarisini kullanarak bir konuşmacı sınıflandırıcısı eğitilmiştir. Voice Cloning Toolkit veri seti üzerinde deneyler yapılmış, global varyans ve insansı bir yorum simülatörü olan MOSNet açısından değerlendirilmiştir. Deneysel sonuçlar, çalışmamızın referans aldığımız ses dönüşüm yönteminin ses çevrim kalitesini önemli ölçüde artırdığını göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$a \sim p(b)$	a has the probability distribution of $p(b)$
b	Bias
D	Discriminator
$\mathbb{E}$	Expectation
f	Function
G	Generator
I	Identity Matrix
L	Loss
L_{critic}	Critic Loss
L_{dec}	Dec Loss
L_{disc}	Discriminator Loss
L_{KL}	KL Loss
L_{perc}	Perceptual Loss
L_{rec}	Reconstruction Loss
L_{vae}	VAE Loss
$L_{vae+perc}$	Summation of VAE and Perceptual Loss
M	Number of distributions, Number of frequency bands
$\mathcal{N}$	Gaussian Distribution
p(a)	Probability of an event a
$p(a, b)$	Joint probability of a and b
$p(a\|b)$	Conditional Probability of a given b
$\mathbb{R}$	Real Number
T	Transpose
tr	Trace
x	Random variable
w	Weight
W	Weight
z	Latent Variable

β	Affine Parameter
Δ	Difference
ϵ	Error
η	Learning Rate
$\in$	Element of
γ	Affine Parameter
Γ	Gamma Distribution
$\int$	Integral
λ_{rec}	Reconstruction Loss Weight Factor
λ_{perc}	Perceptual Loss Weight Factor
λ_{kl}	KL Loss Weight Factor
λ_{adv}	Adversarial Loss Weight Factor
λ_{kl}	Reconstruction Loss Weight Factor
μ	Mean Vector
∇	Gradient Operator
σ	Standard Deviation
Σ	Covariance Matrix
$\sum$	Summation
θ	Model Weights
θ_D	Discriminator Weights
θ_{E_c}	Content Encoder Weights
$\theta_{E_{Dec}}$	Decoder Weights
θ_{E_s}	Speker Encoders Weights
$\otimes$	Hadamard Product
$*$	Convolution Operator
$\lvert\ \rvert$	Absolute Value
$\lvert\rvert\ \rvert\rvert$	L1 Norm
$\lvert\rvert\ \rvert\rvert_2$	L2 Norm

LIST OF ACRONYMS/ABBREVIATIONS

AE	Autoencoder
ANN	Artificial Neural Network
AdaIN	Adaptive Instance Normalization
BN	Batch Normalization
CNN	Convolutional Neural Network
DBLSTM	Deep Bidirectional Long-short Term Memory
DL	Deep Learning
DNN	Deep Neural Network
DTW	Dynamic Time Warping
EM	Expectation Maximization
FD-PSOLA	Frequency Domain Pitch Synchronous Overlap and Add
FFT	Fast Fourier Transform
FP	Floating Point
GAN	Generative Adversarial Network
GLU	Gated Linear Units
GMCC	Generalized Mel Cepstral Coefficients
GMM	Gaussian Mixture Model
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
GV	Global Variance
HMM	Hidden Markov Model
HNM	Harmonic Plus Noise Model
IN	Instance Normalization
JDGMM	Joint Density Gaussian Mixture Model
KL	Kullback-Leibler
LP-PSOLA	Linear Prediction Pitch Synchronous Overlap and Add
LPCC	Linear Predictive Cepstral Coefficients
LSF	Line Spectral Frequencies

LSTM	Long Short Term Memory
MFCC	Mel Frequency Cepstral Coefficients
MOS	Mean Opinion Score
NN	Neural Network
PSOLA	Pitch Synchronous Overlap and Add
RBM	Restricted Boltzman Machine
RNN	Recursive Neural Network
STFT	Shor Term Fourier Transform
STRAIGHT	Speech Transformation and Representation Using Adaptive Interpolation of Weighted Spectrum
TD-PSOLA	Time Domain Pitch Synchronous Overlap and Add
TTS	Text To Speech
VAE	Variational Autoencoder
VC	Voice Conversion
VQ	Vector Quantization
WGAN	Wasserstein Generative Adversarial Network
WGANGP	Wasserstein Generative Adversarial Network with Gradient Penalty

1. INTRODUCTION

Voice Conversion (VC) is a problem whose goal is the modification of the source speech signal to target speech signal in a direction that linguistic aspects of the spoken content do not alter, but source utterance's properties such as style and identity are reconstructed according to target. VC is used in various applications such as speech enhancement, voice restoration, language learning, text-to-speech [10]. VC has been an intensely researched [11–15] problem, containing most of the aspects of speech processing and speech synthesis areas [10, 16].

Corpora for the VC tasks are split into parallel and non-parallel related to uttered text. For parallel data, speakers utter same script; however, in non-parallel data, speakers speak freely without any linguistic restriction. In the literature, various methods for parallel [14, 17, 18] and non-parallel [19–21] data are proposed. VC applications generally designed to expect and generate acoustic features such as formants [22], and spectrums [23]. In a recent work [24], raw-audio transformation is also studied.

Initial VC systems are developed using Vector Quantization (VQ) [14], and Gaussian Mixture Models (GMM) based statistical models [17]. Also, studies using Artificial Neural Networks (ANN) are proposed [25]. Thanks to the improvements on the computational power, training deep neural networks has become easy and fast, thence VC studies using DL proposed such as Convolutional Neural Networks (CNN) [21] and Long Short-Term Memory (LSTM) [26], and they outperform the previous GMM and ANN-based models [19, 27]. By the great achievements on deep generative models, VC research tend to study Generative Adversarial Networks (GAN) [20, 21, 28, 29], Auto Encoders (AE) [30], Variational Auto Encoders (VAE) [31–34].

By the definition of the problem, studies about the separation of content and speaker characteristics intuitively are proposed, which is named as disentangled representations [20, 35] in computer vision area. This allows models to focus on learning content and style separately leading to receive better conversions for the unseen speakers in better quality. Also, one and zero-shot learning VC methods are proposed [8,30], aiming to make the conversion of unseen speaker's voices utterances by using only a few utterances of the speaker. These methods, achieve better performance for converting unseen speakers.

In this thesis, the aim is to propose a new adversarial one-shot voice conversion system that utilizes disentangled representations. Our method is built upon a recent one-shot voice conversion approach [8] based on the VAE model. The proposed model takes and generates Mel-spectrogram features to be able to represent speech utterances. Our main contribution is adding perceptual and adversarial losses into the baseline model to improve the quality of the voice conversion. We also propose a speaker recognition model to extract features to calculate the perceptual loss which improves the quality of generated Mel-spectrogram visually. Experiments are performed on non-parallel VCTK corpora [36]. The proposed method is evaluated on unseen speakers during training for one-shot VC via separation of intergender and intragender by using Global Variance and MOSNet as metrics. Experimental results demonstrate that the proposed method outperforms significantly the baseline approach.

2. RELATED WORK

Voice Conversion problem has been an active research topic since Childers et al. [37] identified this problem. Then, several statistical-based voice conversion methods are proposed until the advances in the deep learning area. Nowadays, deep learning based methods on voice conversion gain more attention. Thus, voice conversion approaches based on statistical and learning are explained, separately, in this chapter.

Statistical Based Methods One of the pioneer voice conversion method proposed by Abe et al. [14] was based on Vector Quantization (VQ). This method represents the correspondence between two speakers by a simple concept as mapping codebooks. Shikano et al. [38] improved this method by representing the source vector as a weighted linear combination of all the codewords to quantize fuzzy vectors to avoid limitations of the discrete space representation. Valbert et al. [39] use a warping function, named as frequency warping, to minimizing spectral distances between source and target spectrums. These proposed methods have produced high quality speech; however, they lack capturing speaker identity to convert because of the stationary in the relative amplitude of the spectrum. To diminish this effect, Gaussian Mixtere Models (GMM) based voice conversion methods have been proposed by Stylianou et al. [40, 41] that assume the probability distribution of the acoustic features as a combination of Gaussians. Kane et al. [42] proposed Joint Density GMM (JDGMM) using the approximation of the joint density of the source and target features by concatenating their aligned forms. Iwanashi et al. [43] proposed another speech spectrum transformation method by interpolating spectral patterns between pre-stored multiple speakers for speech synthesis. These described methods above are required of parallel data, which requires the source, and the target speakers speak the same utterance.

Various alignment methods for nonparallel data containing different utterance for the source and the target speaker are proposed. Duxans et al. [44] proposed a Text-To-Speech (TTS) synthesizer to produce the same text for source-target pairs. HMM-based speech recognition models are used by Ye et al. [13] to labeling frames in source and target speech and create a mapping. Ney et al. [45] proposes class-based alignment by clustering source-target feature vectors independently into a given number of classes and calculates the nearest neighbors of source vectors into the corresponding target class. These methods aligns nonparallel speech pairs in order to convert spectral features.

Learning Based Methods Artificial Neural Network (ANN) are used by Narendranath et al. [46] for spectral conversion in the same manner with GMMs by using formants. Desai Et al. [25] proposed a similiar VC with ANN but using MFCCs as acoustic features. Radial Basis Function networks are proposed by Watanabe et al. [47] by using representation linear predictive coding (LPC) based spectral envelopes; however, they cannot pass the success of GMMs. Chen et al. [48] used restricted Boltzman machine (RBM) instead of GMM in order to model the Spectral joint density in order to capture the inter-speaker and inter-dimensional dependencies, improving the quality of the converted voice in terms naturalness.

Using Deep Belief Networks, which are multiple layered RBMs, Nakashika et al. [19] proposed a VC system to extract higher feature space for speech. Mohammadi et al. [27] uses the same hidden feature space learning idea with deep autoencoders. Variational autoencoders are used by Blaauw et al. [31] by Gaussian distribution to model latent space for VC. In the study of Kameoka et al. [32], VAE model with an auxillary classifier for speakers is proposed for many-to-many non-paralel VC.

Xie et al. [49] introduced a sequence based refinement to frame based voice conversion method using sequence error metric. Using Recurrent Temporal Restricted Boltzmann Machines, Nakashi et al. [50] proposed a sequence based VC. Sun et al. [15] proposed a sequence-to-sequence voice conversion method using Deep Bidirectional Long-short Term Memory (DBLSTM). In the study of Ming et al. [26], DBLSTM based sequence-to-sequence emotional VC is applied.

Kaneko et al. [21] proposed a CycleGAN based non-parallel voice conversion method with gated Convolutional neural networks (CNN). StarGAN based many-to-many non-parallel voice conversion system is proposed by Kameoka et al. [32]. Hsu et al. [34] proposed a VAE based model with Wasserstein GAN for non-paralel VC.

Representation learning is concerning learning representations of the data for making it easier to extract the information while modeling [51]. The underlying reason for the contemporary progress of the neural network models is representation learning. Disentangled representation means factorization, some latent cause, or causes of variation [52]. In voice conversion studies, disentangled representations are employed by factorization of the style and content of the speech utterance. By the factorization of both target and source utterances, the aim is to use target speakers' style representation with the content representation of the source speaker. Chou et al. [20] proposed a many-to-many VC by separating content and style representations of speech. In their other study [8], they applied similar disentangled representation idea by adaptive instance normalization for one-shot VC. An autoencoder model is used by Qian et al. [30] to create a zero-shot VC system by designing a bottleneck for disentangled representations.

3. SPEECH ANALYSIS AND SYNTHESIS

Speech production is the process that translation the speaking intentions to a hearable signal, which defined as speech. This process is created with pneumonic thrust produced by the air from the lungs that generate sound by phonation into the glottis in the larynx that then is modified by the vocal tract into different vowels and consonants [53]. Speech recognition is the process that transformation of the uttered sound waves from air into the brain as meanings. Uttered sound waves flow into the ear canal until they reach the eardrum, which passes the vibrations through the middle ear bones or ossicles into the inner ear, which called the cochlea, containing thousands of tiny hair cells. These hair cells change the vibrations into electrical signals that are sent to the brain through the hearing nerve, which will be transformed into meaning by the brain. With speech production and recognition, human communicate, explain themselves, and understands each other intentions unless some circumstances such as language barrier, noisy, insufficiently quite audio wave or dysfunctionality in hearing or production organs.

Speech Analysis and Synthesis is thereby modeling recognition and production of human speech in a way that is analyzing speech signals by various features and modeling or modifying and reconstructing speech signals by generated features. Typical voice conversion frameworks consist of speech analysis and synthesis modules, which are crucial to be high quality for conversion. Moreover, these two modules have a tight connection; the output of the analysis module should have enough features in order to achieve high quality in the synthesis module while reproducing the speech. As a consequence, in voice conversion frameworks, the analysis module required to convert speech signal into jointly independent representation elements as acoustic features (spectral) and prosodic features (pitch, duration), and the synthesis module required to reconstruct the speech signal from the analyzed parameters with high quality and naturalness. Moreover, it should allow flexible modifications of the acoustic and prosodic features without quality degradation.

The voice conversion task aims to transform the style of the speech to another person's voice, but not altering the linguistic content of it. Consequently, these frameworks require to have a source and a target speaker for the content and style of the synthesized, respectively. The converted speech signal is desired to be natural, having the same language content of the source speaker and similar to the target speaker.

The most commonly applied speech models for building the speech analysis and the speech reconstruction modules in voice conversion studies are concisely explained in the following parts.

Pitch synchronous overlap and add (PSOLA) PSOLA is a method providing high-quality synthesized speech with artifact-free prosodic modification based on the decomposition of a speech signal into various overlapping speech segments. Time Domain PSOLA (TD-PSOLA) [54] is one of the methods basing on PSOLA method, which works on time-domain for speech synthesis. It cannot be applied to voice conversion tasks because it assumes no model for the speech signal, and it runs directly on the samples in the time-domain. Consequently, it has no control over spectral envelopes.

For enabling spectral transformations, several variants of PSOLA are proposed working on other then time-domain as TD-PSOLA and Frequency Domain PSOLA (FD-PSOLA). Unlike TD-PSOLA, FD-PSOLA method transforms the speech signal in the frequency domain, hence allowing for spectral manipulation. In Linear Predictive PSOLA (LP-PSOLA), the PSOLA method is combined with a residual-excited LPC model of speech by separating the signal into a time-domain excitation and a time-varying spectral envelope. With this division, adjustments on the excitation signal are combined with re-synchronized spectral envelopes in order to produce the converted speech. As a result, FD-PSOLA and LP-PSOLA are more suitable for voice conversion than TD-PSOLA and have been used in some of the voice conversion systems proposed in the literature [39, 55, 56].

Sinusodial Models A sinusoidal model represents the speech waveform locally as a sum of sinusoids whose parameters change with time [57]. A harmonic model which is a particular type of a sinusoidal model whose sinusoids are estimated only at frequencies which are multiples of the local fundamental frequency.

The sinusoidal model is used in many voice conversion systems [40, 58] because of proving high-quality speech reconstruction and prosodic modification. Also, parameters of this model contain information about waveform and spectrum, which can be used to estimate magnitude and phase spectral envelopes in order to manipulate and convert voice.

Harmonic Plus Noise Model (HNM) HNM models are based on separation in the frequency domain by a cut-off frequency, which called as maximum voiced frequency, to decompose the speech signal into a harmonic and a noise component. The harmonic component is modeled as the sum of harmonic sinusoids up to the maximum voiced frequency, while the noise component is modeled as Gaussian noise, filtered by a time-varying autoregressive filter. The decomposition of a speech signal into harmonic and noise components allows for flexible modifications hence this model has been applied for voice conversion systems in [40, 59].

Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum (STRAIGHT) The STRAIGHT [60] model divides speech information into the mutually independent source and filter parameters by using the same theory of the source-filter model [61]. It uses pitch-adaptive spectral analysis consolidated with a surface reconstruction method in the time-frequency domain, and an excitation source design based on phase manipulation. By using these procedures, it decomposes the speech signal into three components as a smooth spectrogram, which is free from periodicity in time and frequency, a fundamental frequency (F_0) contour, and a time-frequency periodicity map which captures the spectral shape of the noise and also its temporal envelope.

STRAIGHT is a high-quality vocoder, and it allows flexible speech manipulation, as the three components are mutually independent. Also, this kind of representation is adequate to interpolate spectral envelopes and to extract speech parameters like the cepstral coefficients. It has been used in a large range of voice conversion applications [12, 17, 62, 63].

Griffin and Lim, multiband excitation vocoder Griffin-Lim model [64] represents the short-time spectrum of speech as the product of an excitation spectrum and a spectral envelope. It smoothly approximates the speech spectrum as a spectral envelope. A fundamental frequency, a voiced/unvoiced, the decision for each harmonic of the fundamental frequency, and the phase of each harmonic represents the excitation spectrum. While synthesis, model parameters are approximated in the time domain for voice part of the speech in the frequency domain for the unvoiced part of the speech. This model has been used widely in voice conversion systems [65].

3.1. Feature Extraction

Feature extraction module (or speech parameterization) uses outputs of speech analysis modules to extract features in many speech-related applications, including voice conversion task. In some cases, results of analysis like signal periods, short-time spectrum samples, LPC coefficients, amplitudes/frequencies/phases are used directly bypassing this step [24, 66], yet using these makes the voice conversion problem more complicated and computationally expensive. Some popular feature extraction methods used in voice conversion in the literature are discussed briefly on the following.

Formants Formants are resonating frequencies of the vocal tract, which are peaks of the spectral envelope. They have played a dominant role in the studies of both speech production and perception, particularly with vowels. Formants are dependent on speaker identity because the length of the pharyngeal-oral tract depends on the physical size of the speaker, which effecting the frequency locations of all vowel formants. Parameters related to formants, like formant frequencies, bandwidths, and intensity, are used in several voice conversion tasks [14, 22, 67].

Cepstral Coeficients A cepstrum is a result of taking the Inverse Fourier transform of the logarithm of the estimated spectrum of a signal. The name "cepstrum" was derived by reversing the first four letters of "spectrum". They model both spectrum peaks and valleys. Cepstral Coefficients have been widely used in speech-related systems; also, they provide a reliable measure of the acoustic distance between different frames, which is a necessary property for alignment tasks. Mel-Cepstral Coefficients (MCCs) [18], Generalized Mel-Cepstral Coefficients (GMCCs) [68], and Linear Predictive Cepstral Coefficients (LPCC) [69] are popular cepstral features used for voice conversion applications.

Line spectral frequencies (LSF) LSFs are individual representations of all-pole filters. The coefficients are close to formants, ordered ascendingly, which guaranteeing filter stability, and perturbations of them have only affect locally on the spectrum [70]. LSF is used in some voice conversion tasks [58, 71].

4. AN OVERVIEW OF VOICE CONVERSION

In this chapter, a background information about voice conversion methods is explained. Different alignment techniques for features are also discussed for parallel and non-parallel speech corpus.

4.1. Feature Alignment

In voice conversion problems, there are two sorts of training sets as parallel and nonparallel data. In parallel data, source and target speakers utter the same text during recording; however, in nonparallel data, speakers speak freely without any linguistic restriction except language. Parallel and nonparallel data require different feature alignment methods. However, in voice conversion, alignment may not be required depending on feature conversion methods.

Parallel Data When parallel data is available, the most applied algorithm is dynamic time warping (DTW) for aligning frames [72] in voice conversion [14, 18]. However, it does not take into account the differences between speakers because of searching the path of minimal global distortion among source-target pairs. There are some additions made on DTW for this purpose in the literature [16].

Another technique for frame alignment is based on Hidden Markov Models (HMM) requiring phonetic transcription of frames. In this method, utterances segmented into phonetic parts using speaker-dependent models and linear time warping [44] or dynamic time warping [58] are used to obtain higher similarity within the source and target feature vectors. However, this approach needs extremely data from both speakers.

Nonparallel Data In a practical voice conversion application, there may be only nonparallel corpora ready for the training stage. Several methods are proposed for aligning nonparallel data.

Using a Text-To-Speech (TTS) synthesizer, the alignment problem can be solved if there is enough data for the speaker to train the TTS model. In this case, after training the TTS module, the same texts are synthesized for source-target speaker pairs and then aligned by parallel data alignment methods [44].

In class-based frame alignment, clustering source and target feature vectors independently into the number of classes is aimed [45]. After the mapping among speaker pairs is established, alignment is made by finding the nearest neighbor of each source vector into the corresponding target class.

The reverse of TTS based alignment, HMM-based speaker-independent speech recognition model is used to label all frames in source and target speech with HMM state identity to create a mapping between them. Subsequent to the mapping, the longest matching sequence from source to target speech is found [13]. The performance of speech recognizer is indispensable in this approach.

4.2. Feature Conversion

Voice conversion tasks aim to transform the identity of a speaker into another speaker's content. This identity can be represented in various features such as the pitch contour, the spectral envelope, the speaking rate, the duration of the pauses, or the prosody of the speaker. However, modeling all these features are not always essential because of complexity and information they contain is not suitable for voice conversion tasks. Spectral envelopes, used as the gist of the many voice conversion tasks [10], have discriminative features for speaker identity than other features [73]. In this section, some popular spectral envelope conversion, called generally as spectral conversion, methods will be discussed briefly.

4.2.1. Mapping Codebooks

One of the simple basic voice conversion method is codebook mapping, which requires aligned source and target utterances. The primary intention is generating a mapping codebook as a compound of the source and target speaker frame-wise feature vectors.

Some distinct approaches are proposed in the literature for this kind of mapping, such as quantizing vectors independently with DTW for the frame-level match, then calculating the correspondences for each source entry to the target codebook entries in the form of histograms [14]. In the conversion phase, the source speaker's input utterance is analyzed, and the spectrum parameters are vector-quantized using the source speaker's codebook in order to decode using the source-target mapping codebooks. Furthermore, by representing the source vector as a weighted linear combination of all the codewords, fuzzy vector-quantization [38] is formed by using every correspondence within the two sets of codewords in the conversion.

4.2.2. Gaussian Mixture Models

In the voice conversion, using statistical methods for spectral envelopes transformation opens a new route. Previously mentioned methods based on the vector-quantization algorithm have drawbacks for discontinuities in the conversion function near the transitions between classes. To diminish the effect of this detriment, the division of the acoustic space into overlapping classes to which input vectors belong with a certain probability is applied. Taking this idea into consideration, spectral conversion methods based on GMM are developed [40, 41]. In these approaches, it has been assumed that the probability distribution of the acoustic features are represented as a combination of Gaussian distributions which are fitted to the training acoustic vectors of the source speaker utilizing expectation-maximization (EM) algorithm.

In Equation 4.1 feature vectors are expressed as multivariate distributions for GMM spectral conversion

$$p(x) = \sum_{c=1}^{M} \alpha_c N(x; \mu_c, \Sigma_c) \tag{4.1}$$

where p is the dimension of feature vectors, $N(x; \mu_c, \Sigma_c)$ are Gaussians representing the source speaker's acoustic space defined by the mean vector of size p, μ and the covariance matrix Σ of size $p \times p$, and M is the number of distributions, and α_c is the weight assigned to the c^{th} gaussian component of the model. A GMM-based spectral conversion function can be seen in Equation 4.2.

$$F(\mathbf{x}) = \sum_{c=1}^{M} p_c(\mathbf{x}) \left[\mathbf{v}_c + \mathbf{\Gamma}_c \mathbf{\Sigma}_c^{-1} (\mathbf{x} - \mu_i)\right] \tag{4.2}$$

where the parameters v_c and Γ_c are a mean vector and a covariance matrix Σ_c, which are calculated during the training phase by minimizing the least squares error given by the distance between the transformed vectors $F(x_k)$, and the corresponding aligned target vectors y_k. Converted acoustic features are a combination of different mean and variance transformations of the source frames, where each transformation is weighted by its posterior probabilities.

A GMM may also be used for approximation of joint density of the source and target features by concatenating their aligned forms, named as Joint Density (JDGMM) [42], and can be calculated as Equation 4.3.

$$F(\mathbf{x}) = \sum_{c=1}^{M} p_c^x(\mathbf{x}) \left[\mu_c^y + \Sigma_c^{yx} \Sigma_c^{x-1} (\mathbf{x} - \mu_c^x)\right], \boldsymbol{\mu}_c = \begin{bmatrix} \mu_c^x \\ \mu_c^y \end{bmatrix} \quad \Sigma_c = \begin{bmatrix} \mathbf{\Sigma}_c^{xx} & \Sigma_c^{xy} \\ \Sigma_c^{yx} & \Sigma_c^{yy} \end{bmatrix} \tag{4.3}$$

4.2.3. Frequency Warping

Frequency warping methods intend to find a warping function that minimizes the spectral distance between the source and target spectrum. On the conversion stage, the warping function is applied to the source spectrum frame by frame to produce target speech [71]. Customized versions of this idea are implemented in the literature. One of them is dynamic time warping, which uses dynamic programming for finding the warping function generating a frequency warping path that achieves the lowest spectral distance [39].

In the frequency warping quality of the speech is reported to be very high, but the identity of the target speaker is not entirely captured because of stationary in the relative amplitude of the spectrum. There is also a combination of frequency warping with HMM models [74] in order to improve dissimilarity issues; even though, they still have limitations in the identity conversion.

4.2.4. Deep Learning

Neural Network (NN) is a non-linear statistical model for approximation between input and output samples, and it has been applied for various voice conversion tasks in the likewise reason with GMMs for spectral mapping [75]. A neural network is formed as input, optionally no or one hidden layer but generally multiple hidden layers, and output layer. Each layer is made of artificial neurons that have parameters such as activation functions, weights, and biases [76] and can be identified as Equation 4.4.

$$y = f(Wx + b) \tag{4.4}$$

where W, b, f, y, represent weight, bias, activation function, and output, respectively, during training process W and b are optimized by a certain objective function. The activation function, f, is a mathematical "gate" in between the input feeding the current neuron and its output going to the next layer, which is a transformation that maps the input signals into output signals.

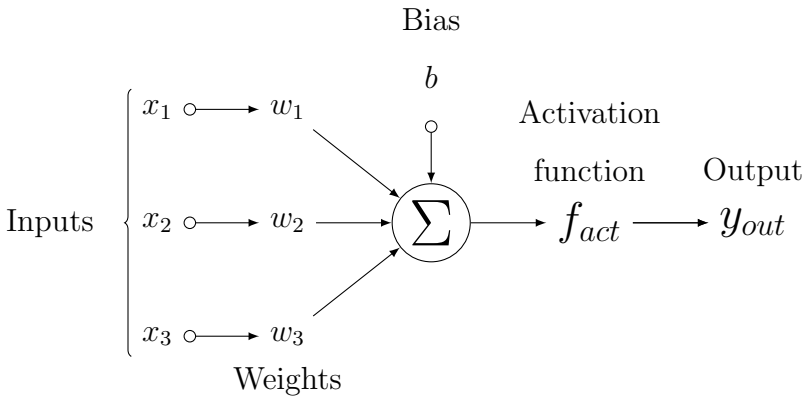

Figure 4.1. Artificial Neuron diagram [1].

In a neural network, numeric data points, called inputs, are fed into the neurons in the input layer. Each neuron has a weight, and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer.

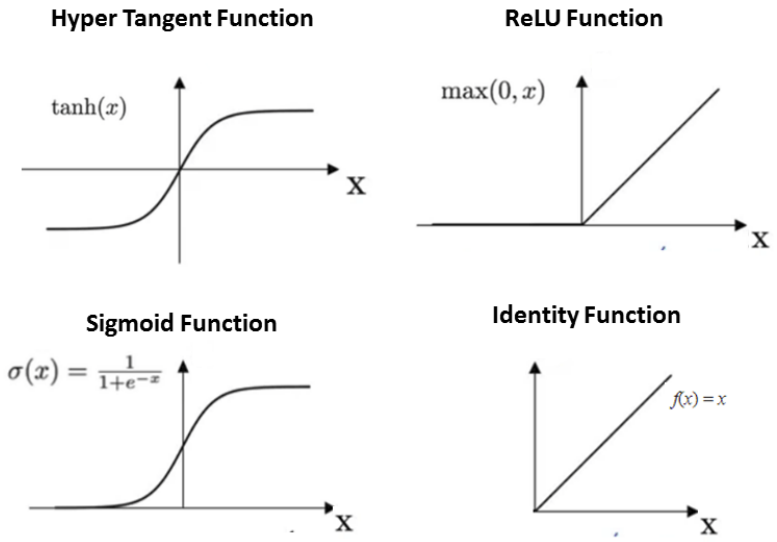

Figure 4.2. Common activation functions in neural networks [2].

Neural networks are trained by an algorithm named backpropagation that aims to minimize the cost function, which is the average loss function for the training set, iteratively by modifying the network's weights and biases. The level of change is decided by the gradients of the cost function with respect to those parameters [76].

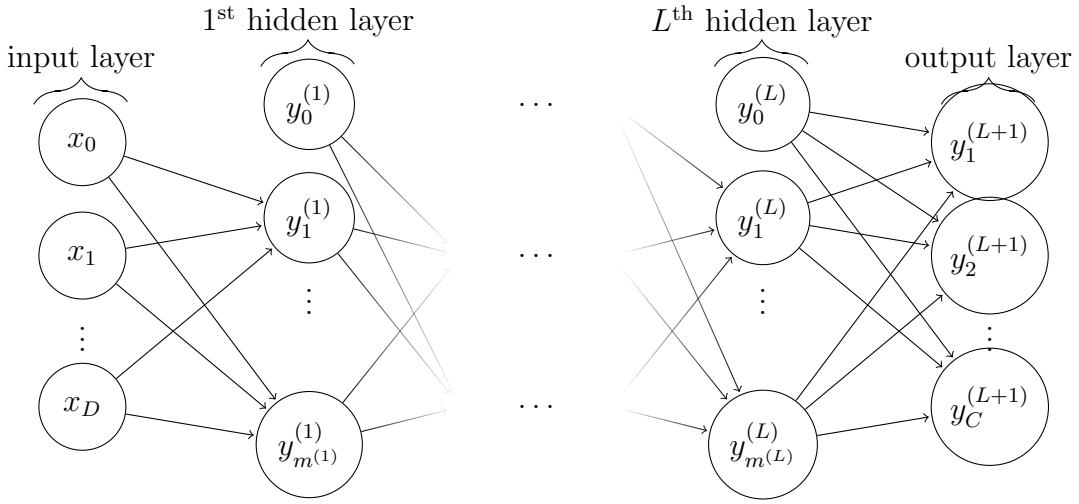

Figure 4.3. Network graph of a $(L+1)$-layer perceptron with D input units and C output units. The l^{th} hidden layer contains $m^{(l)}$ hidden units. [1]

Gradient Descent is one of the most famous optimizers which are used for updating the weights of the neural network. It updates the weights by summing the result of multiplying the gradient in that weight by the given parameter called learning rate [77], formulation can be seen in Equation 4.5.

$$\begin{aligned} \Delta \boldsymbol{w} &= -\eta \frac{\partial L}{\partial \boldsymbol{w}} \\ \boldsymbol{w}[n+1] &= \boldsymbol{w}[n] + \Delta \boldsymbol{w} \end{aligned} \tag{4.5}$$

where, w is the weight vector, n is the iteration, η represents the Learning Rate and L is the loss function.

Early ANN models are used for spectral transformation methods with different acoustic representations in the literature [23, 46]; however, these systems are outperformed by GMM models. Also, inspiring by the JDGMM based voice conversion methods, restricted Boltzmann machines (RBM) are utilized to represent the joint distributions of source and target speech features as probably density model [48].

Following the impression of Deep Learning a particular sort of machine learning algorithm that makes use of a cascade of multiple layers of nonlinear processing units for feature extraction and transformation [78], different forms Deep Neural Networks (DNN), are produced for voice conversion. Deep Belief Networks, which contain multiple RBM layers, are used to obtain higher-order feature space of source and target utterances in order to convert [19]. For the identical responsibility among DBN concerning extracting higher representations, Deep Autoencoder models are used for conversion tasks [27].

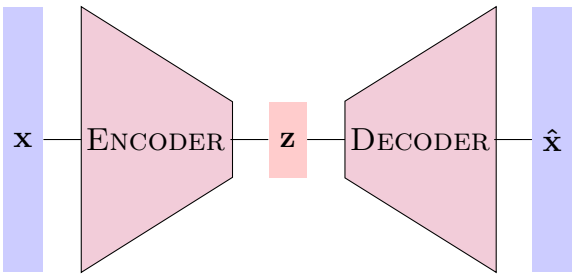

Figure 4.4. Autoencoder.

A typical autoencoder has encoder and decoder components joined by a hidden layer, which named latent representation. The encoder tries to reduce input x into z lower-dimensional space, and the decoder tries to decompress z latent space into x̂, reconstruction of input x. Nevertheless, the fundamental problem with autoencoders is that their latent space may not be continuous or allow smooth interpolation, which brings two concerns. One of them is they always replicate their inputs without any variation, the other one is that if a latent space has discontinuities and sampled from these gaps, then the decoder will produce an unrealistic output because of unseen encoded vectors coming from that range of the latent space. To handle these problems, Variational Autoencoders (VAE) are proposed whose continuous latent spaces providing simple random sampling and interpolation, which makes them valuable for generative modeling [79].

VAE, one of the most common continuous latent variable models, learns a latent space $\mathcal{Z} = \mathbb{R}^Q$ using a given set of samples $\{y\} \subseteq \mathcal{Y} = \mathbb{R}^R$ where $Q \ll R$. Figure 4.5 gives an overview of variational autoencoders, x is represented as y in equations. The model consists of two components which are the generative model $p(y|z)$, named "decoder" in VAE, given a fixed prior $p(z)$, and the inference model $q(z|y)$ named marginal likelihood also "encoder" in VAE.

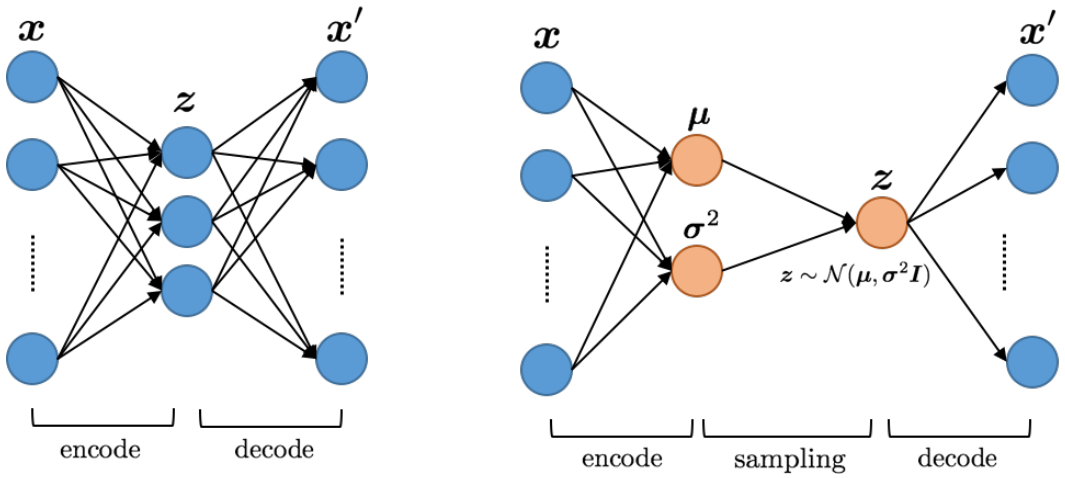

Figure 4.5. Difference between AE and VAE [3].

$$p(y) = \int p(y, z)dz = \int p(y|z)p(z)dz \tag{4.6}$$

Approximation of data distribution is given in Equation 4.6. For simple Gaussians can be determined and maximized analytically, however this is not always possible. In VAE, the main idea is to attempt to sample values of z that are likely to have produced y, and compute $p(y)$ from z using a new function $q(z|y)$ which can take a value of y and returns a distribution over z values that are likely to produce y. This procedure is identified as variational inference that is the problem of finding a model distribution $q(z)$ to approximate the true posterior $p(z|y)$.

While training a VAE model, this is accomplished by minimizing the distance between $q(z)$ and $p(z|y)$ using Kullback-Leibler (KL) divergence as a distance measure on probability distributions.

$$\mathrm{KL}(q(z)|p(z|y)) = \mathbb{E}_{q(z)}\left[\ln \frac{q(z)}{p(z|y)}\right] \tag{4.7}$$

where $\mathbb{E}_{q(z)}$ represents the expectation with respect to the distribution $q(z)$.

Proof. Rewrite KL divergence by using Equation 4.6.

$$\begin{aligned}
\mathrm{KL}(q(z)|p(z|y)) = &\mathbb{E}_{q(z)}\left[\ln \frac{q(z)}{p(z|y)}\right] \\
=&\mathbb{E}_{q(z)}[\ln q(z)] - \mathbb{E}_{q(z)}[\ln p(z|y)] \\
=&\mathbb{E}_{q(z)}[\ln q(z)] - \mathbb{E}_{q(z)}[\ln p(z,y)] + \ln p(y)
\end{aligned} \tag{4.8}$$

Re-arranging Equation 4.8 leads to the evidence lower bound, also referred to as variational lower bound.

$$\begin{aligned}
\ln p(y) =&\mathrm{KL}\left(q(z)|p(z|y)\right) - \mathbb{E}_{q(z)}[\ln q(z)] + \mathbb{E}_{q(z)}[\ln p(z,y)] \\
\geq& -\mathbb{E}_{q(z)}[\ln q(z)] + \mathbb{E}_{q(z)}[\ln p(z,y)] \\
=& -\mathbb{E}_{q(z)}[\ln q(z)] + \mathbb{E}_{q(z)}[\ln p(z)] + \mathbb{E}_{q(z)}[\ln p(y|z)] \\
=& -\mathrm{KL}(q(z)|p(z)) + \mathbb{E}_{q(z)}[\ln p(y|z)]
\end{aligned}$$

As a result, maximizing the intractable marginal likelihood $p(y)$ in Equation 4.6 transformed into approximation by maximizing the evidence lower bound, which defined in Equation 4.9.

$$-\mathrm{KL}(q(z)|p(z)) + \mathbb{E}_{q(z)}[\ln p(y|z)] = \mathbb{E}_{q(z)}\left[\ln \frac{p(y,z)}{q(z)}\right] \tag{4.9}$$

In the circumstances concerning latent variable models, $q(z)$ in Equation 4.9 shall depend on y explicitly in the same manner with $q(z|y)$ in order to reconstruct any y from the corresponding latent z.

$$-\mathrm{KL}(q(z|y)|p(z)) + \mathbb{E}_{q(z|y)}[\ln p(y|z)] = \mathbb{E}_{q(z|y)}\left[\ln \frac{p(y,z)}{q(z|y)}\right] \tag{4.10}$$

In Equation 4.10, $q(z|y)$ represents the encoder, and $p(y|z)$ the decoder which can be combined as autoencoder and implemented as a neural network, hence it can be trained by maximizing the right-hand-side of Equation 4.10 after choosing suitable parameters for distributions $p(z)$ and $q(z|y)$. Back-propagating the error through a layer that samples z from $q(z|y)$, which is a non-continuous operation and has no gradient, hence reparameterization trick is applied by a differentiable transformation $g(z,\epsilon)$ based on an auxiliary variable ϵ drawn from a distrubition. Generally, in VAE $q(z|y)$ and $p(z)$ are Gaussians because of their easy reparameterization [80].

$$q(z|y) = \mathcal{N}\left(z; \mu(y;w), \mathrm{diag}\left(\sigma^2(y;w)\right)\right) \tag{4.11}$$

$$p(z) = \mathcal{N}\left(z; 0, I_Q\right) \tag{4.12}$$

In Equation 4.11, w represents the weights of the neural network, which are parameters of mean and covariance matrix of $q(z|y)$. $\mathcal{N}$ is the Gaussian distribution and $I_Q \in \mathbb{R}^{Q\times Q}$ in Equation 4.12 is the identity matrix.

$$\begin{aligned} z_i &= g_i(y, \epsilon_i) \\ &= \mu_i(y) + \epsilon_i \sigma_i^2(y) \\ \epsilon_i &\sim \mathcal{N}(\epsilon; 0, 1) \end{aligned}$$

In conclusion, given a sample $y_m \in \mathbb{R}^R$, the objective to be minimized has the form.

$$\mathcal{L}_{\mathrm{VAE}}(w) = \mathrm{KL}(q(z|y_m)|p(z)) - \frac{1}{L}\sum_{l=1}^{L} \ln p(y_m|z_{l,m})$$

□

VAEs are used in many voice conversion systems [8, 28, 31–33]. GANs are one of the most popular deep generative models that aim to model data distribution based on differentiable generators networks and discriminator networks. The generator network produces samples from $x = g(z; \theta(g))$, which z is known distribution. Its adversary, the discriminator network, attempts to distinguish between samples drawn from the training data and samples drawn from the generator as probability value given by $d(x; \theta(d))$, indicating the probability that x is a real training example rather than a fake sample drawn from the model. Formally, the game between the generator G and the discriminator D is the minimax objective:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\text{data}}(x)[\log D(x)] + \mathbb{E}_z p_z(z)[\log(1 - D(G(z)))] \tag{4.13}$$

GANs have several prevalent failure conventions being areas of active research. One of the typical examples of these failures is gradient vanishing on the generator, which happens when the discriminator too powerful than the generator part. Another one is mode collapse, which the generator is always trying to find the one output that seems most plausible to the discriminator, so it always generates the same output. Moreover, in usual, GANs are hard to converge [81]. None of these problems have been entirely solved in the literature; though, some methods are introduced to solve them [82, 83]. Various GAN based systems are used in VC applications [21, 29].

RNNs, a specific type of DNNs whose main course is using the sequential information to learn patterns over time [84], are used in voice conversion systems to obtain more context than one frame. Using RNN, the temporal behavior of frames is modeled by regarding the former hidden layer state in addition to the current frame.

Where x_t is the input at time step t, A_t is the hidden state at time step t, which calculated based on the previous hidden state A_{t-1} and the input at the current step x_t, and h_t is the output at step t. Described RNN models are not proper for modeling long term dependencies because of the "vanishing gradient" problem, which occurs when gradient multiplied many times by a number whose absolute value is lower than 1, and "exploding gradients" problems occurring as the opposite of a vanishing.

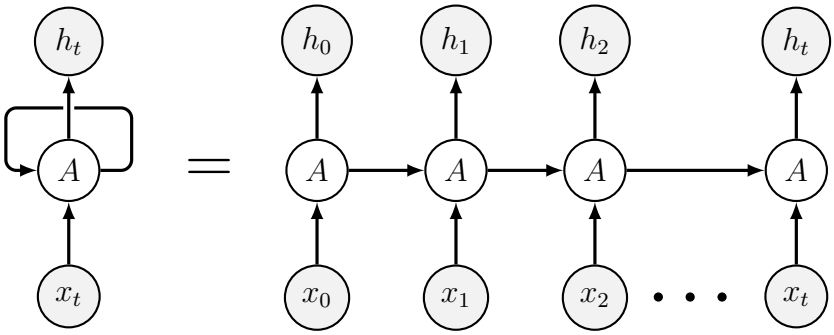

Figure 4.6. Recursive Neural Networks [4].

The vanishing gradient problem causes gradients to reach values close to 0 when propagated several timesteps back, and the exploding gradient problem causes the gradients multiplication by big numbers repeatedly, giving a lead to infinity [85]. Long Short-Term Memory (LSTM) [86] and the Gated Recurrent Unit (GRU) [87] are the most well-known introduced architectures in order to solve these difficulties.

LSTM cells contain A Cells state and 3 "gates" controlling the flow of information that enters and leaves the cell. These gates, named Input, Forget, and Output and they are trained to allow all, a part, or none of the information in the input to be taken into account to adjust the weights in the DNN of a timestep. GRU has a similar arrangement with LSTM cell, but with the Forget and Input gates blended into a single Update gate and the Cell state and the hidden state fused, in order to have the same performance with less complexity. LSTM and GRU are used in voice conversion tasks [15, 88].

CNN is a sort of deep learning model inspired by the structure of the animals' visual cortexes. CNNs are mostly used for processing 2-dimensional data such as images; however, they are additionally used in 1 and 3-dimensional data. In voice conversion systems, 1D and 2D CNNs are used for modeling acoustic features as well as directly speech signals.

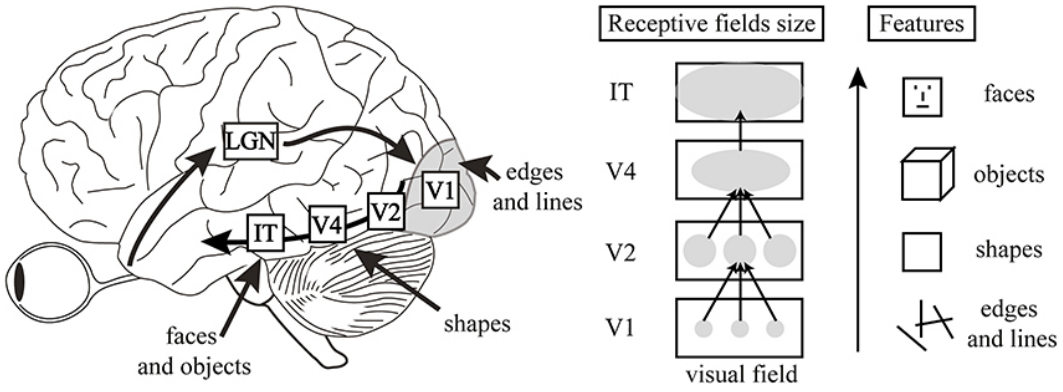

Figure 4.7. Similarity between CNNs and animal visual cortex [5].

CNNs are composed of several consecutive layers such as convolution, pooling, and fully connected layers. Convolution and pooling layers perform feature extraction, whereas a fully connected layer maps the extracted features into the final output as the prediction. They learns spatial hierarchies of features, from low-level to high-level patterns by one block after another as in Figure 4.7.

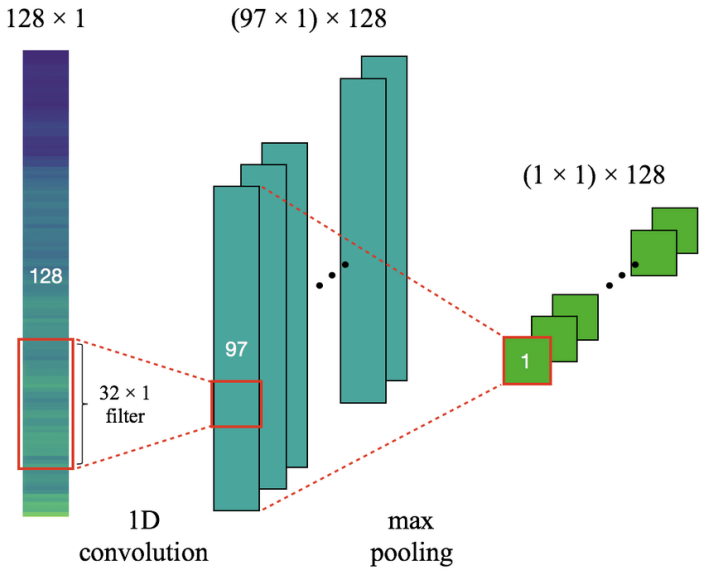

Figure 4.8. 1D convolution and max pooling operations over a spectrogram frame [6].

A convolution layer plays a crucial role in CNNs, which is composed of a stack of mathematical operations that are convolution, a specialized type of linear operation, and nonlinear activation functions that typically used in neural networks. A small grid named as kernel is slid on entire input to extract features by a constant distance which called the stride. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex.

A pooling layer provides a downsampling operation by reducing the in-plane dimensionality of the feature maps that extract by convolutional layers. This operation enables small shifts and distortions to be translation invariant; furthermore, it decreases the number of succeeding learnable parameters. Different pooling methods are proposed in the literature as max pooling and average pooling [89].

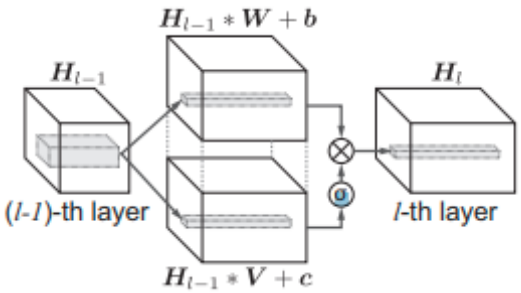

Figure 4.9. Gated Convolutional Neural Network [7].

CNNs are useful for learning features; however, they face difficulties for long-term dependency modeling. In order to solve this problem, gated convolutional neural networks are implemented similar to LSTM based gating mechanism [90]. In a gated CNN, gated linear units (GLUs) are used as an activation function instead of other activation functions. In Equation 4.14, l^{th}layer output is calculated by GLU.

$$\boldsymbol{H}_l = (\boldsymbol{H}_{l-1} * \boldsymbol{W} + \boldsymbol{b}) \otimes \sigma\left(\boldsymbol{H}_{l-1} * \boldsymbol{V} + \boldsymbol{c}\right) \tag{4.14}$$

where H_{l-1} is the previous layer's output, W, V , b, and c learnable model parameters, σ is the sigmoid function and $\otimes$ is the element wise multiplication [91]. Gated CNNs are used in many VC applications to capture broad region and long-term dependencies of acoustic features [7, 21].

5. ADVERSARIAL ONE-SHOT VOICE CONVERSION

Inspired by [8], a one-shot voice conversion method is proposed benefiting from perceptual and adversarial losses besides disentanglement representation with AdaIN. VC model also contains VAE model by disentangling the speaker and content information of the speech utterance. The underlying idea of this disentanglement is capturing the information utilizing stationarity of the speech signal. While the speaker information is invariant within an utterance, the content information varies. Hence, the model achieves this factorization by using normalization and pooling layers. Designed architecture for the training stage can be demonstrated in Figure 5.1.

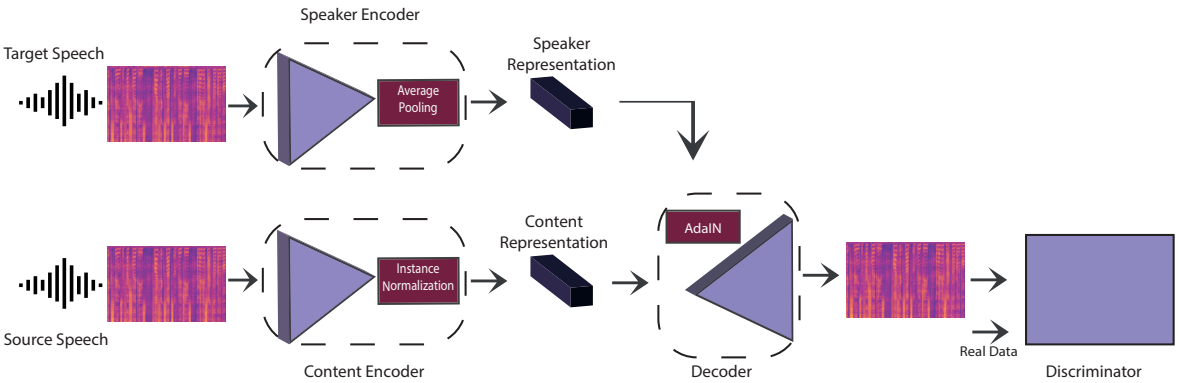

Figure 5.1. One Shot Voice Conversion with AdaIN - Training.

The proposed methods are evaluated in terms of objective and subjective. Global Variance as an objective evaluation metric, and MOSNet, which is an imitation of the mean opinion score as a subjective evaluation metric, are chosen. Experimental results show that the proposed method performed better than the baseline VAE model for both metrics.

5.1. One-shot Voice Conversion by Disentangled Representations with Instance Normalization

A VAE scheme is designed to contain two encoders, which are style and content encoders, and one decoder. Encoders aim to transform the given speech inputs into hidden disentangled representations as content and style of the utterance; then, decoders mix the representations to create a converted speech. By disentangling the speaker and content representations, encoder models learn independently from the seen samples in the training. In this way, the model shows better performance for conversion between unseen speech utterances.

In the training phase, encoder inputs and decoder outputs are the same speaker utterance, which aim is to train unsupervised way to have more robustness for the out of dataset conversions. For testing, target utterance is fed into speaker encoder, and source utterance is fed into the content encoder; hence the generated output is the converted voice from source to target speaker. During conversion, the model needs only a small utterance of the target speaker for converting the style of the source utterance. The conversion architecture in the test stage can be seen in Figure 5.2. Besides, speakers used in the testing phase do not have any utterance in the training set; therefore, the model transforms voice with one shot example.

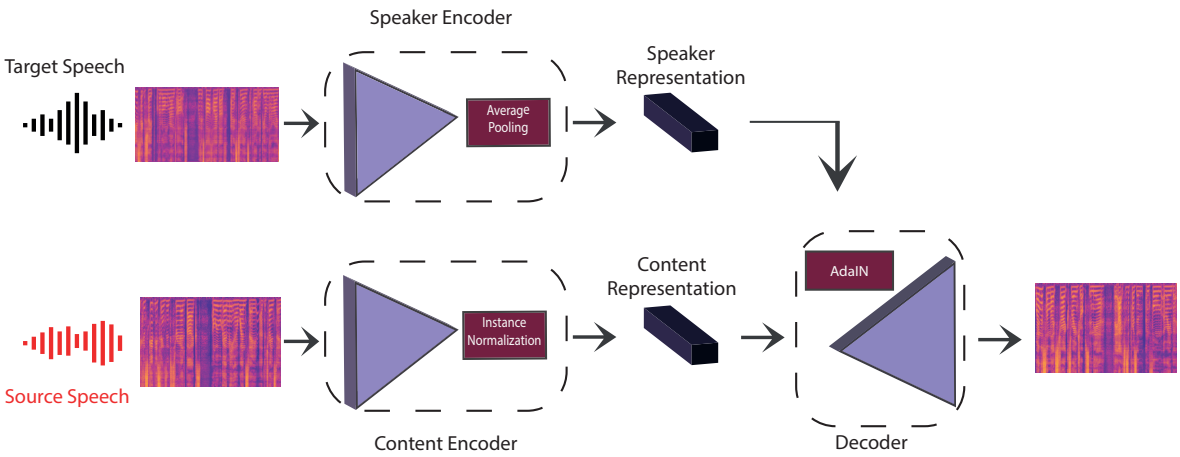

Figure 5.2. One Shot Voice Conversion with AdaIN - Test.

The input for this system is one of the popular cepstral features, Mel spectrograms. They are calculated by transforming into the frequency domain by FFT to extract magnitude information. Then, logarithm function is applied to the magnitude part followed by a scale transformation using a specialized triangular filter, named as a mel-filter bank. As a result, the input $x \in R^{N \times T}$ as Mel-spectrograms where N and T represents the number of Mel frequency bands and time-steps, respectively.

Speaker encoder E_s tries to extract speech invariant features by using the average pooling layer to learn global information over the time dimension. Content encoder E_c, however, normalizes the speaker information by instance normalization to learn content information. This is proven by disabling instance normalization layer and training the conversion model and a speaker classifier model, which tries to see how much speaker information in content representations. The lower accuracy of this classifier shows the less information is carried, and the disabling instance normalization layer shows lower accuracy results on the classifier. As a result, instance normalization and average pooling layers disentangle the representations of the speech utterance as content and speaker information.

For the voice conversion phase, the decoder part of the VAE, Dec is used with the AdaIN layer. The aim of using AdaIN is the same purpose as Huang et al. [92] work, transforming the style on the feature space in the network. The decoder performs this transformation by PixelShuffle-1D layer [93] for upsampling and generating converted Mel-spectrograms.

The base model is trained with a combination of two losses. To formulate the losses, let speaker and content encoders be, E_s, E_c and decoder be Dec. x is acoustic feature segment where randomly sampled from X, the all acoustic feature segments in the training data. E_c aims to learn content representation z_c, and E_s aims to learn speaker representation z_s.

As a general assumption in VAE models, $p(z_c|x)$ is assumed as conditionally independent Gaussian distribution with unit variance, $\mathcal{N}\left(\mathrm{E_c}(x), I\right)$. The speaker encoder model aims to minimize the reconstruction loss between different utterances belonging the same speaker, which can be formulated as:

$$L_{rec}\left(\theta_{\mathrm{E}_s}, \theta_{\mathrm{E}_c}, \theta_{\mathrm{D}}\right) = \underset{x\sim p(x), z_c\sim p(z_c|x)}{\mathbb{E}} \left[\left\|\mathrm{D}\left(\mathrm{E_s}(x), z_c\right) - x\right\|_1^1\right] \tag{5.1}$$

To learn the content representations, the distance between posterior distribution $p(z_c|x)$ and the latent prior distribution $N(0, I)$ is minimised using KL divergence. KL divergence calculates the distance between two distributions, which can be seen in Equation 5.2.

$$D_{\mathrm{KL}}\left[p_1 \| p_2\right] = \frac{1}{2}\left[\log\frac{|\Sigma_2|}{|\Sigma_1|} - n + \mathrm{tr}\left\{\Sigma_2^{-1}\Sigma_1\right\} + \left(\mu_2 - \mu_1\right)^T \Sigma_2^{-1}\left(\mu_2 - \mu_1\right)\right] \tag{5.2}$$

Since the latent prior distribution is a zero mean unit variance Gaussian, the values can be changed by $\mu_1 = \mu, \Sigma_1 = \Sigma, \mu_2 = \overrightarrow{0}, \Sigma_2 = I$ as in Equation 5.3

$$\begin{aligned} D_{\mathrm{KL}}\left[p_1 \| p_2\right] &= \frac{1}{2}\left[\log\frac{|\Sigma_2|}{|\Sigma_1|} - n + \mathrm{tr}\left\{\Sigma_2^{-1}\Sigma_1\right\} + \left(\mu_2 - \mu_1\right)^T \Sigma_2^{-1}\left(\mu_2 - \mu_1\right)\right] \\ &= \frac{1}{2}\left[\log\frac{|I|}{|\Sigma|} - n + \mathrm{tr}\left\{I^{-1}\Sigma\right\} + (\overrightarrow{0} - \mu)^T I^{-1}(\overrightarrow{0} - \mu)\right] \\ &= \frac{1}{2}\left[-\log|\Sigma| - n + \mathrm{tr}\{\Sigma\} + \mu^T\mu\right] \end{aligned} \tag{5.3}$$

By the expansion and removal of the unnecessary terms, the Equation 5.3 reduces L2 regularization of E_c.

$$\begin{aligned} D_{\mathrm{KL}}\left[p_1 \| p_2\right] &= \frac{1}{2}\left[-\log \prod_i \sigma_i^2 - n + \sum_i \sigma_i^2 + \sum_i \mu_i^2\right] \\ &= \frac{1}{2}\left[-\sum_i \log \sigma_i^2 - n + \sum_i \sigma_i^2 + \sum_i \mu_i^2\right] \\ &= \frac{1}{2}\left[-\sum_i \left(\log \sigma_i^2 + 1\right) + \sum_i \sigma_i^2 + \sum_i \mu_i^2\right] \\ L_{kl}\left(\theta_{\mathrm{E_c}}\right) &= \mathop{\mathbb{E}}_{x \sim p(x)}\left[\left\|\mathrm{E_c}(x)^2\right\|_2^2\right] \end{aligned} \tag{5.4}$$

Combining Equations 5.4 and 5.1, the loss function for VAE model becomes:

$$\mathrm{L_{vae}} = \min_{\theta_{\mathrm{E}}, \theta_{\mathrm{E_c}}, \theta_{\mathrm{Dec}}} L\left(\theta_{\mathrm{E_s}}, \theta_{\mathrm{E_c}}, \theta_{\mathrm{Dec}}\right) = \lambda_{rec} L_{rec} + \lambda_{kl} L_{kl} \tag{5.5}$$

5.1.1. Adaptive Instance Normalization

In neural networks normalization is applied to the neurons by adjusting and scaling the activations. These methods decrease the model's training time by a huge factor and reduces complexity [92]. Several normalization methods proposed in the literature such as batch normalization (BN), instance normalization (IN) and, adaptive instance normalization (AdaIN) are described.

Batch normalization (BN) uses the distribution of the summed input to a neuron over a mini-batch of training instances to compute a mean and variance parameters for normalization [94]. This normalization may be used in all or some parts of neural network with BN layers.

The primary motivation of BN is that it reduces the effect of internal covariate shift between two layers, which are the tendency of the distribution of activations to drift. Thus, BN enables training of a neural network with higher learning rates, which is the basis for faster convergence and better generalization [95]. Given a four dimensional tensor, $x \in \mathbb{R}^{N \times C \times H \times W}$ where N is mini-batch size, C is channel size, H and W are two spatial dimensions, BN normalizes the mean $\mu(x)$ and standard deviation $\sigma(x)$ for each individual feature channel across batch size and spatial dimensions independently:

$$\mathrm{BN}(x) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta \tag{5.6}$$

where affine parameters, $\gamma, \beta \in \mathbb{R}^C$, are learned from data.

$$\mu_c(x) = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw} \tag{5.7}$$

$$\sigma_c(x) = \sqrt{\frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(x_{nchw} - \mu_c(x) \right)^2 + \epsilon} \tag{5.8}$$

BN layers are initially invented to accelerate the training of networks but have also been discovered beneficial in style transfer applications.

Instance Normalization (IN) is similar to BN, which normalizes the activation of neurons using measured mean and standard deviation and used for style transfer applications [96]. Nevertheless, unlike batch normalization, the computation is made over spatial dimensions independently for each channel and each sample.

$$\mathrm{IN}(x) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta \tag{5.9}$$

where mean and standard deviation is calculated;

$$\mu_{nc}(x) = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw} \tag{5.10}$$

$$\sigma_{nc}(x) = \sqrt{\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(x_{nchw} - \mu_{nc}(x) \right)^2 + \epsilon} \tag{5.11}$$

Adaptive Instance Normalization (AdaIN) is a simple extension to IN which computes the affine parameters from the style input adaptively [92]. In voice conversion, this style input represents the style of the target speaker, which enables feature level conversion.

$$\mathrm{AdaIN}(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \tag{5.12}$$

where $\mu(y)$ and $\sigma(y)$ are the mean and standard deviation of the style input y. Plainly, normalized content input is scaled with $\sigma(y)$, and shift it with $\mu(y)$, in order to perform style transfer in the feature space by transferring feature statistics, specifically the channel-wise mean and variance.

5.2. Adversarial Loss

In essence, a GAN is composed of a generative model with a discriminator model, and they are trained jointly. Chiefly, the loss function of the generator is called reconstruction loss and adversarial loss for the discriminator. In the literature, various types of adversarial and reconstruction losses are applied with different designs of discriminator and generators are proposed regarding to the tasks. In this work, a discriminator module is combined with a VAE model that leads to the addition of another term, adversarial loss, in Equation 5.5.

The Wasserstein GAN [97] is a more modern modification of the traditional GAN, in which the goal of the discriminator is to estimate the "distance" between the two distributions. In particular, the 1-Wasserstein distance, also known as the earth-mover distance, is used. Wasserstein distance for the real data distribution $\mathbb{P}_r$ and generated data distribution $\mathbb{P}_g$ can be calculated as:

$$W\left(\mathbb{P}_r, \mathbb{P}_g\right) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{5.13}$$

where $\Pi\left(\mathbb{P}_n, \mathbb{P}_g\right)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. This distance shows the minimum cost of transporting mass in converting the data distribution $\mathbb{P}_g$ to the data distribution $\mathbb{P}_r$. The infimum (greatest lower bound) for this distance is highly intractable; however, using Kantorovich-Rubinstein duality [98] the equation can be expressed using supremum (least upper bound) which is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$.

$$W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)] \tag{5.14}$$

where $\mathbb{P}_\theta$ be the distribution of $g_\theta(Z)$ with Z a random variable with density p. In Equation 5.14, $\|f\|_L \leq 1$ can be replaced with $\|f\|_L \leq K$ ending up to $K \cdot W\left(\mathbb{P}_r, \mathbb{P}_g\right)$. Having a parameterized family of functions $\{f_w\}_{w\in\mathcal{W}}$ that are all K -Lipschitz for some K, leads to re-defining the Equation 5.14.

$$\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}\left[f_w(x)\right] - \mathbb{E}_{z\sim p(z)}\left[f_w\left(g_\theta(z)\right)\right] \tag{5.15}$$

where g_θ is a function satisfying Lipschitz condition. $W\left(\mathbb{P}_r, \mathbb{P}_g\right)$ may be differentiated by backproping through Equation 5.14 via estimating $\mathbb{E}_{z\sim p(z)}\left[\nabla_\theta f_w\left(g_\theta(z)\right)\right]$. This allows finding a solution $f : \mathcal{X} \to \mathbb{R}$ to the maximization problem defined in Equation 5.16 using function $\mathbb{E}_{z\sim p(z)}\left[\nabla_\theta f_w\left(g_\theta(z)\right)\right]$ named as "critic".

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)] \tag{5.16}$$

$$\nabla_\theta W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = -\mathbb{E}_{z\sim p(z)}\left[\nabla_\theta f\left(g_\theta(z)\right)\right] \tag{5.17}$$

To approximate the function f for solving the maximization problem in Equation 5.16, neural networks having weights w can be used, and backproped using the Equation 5.17. For enforcing the Lipschitz constraint, Arjovsky et al. proposed al. [97] clipping the weights of neural network after each gradient update. As a result, they define the loss function for critic as:

$$L = \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})] \tag{5.18}$$

Choosing clipping parameters too small or too big may cause convergency and vanishing gradient problems, therefore, Gulrajani et al. [83] proposed a better method for enforcing Lipschitz constraint by penalizing gradients. In their work, a function has been explained as 1-Lipschitz if the norm of the function's gradients at most 1 everywhere. Therefore, the gradient norm of the critic output is adjusted concerning its input in order to enforce 1-Liptschitz.

In order to bypass tractability issues, $\mathbb{P}_{\hat{\boldsymbol{x}}}$ sampling function is defined along straight lines between data and generator distributions. Using random samples $\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}$, a soft version of the constraint with a penalty on the gradient norm is enforced by $\lambda \left(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1 \right)^2$ where λ defines the penalty coefficient. In conclusion, the loss function of Wasserstein GAN with Gradient Penalty can be defined as:

$$L_{disc} = \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[\left(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1 \right)^2 \right] \tag{5.19}$$

5.3. Perceptual Loss

Generative models are trained by minimizing the reconstruction error between the input and generated instances. For instance, in speech construction, this will be an acoustic feature distance between utterances, or for image generation, this will be a pixel-wise distance between images. For example, if two images that are perceptually the same, but different from each other based on even one pixel, then based on per-pixel loss functions, they will be very different from each other. For solving this problem, Johnson et al. [99] proposed a higher level comparison for calculating the style or perceptual discrepancies and named "perceptual loss". By using a pre-trained model, higher-level features are extracted for the outputs of the generation model and used for distance calculation in order to minimize the perceptual loss by adding another term to model loss.

Extracting higher-level features, generally is applied with a classification network, which is trained on a different dataset regardless of the main problem. The performance of the classification is not expected to be excellent inasmuch as feature learning is not directly dependent on the auxiliary task. After training the auxiliary model, the model can be clipped from inputs until the chosen intermediate layers, frequently convolutional layers. During the training of the transformation model, activation of these intermediate layers is used for higher-level feature extraction.

$$\mathcal{L}_{\text{Percep}} = \lambda \left\| D(x) - D(\hat{x}) \right\|_1 \tag{5.20}$$

where D is the pretrained model for feature extraction, x and $\hat{x}$ are the real and generated data, λ is the perceptual loss weight. Mean absolute error is calculated on higher-level activations regarding to inputs, x and $\hat{x}$ to calculate perceptual loss.

5.4. Learning Objective

The models are trained with various objective functions in order to find the most effective VC configuration. Some of the loss functions affect all blocks of models; some of them only affect some parts. To compare the effects of the losses, they are added on top of the baseline one-shot VC model, cumulatively and evaluated one by one. The baseline conversion model is trained with reconstruction and content losses detailed in Equation 5.5.

5.4.1. Adversarial VC

Adversarial loss, critic, is added as another loss term to L_{vae} loss in Equation 5.5. Discriminator is trained by a WGAN-GP defined in Equation 5.19. In this approach, VAE loss which defined in Equation 5.22, is combined with the critic loss L_{critic} multiplied by a λ_{adv} parameter where D is the discriminator and Dec is the decoder part of the models.

$$\mathrm{L}_{\mathrm{critic}} = -\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] \tag{5.21}$$

$$\mathrm{L}_{\mathrm{total}} = L_{vae} + \lambda_{adv} L_{critic} \tag{5.22}$$

5.4.2. Adversarially Decoded VC

In this approach, a similar strategy in the subsection 5.4.1 is developed with the difference of applying the L_{critic} only to the decoder part of the VC model, which can be seen in the Equation 5.23. The losses of the speaker and content encoders, and the decoder are calculated by L_{vae}.

$$\mathrm{L}_{\mathrm{dec}} = L\left(\theta_{\mathrm{Dec}}\right) = \lambda_{adv} L_{critic} \tag{5.23}$$

$$\mathrm{L}_{\mathrm{total}} = \mathrm{L}_{\mathrm{vae}} + \mathrm{L}_{\mathrm{dec}} \tag{5.24}$$

5.4.3. Adversarially Decoded & Perceptually VC

The perceptual loss is calculated by L_2 form of intermediate output of the pre-trained model using converted and real utterances and formulated in Equation 5.20. In this strategy, the L_{total} loss defined in Equation 5.24 is summed up with the perceptual loss, multiplied by a λ_{perc} parameter. The calculated perceptual loss affects both encoders and decoders similarly with the L_{vae} loss.

$$\mathrm{L_{vae+perc}} = L\left(\theta_{\mathrm{E_s}}, \theta_{\mathrm{E_c}}\right) = L_{vae} + \lambda_{perc}\mathrm{L_{perc}} \tag{5.25}$$

$$\mathrm{L_{dec}} = L\left(\theta_{\mathrm{Dec}}\right) = L_{vae} + \lambda_{perc}\mathrm{L_{perc}} + \lambda_{adv}L_{critic} \tag{5.26}$$

$$\mathrm{L_{total}} = \mathrm{L_{vae+perc}} + \mathrm{L_{dec}} \tag{5.27}$$

Previously described VC training approaches are summarized in the Table 5.1 by the models and losses effected. The models containing discriminator blocks use WGAN-GP loss defined in Equation 5.19 for the discriminator.

Table 5.1. Experiments and according model settings.

Models	**Encoders**	**Decoder**
Baseline VC	$L_{rec} + L_{KL}$	$L_{rec} + L_{KL}$
Adversarial VC	$L_{rec} + L_{KL} + L_{critic}$	$L_{rec} + L_{KL} + L_{critic}$
Adversarially Decoded VC	$L_{rec} + L_{KL}$	$L_{rec} + L_{KL} + L_{critic}$
Adversarially Decoded & Perceptually VC	$L_{rec} + L_{KL} + L_{Perc}$	$L_{rec} + L_{KL} + L_{critic} + L_{Perc}$

5.5. Architecture Details

The model contains speaker and content encoder blocks with a decoder block. Besides, a discriminator for critic and VGG-16 block for perceptual loss is designed. The encoders and decoder use 1D convolution; however, the discriminator and VGG-16 model use 2D convolution. Both encoders use ConvBank layer in their first layers, which has been proposed by Wang et al. [100] to capture long-term dependencies similar to Gated CNNs.

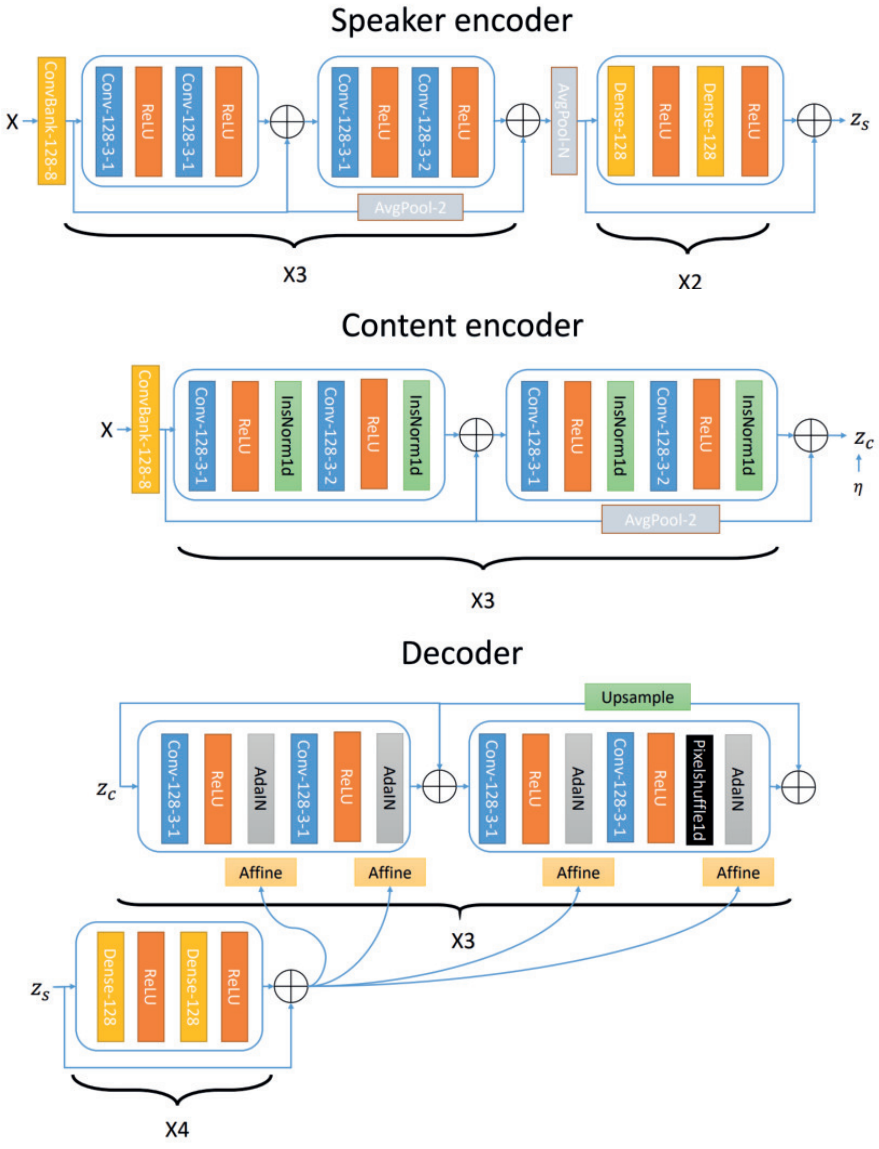

Figure 5.3. Encoder and Decoder architectures of the conversion model [8].

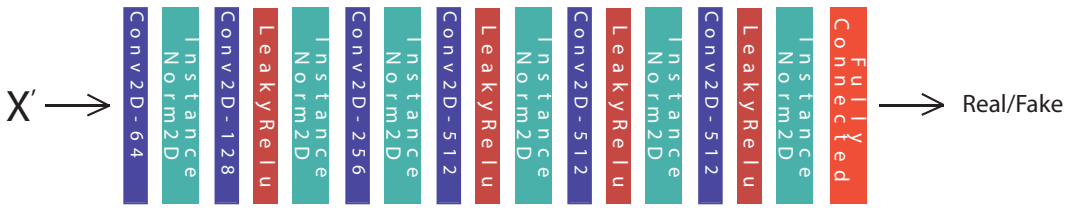

Figure 5.4. Discriminator Architecture.

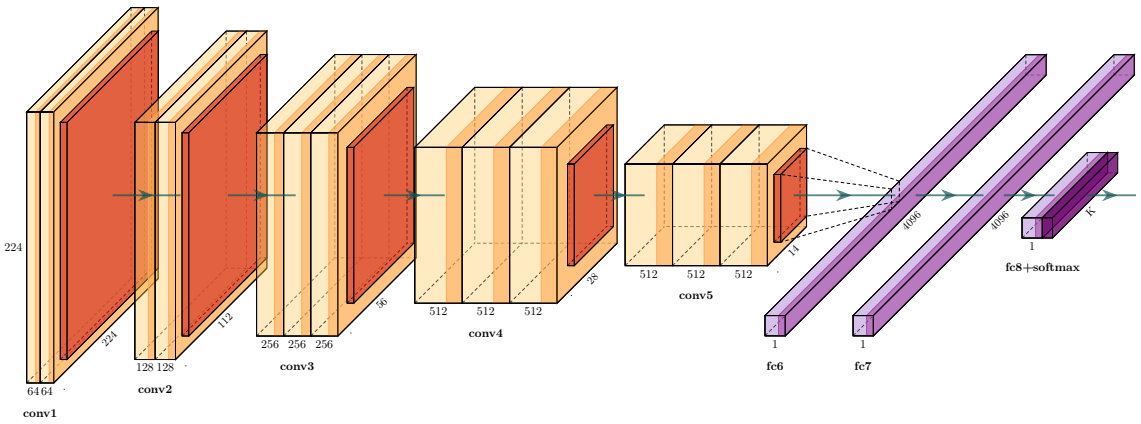

Figure 5.5. Architecture of VGG16 [9].

The input shape of the original VGG-16 model is originally $224 \times 224 \times 3$ RGB images, in this study, it is altered as $128 \times 256 \times 1$ for accepting spectrograms instead of images.

6. EXPERIMENTS AND RESULTS

6.1. Datasets

6.1.1. VCTK

The Voice Cloning Tool Kit (VCTK) Corpus [36] is published by the Centre for Speech Technology Voice Cloning Toolkit (CSTR), including speech utterances from 109 native speakers of English with various accents. Each speaker reads out about 400 sentences. This dataset is used for training and evaluating the voice conversion model.

This dataset is split into training and test sets. The training dataset contains 89 random speaker utterances, each randomly chopped into segments having length of 128. Approximately 2.5M segments are used for training set and 5K of these segments are formed validation set for parameter tuning and model selection.

For the test set, two arrangements are made in order to observe the performance of one-shot VC. For the seen speakers dataset, 8 male and 8 female speakers whose some utterances belong to the training set are chosen. Then, 10 random utterances for each speaker which not used in training set selected and randomly clipped again into 128 length segments forming a seen test set containing 80 male and 80 female utterances.

Towards generating unseen dataset, 8 female and 8 male speakers are chosen which any of their utterances are not used in the training set. 10 random utterances are selected and sampled into 128 length segments to create 80 female and 80 male utterances for the unseen test set.

6.1.2. Librispeech

The LibriSpeech corpus [101] contains approximately 1000 hours speech utterances read in English. The published dataset is divided into various categories, respectively, with their recording properties named "clean" and "other" partitioned as train, development, and test splits with the length of the recordings. A part of the development set is used for training perceptual loss model. The utterances of the 40 randomly chosen speakers are sampled into 128 length segments, forming training and validation sets.

6.2. Evaluation Metrics

6.2.1. Global Variance

In VC problems, global variance (GV) [102] has been used to see whether the result of the converted voice matches to the target speaker in terms of spectral variance distribution. GV for spectral features is calculated for each frequency index by calculating the variance of the spectral value for utterance. Generally, the calculated GV is visualized, and conversions are compared with the original target speakers.

6.2.2. MOSNet

Mean opinion score (MOS) is a quality metric used in the various domains such that objective metrics are not enough to represent overall quality. MOS is calculated by taking the mean over individual values on a scale that a subject assigns to his/her opinion of the quality of the performance. These ratings may be obtained from a subjective evolution test. MOS is the general subjective evaluation metric for VC problems.

Lo et al. [103] proposed an evaluation system to mimic subjects' opinions to a deep learning model. The proposed system, named MOSNet, contains convolutional and recurrent neural network models to predict human ratings of converted speech.

6.3. Experimental Results

6.3.1. Implementation Details

Proposed VC system development was made in Python language on version 3.7 [104] and with various Python based frameworks. For the preprocessing and feature extraction part, Librosa framework is employed [105]. Speech utterances are trimmed, to remove leading and trailing silences, following by a preemphesis stage in order to emphasize lower frequency energy to higher. In order to extract acoustic features, Short Term Fourier Transform (STFT) is applied with a 50 milliseconds window length, a 12.5 milliseconds hop length, and a 2048 STFT window size. The magnitude part of the STFT response is transformed into 256-bin Mel-scale spectrograms. Normalization by mean subtraction standart deviation division is applied to extract Mel-spectrograms. For the perceptual loss, a speech classification model is trained. For this model, the same preprocessing is applied; however, the extracted Mel-spectrograms are normalized using the calculated mean and standard deviation from the dataset used for training of the conversion model.

Neural network architectures in this application are utilized with Pytorch framework [106]. Networks are trained in NVIDIA's RTX-2070 Ti and Tesla V-100 Graphic Processing Unit (GPU)s. Automatic mixed precision [107] is used for training models by a Pytorch extension framework, named NVIDIA APEX [108]. Half precision training aims to use half-precision floating-point numbers for storing neural networks and applying matrix operations. This approximately halves the memory usage in a GPU and also speeds up the operations on some GPUs having NVIDIA's Tensor cores [109]. Tensor core allows multiplication floating point 16 (FP16) input matrices and accumulation products into either FP16 or FP32 outputs.

Apex allows usage of both FP16 to and FP32 to gain benefits from both of them without their own weaknesses increase speed, use less memory and for storing and arithmetic FP16. Weights, activations and gradients are stored using in FP16, an FP32 master copy of weights is used for updates.

Loss-scaling is used for some applications containing overflow or underflow risk. Experiments with FP16 arithmetic used Tensor Core operations with accumulation into FP32 for convolutions, fully-connected layers, and matrix multiplies in recurrent layers.

The proposed model is trained with Adam optimizer [110] with a 0.0005 learning rate, and $\beta_1 = 0.9$, $\beta_2 = 0.999$ and 32 batch size. To prevent over-fitting, weights are multiplied with weight decay rate as 0.0001 in each iteration. For a stable training, the gradient norm of the model weights are clipped to have maximum norm as 5. λ_{rec} is set to 10, λ_{kl} is set to 0.01, λ_{disc} is set to 1 and, λ_{perc} is to 10. All λ values are found empirically. The model is trained for 800K mini batches containing segments from training set.

6.3.2. Results

The VC experiments are employed on the VCTK dataset with the several setups which defined and explained in Section 5.4. The same setup names are identified in this section.

Experimental results are provided according to Global Variance and MOSNet evaluation metrics. Moreover, intergender and intragender conversions with unseen and seen speakers used in the training dataset are compared. In both speaker conversions, only one utterance of each speaker is used.

Voice conversion task is applied for F2F, M2M, F2M, and M2F conversions both unseen and seen test set. Intensive explanation of unseen and seen test sets given in Subsection 6.1.1. Results are visualized on Figure 6.1 and Figure 6.2 in terms of Global Variance in order to show that converted example match the target speaker identity. This matching can be analyzed by observing the frequency variances of the target characteristics and converted examples. As expected, the closest distance between plots one shows a better conversion. Both figures indicate that proposed methods performed better than the referenced baseline one-shot VC work.

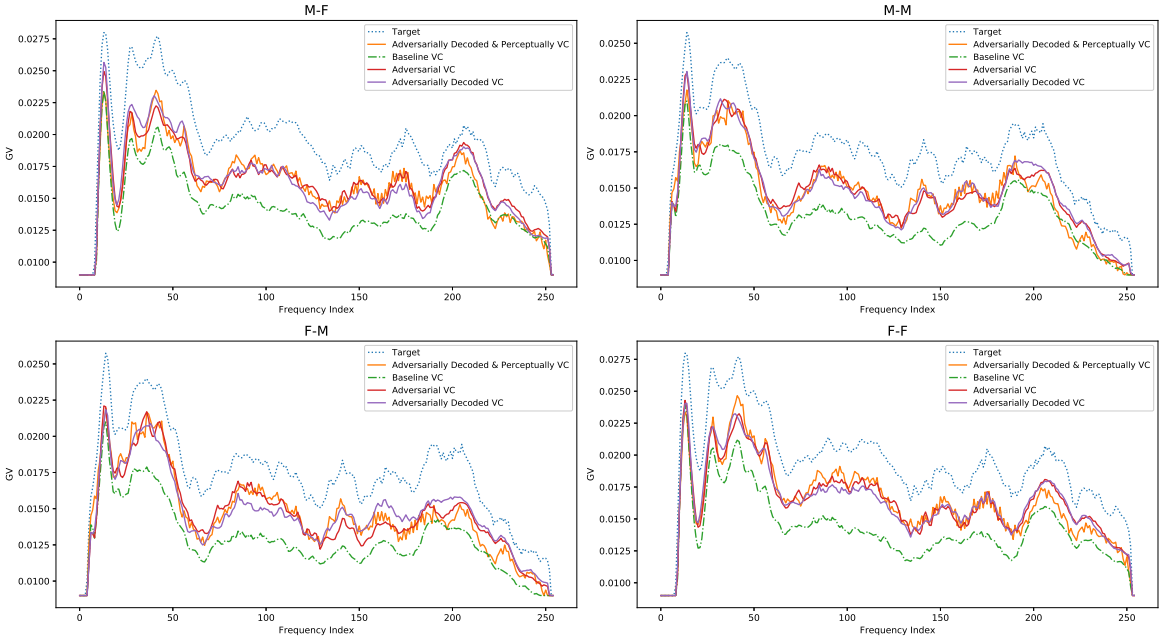

Figure 6.1. Global Variance of the seen speakers conversions.

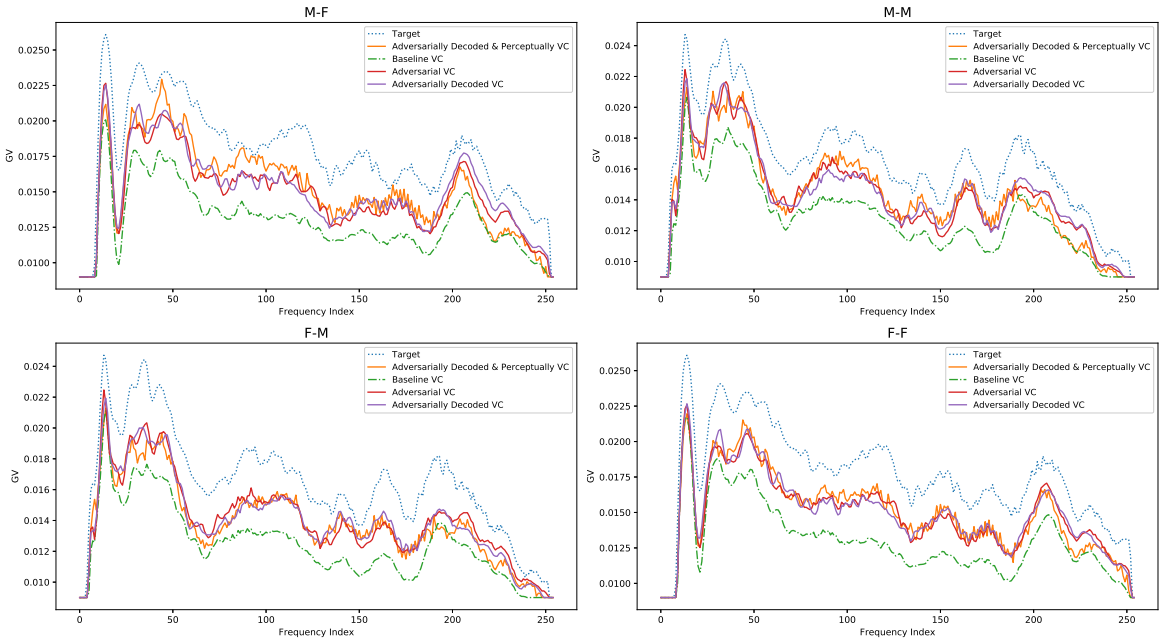

Figure 6.2. Global Variance of the unseen speakers conversions.

The Adversarial VC model matches the most speaker characteristics; however, two other developed models generate a better understanding of some frequency areas.

The converted utterances in the unseen and seen test sets are synthesized to audio from mel spectrograms by Griffin-Lim algorithm [64]. Then the implementation of the MOSNet [103, 111] which pretrained on the Voice Conversion Challenge 2018 dataset [11] is used to generate mean opinion score as 1 for the worst 5 for the best conversion. Similarly to the Global Variance results, Adversarial VC model has the most promising results on each test cases in the overall. The MOSNet results of the conversions can be seen from Table 6.1 and Table 6.2, representing the highest mean opinion score values in bold.

Table 6.1. Seen speakers conversions MOSNet Results.

Models	**M-F**	**M-M**	**F-M**	**F-F**
Baseline VC	3.66	3.28	3.25	3.63
Adversarial VC	**4.20**	4.23	4.17	**4.18**
Adversarially Decoded VC	4.11	**4.25**	**4.18**	4.12
Adversarially Decoded & Perceptually VC	4.08	3.92	3.83	4.06

Table 6.2. Unseen speakers conversions MOSNet Results.

Models	**M-F**	**M-M**	**F-M**	**F-F**
Baseline VC	3.62	3.33	3.34	3.60
Adversarial VC	**4.17**	**4.21**	4.13	**4.17**
Adversarially Decoded VC	4.04	4.18	**4.15**	4.06
Adversarially Decoded & Perceptually VC	3.99	3.83	3.79	3.99

The MOSNet and Global Variance results demonstrate that adding a discriminator to the VAE model improves the VC task significantly. The critic loss from WGAN-GP not only improves the performance of the decoder but also improves the latent representation of the speaker and content encoders by comparing the results of the Adversarial VC and Adversarially Decoded VC.

The perceptual loss from the VGG16 model is not as powerful as the adversarial loss on our VC experiments. The model used for extracting this loss is a successful feature extractor model in the computer vision field and did not show high-quality results for speech-related tasks as in our attempts.

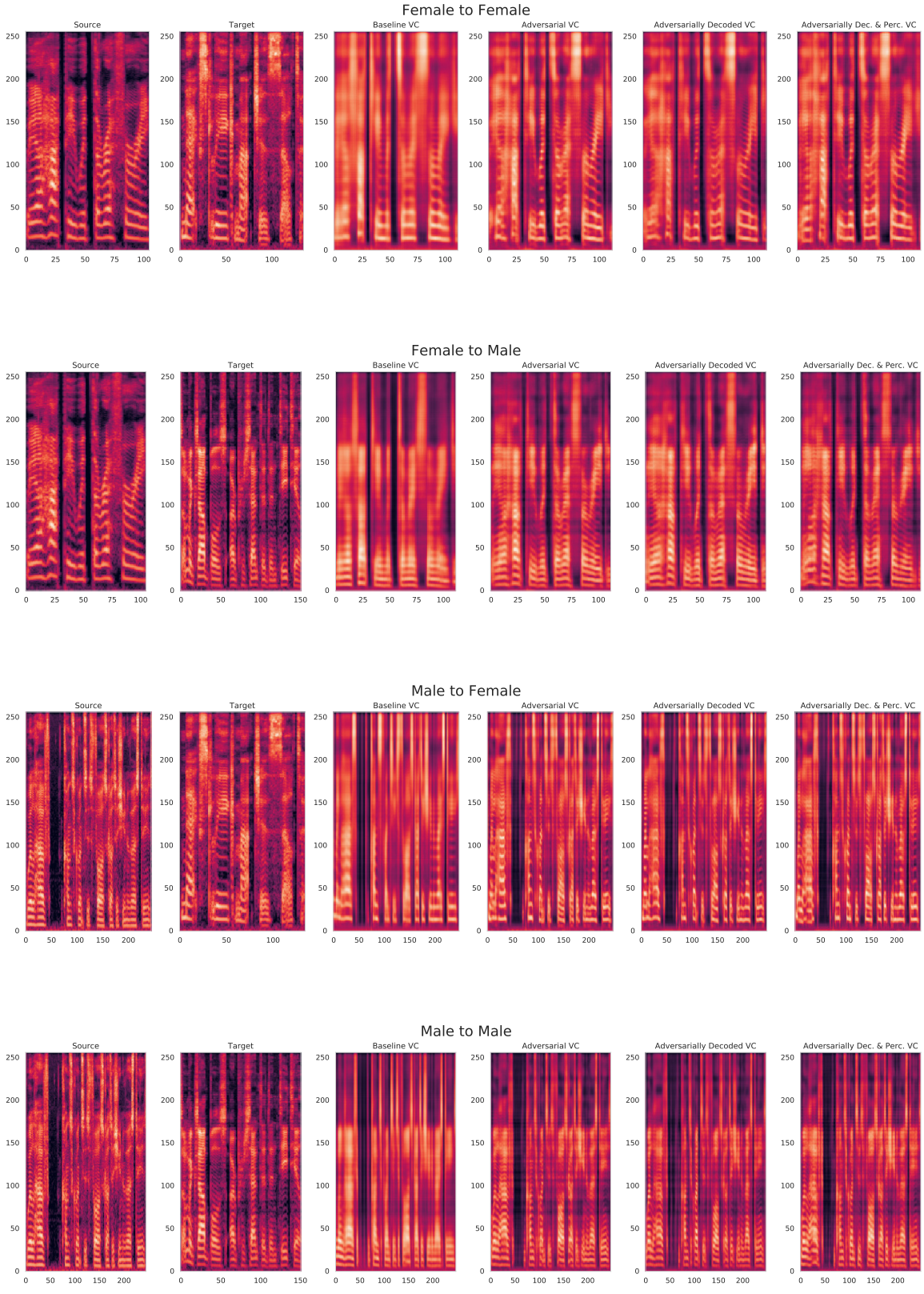

Figure 6.3. Spectrogram Heatmaps of the source, target and converted utterances for seen test set.

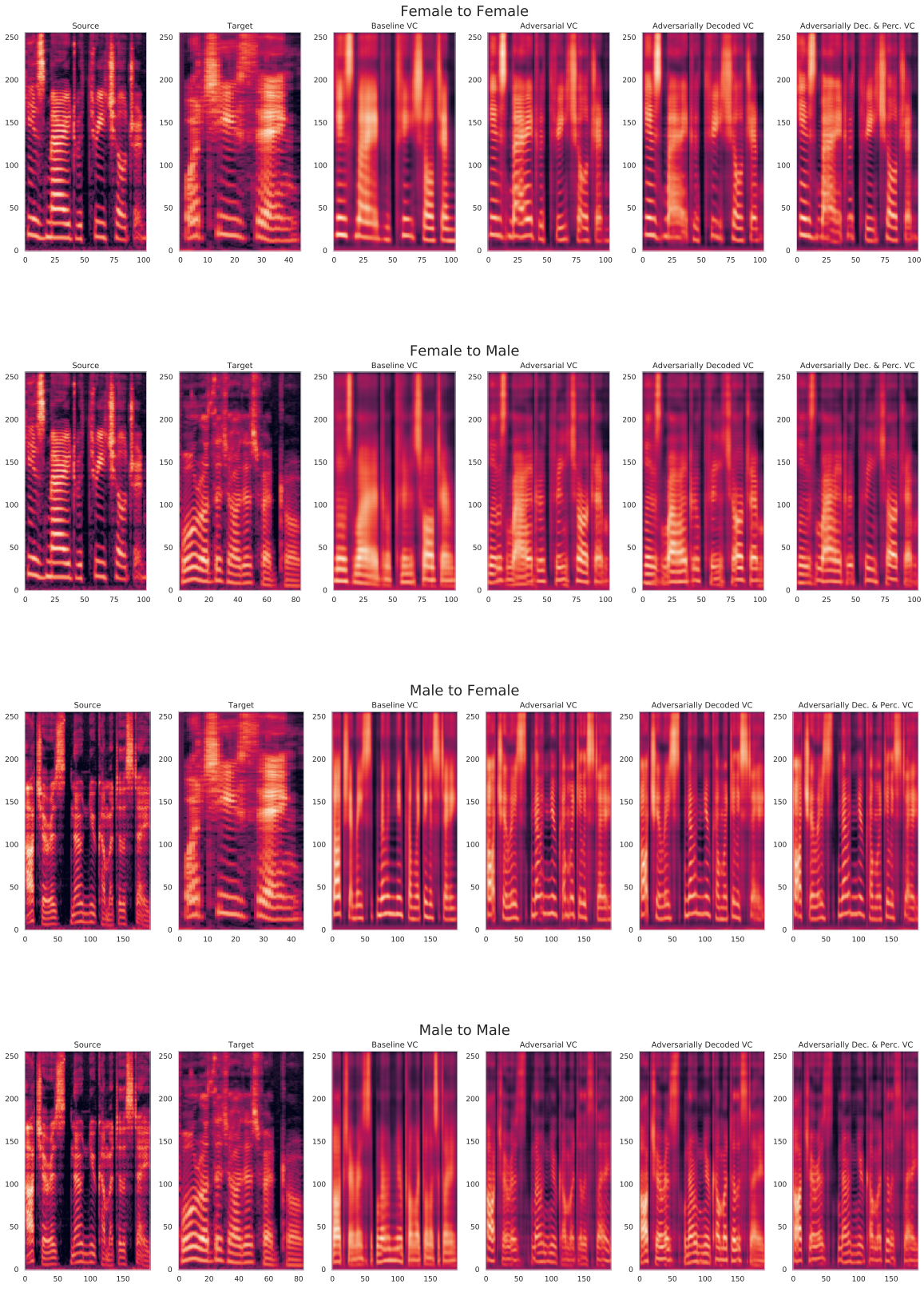

Figure 6.4. Spectrogram Heatmaps of the source, target and converted utterances for unseen test set.

7. CONCLUSION AND FUTURE WORK

In this work, a new one-shot voice conversion method was proposed. The VC system was built on disentangling speech representations by learning the content characteristics and the speaker's style of the utterance separately. The proposed VC model was a variational autoencoder using Mel-spectrograms for source and target utterance and generated a converted spectral features with the same linguistic content of the source utterance and the style of the target speaker. The model is trained unsupervised without using any target utterance. During training, in each iteration, the same utterance was used as both target and source speaker input. In the testing phase, the model could convert any source and target utterance without any restrictions, moreover allow converting unseen speakers' utterances.

The main contributions of this work was adding perceptual and adversarial losses to the VC model. For the adversarial loss, a discriminator is designed to distinguish if a converted utterance is real or fake. Concerning the perceptual loss, a speaker recognition model has been trained from a dataset different than the VC model is trained. By using this model, the perceptual loss has been calculated using converted and the source utterances. The VC experiments were performed for the utterances belonging to both seen and unseen speakers during training. We also provided intragender and intergender speaker conversions and Mel-spectrogram figures of the converted utterances. Our experiments implied that our contributions had been significantly improved the quality of the voice conversion in terms of Global Variance and MOSNet. The perceptual loss from the speaker recognition model trained by Mel-spectrogram was not much success as an adversarial loss.

In future work, we have several ideas: (i) We plan to evaluate our VC method using humans instead of MOSNet for subjective evaluation. (ii) VC tasks are moving towards using raw data as inputs bypassing the acoustic feature extraction process [24]. Moreover, we analyzed that the synthesis modules in the VC frameworks always have a bit of loss, and generated sounds are noisy.

A specialized vocoder module or directly transforming the audio signal would be helpful to increase the quality of the VC. Eventually, we aim to convert this model into a raw audio-based end-to-end VC model. (iii) We intend to try different various feature extraction methods for the perceptual loss other than using spectral features, such as SincNet [112] features that extract hidden representation from the raw audio form.

REFERENCES

1. "Collection of LaTeX resources and examples.", https://github.com/davidstutz/latex-resources, 2019, (Accessed on 11/28/2019).

2. "Neural Network Models in R", https://www.datacamp.com/community/tutorials/ neural-network-models-r, 2019, (Accessed on 01/02/2020).

3. "Variational Auto-Encoder", https://www.renom.jp/notebooks/tutorial/generative-model/VAE/notebook.html, 2018, (Accessed on 02/24/2020).

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

5. "Deep Convolutional Neural Networks as Models of the Visual System", https://neurdiness.wordpress.com/2018/05/17/deep-convolutional-neural-networks- as-models-of-the-visual-system-qa/, 2018, (Accessed on 11/28/2019).

6. Lim, H., J. Park and Y. Han, "Rare sound event detection using 1D convolutional recurrent neural networks", *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE2017)*, pp. 80–84, 2017.

7. Kaneko, T., H. Kameoka, K. Hiramatsu and K. Kashino, "Sequence-to-Sequence Voice Conversion with Similarity Metric Learned Using Generative Adversarial Networks.", *INTERSPEECH*, pp. 1283–1287, 2017.

8. Chou, J.-c., C.-c. Yeh and H.-y. Lee, "One-shot Voice Conversion by Separating Speaker and Content Representations with Instance Normalization", *arXiv preprint arXiv:1904.05742*, 2019.

9. Iqbal, H., "HarisIqbal88/PlotNeuralNet v1.0.0", https://doi.org/10.5281/zenodo.2526396, Dec. 2018.

10. Machado, A. F. and M. Queiroz, "Voice conversion: A critical survey", *Proc. Sound and Music Computing (SMC)*, pp. 1–8, 2010.

11. Lorenzo-Trueba, J., 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", *arXiv preprint arXiv:1804.04262*, 2018.

12. Wu, C.-H., 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.

13. Ye, H. and S. Young, "Voice conversion for unknown speakers", *Eighth International Conference on Spoken Language Processing*, 2004.

14. Abe, M., S. Nakamura, K. Shikano and H. Kuwabara, "Voice conversion through vector quantization", *Journal of the Acoustical Society of Japan (E)*, Vol. 11, No. 2, pp. 71–76, 1990.

15. Sun, L., S. Kang, K. Li and H. Meng, "Voice conversion using deep bidirectional long short-term memory based recurrent neural networks", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4869–4873, 2015.

16. Stylianou, Y., "Voice transformation: a survey", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3585–3588, 2009.

17. Toda, T., H. Saruwatari and K. Shikano, "Voice conversion algorithm based on Gaussian mixture model with dynamic frequency warping of STRAIGHT spectrum", *IEEE International Conference on Acoustics, Speech, and Signal Process-*

ing, Vol. 2, pp. 841–844, 2001.

18. Toda, T., A. W. Black and K. Tokuda, "Spectral conversion based on maximum likelihood estimation considering global variance of converted parameter", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. I–9, 2005.

19. Nakashika, T., R. Takashima, T. Takiguchi and Y. Ariki, "Voice conversion in high-order eigen space using deep belief nets.", *INTERSPEECH*, pp. 369–372, 2013.

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

21. Kaneko, T. and H. Kameoka, "Cyclegan-vc: Non-parallel voice conversion using cycle-consistent adversarial networks", *IEEE European Signal Processing Conference (EUSIPCO)*, pp. 2100–2104, 2018.

22. Shuang, Z.-W., R. Bakis, S. Shechtman, D. Chazan and Y. Qin, "Frequency warping based on mapping formant parameters", *Ninth International Conference on Spoken Language Processing*, 2006.

23. Bandoin, G. and Y. Stylianou, "On the transformation of the speech spectrum for voice conversion", *IEEE Fourth International Conference on Spoken Language Processing*, Vol. 3, pp. 1405–1408, 1996.

24. Serrà, J., S. Pascual and C. Segura, "Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion", *arXiv preprint arXiv:1906.00794*, 2019.

25. Desai, S., A. W. Black, B. Yegnanarayana and K. Prahallad, "Spectral mapping using artificial neural networks for voice conversion", *IEEE Transactions on*

Audio, Speech, and Language Processing, Vol. 18, No. 5, pp. 954–964, 2010.

26. Ming, H., D. Huang, L. Xie, J. Wu, M. Dong and H. Li, "Deep bidirectional LSTM modeling of timbre and prosody for emotional voice conversion", , 2016.

27. Mohammadi, S. H. and A. Kain, "Voice conversion using deep neural networks with speaker-independent pre-training", *IEEE Spoken Language Technology Workshop (SLT)*, pp. 19–23, 2014.

28. Kim, S. and H. Choi, "Emotional voice conversion using generative adversarial networks", *GAN*, Vol. 8, No. 3.169, pp. 5–784, 2017.

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

30. Qian, K., Y. Zhang, S. Chang, X. Yang and M. Hasegawa-Johnson, "AUTOVC: Zero-Shot Voice Style Transfer with Only Autoencoder Loss", *International Conference on Machine Learning*, pp. 5210–5219, 2019.

31. Blaauw, M. and J. Bonada, "Modeling and transforming speech using variational autoencoders.", *INTERSPEECH*, pp. 1770–1774, 2016.

32. Kameoka, H., T. Kaneko, K. Tanaka and N. Hojo, "ACVAE-VC: Non-parallel many-to-many voice conversion with auxiliary classifier variational autoencoder", *arXiv preprint arXiv:1808.05092*, 2018.

33. Tobing, P. L., Y.-C. Wu, T. Hayashi, K. Kobayashi and T. Toda, "Non-Parallel Voice Conversion with Cyclic Variational Autoencoder", *arXiv preprint arXiv:1907.10185*, 2019.

34. Hsu, C.-C., H.-T. Hwang, Y.-C. Wu, Y. Tsao and H.-M. Wang, "Voice conversion from unaligned corpora using variational autoencoding wasserstein generative ad-

versarial networks", *arXiv preprint arXiv:1704.00849*, 2017.

35. Gburrek, T., T. Glarner, J. Ebbers, R. Haeb-Umbach and P. Wagner, "Unsupervised Learning of a Disentangled Speech Representation for Voice Conversion", *Proceedings of the 10 Speech Synthesis Workshop (SSW10)*, 2019.

36. Veaux, C., J. Yamagishi, K. MacDonald *et al.*, "CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit", *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2017.

37. Childers, D., B. Yegnanarayana and K. Wu, "Voice conversion: Factors responsible for quality", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 10, pp. 748–751, 1985.

38. Shikano, K., S. Nakamura and M. Abe, "Speaker adaptation and voice conversion by codebook mapping", *IEEE International Sympoisum on Circuits and Systems*, pp. 594–597, 1991.

39. Valbret, H., E. Moulines and J.-P. Tubach, "Voice transformation using PSOLA technique", *Speech Communication*, Vol. 11, No. 2-3, pp. 175–187, 1992.

40. Stylianou, Y., 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.

41. Stylianou, Y., "Harmonic plus noise models for speech, combined with statistical methods, for speech and speaker modification", *Ph.D thesis, Ecole Nationale Superieure des Telecommunications*, 1996.

42. Kain, A. and M. W. Macon, "Spectral voice conversion for text-to-speech synthesis", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 1, pp. 285–288, 1998.

43. Iwahashi, N. and Y. Sagisaka, "Speech spectrum transformation by speaker interpolation", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 1, pp. I–461, 1994.

44. Duxans, H., D. Erro, J. Pérez, F. Diego, A. Bonafonte and A. Moreno, "Voice conversion of non-aligned data using unit selection", *TC-STAR WSST*, 2006.

45. Ney, H., D. Suendermann, A. Bonafonte and H. Höge, "A first step towards text-independent voice conversion", *Eighth International Conference on Spoken Language Processing*, 2004.

46. Narendranath, M., H. A. Murthy, S. Rajendran and B. Yegnanarayana, "Transformation of formants for voice conversion using artificial neural networks", *Speech Communication*, Vol. 16, No. 2, pp. 207–216, 1995.

47. Watanabe, T., T. Murakami, M. Namba, T. Hoya and Y. Ishida, "Transformation of spectral envelope for voice conversion based on radial basis function networks", *Seventh International Conference on Spoken Language Processing*, 2002.

48. Chen, L.-H., Z.-H. Ling, Y. Song and L.-R. Dai, "Joint spectral distribution modeling using restricted boltzmann machines for voice conversion.", *INTERSPEECH*, pp. 3052–3056, 2013.

49. Xie, F.-L., Y. Qian, Y. Fan, F. K. Soong and H. Li, "Sequence error (SE) minimization training of neural network for voice conversion", *INTERSPEECH*, 2014.

50. Nakashika, T., T. Takiguchi and Y. Ariki, "High-order sequence modeling using speaker-dependent recurrent temporal restricted Boltzmann machines for voice conversion", *INTERSPEECH*, 2014.

51. Bengio, Y., A. Courville and P. Vincent, "Representation learning: A review and new perspectives", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 8, pp. 1798–1828, 2013.

52. Desjardins, G., A. Courville and Y. Bengio, "Disentangling factors of variation via generative entangling", *arXiv preprint arXiv:1210.5474*, 2012.

53. Cohen, D., "Acoustic theory of speech production, with calculations based on X-ray studies of Russian articulations", , 1962.

54. Moulines, E. and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones", *Speech Communication*, Vol. 9, No. 5-6, pp. 453–467, 1990.

55. Turk, O., "Cross-lingual voice conversion", *Bogaziçi University*, Vol. 3, 2007.

56. Arslan, L. M., "Speaker transformation algorithm using segmental codebooks (STASC)", *Speech Communication*, Vol. 28, No. 3, pp. 211–226, 1999.

57. McAulay, R. and T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 34, No. 4, pp. 744–754, 1986.

58. Ye, H. and S. Young, "Quality-enhanced voice morphing using maximum likelihood transformations", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 14, No. 4, pp. 1301–1312, 2006.

59. Stylianou, Y. and O. Cappe, "A system for voice conversion based on probabilistic classification and a harmonic plus noise model", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 1, pp. 281–284, 1998.

60. Kawahara, H., "Speech representation and transformation using adaptive interpolation of weighted spectrum: vocoder revisited", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp. 1303–1306, 1997.

61. Fant, G., *Acoustic theory of speech production*, 2, Walter de Gruyter, 1970.

62. Assmann, P. F. and W. F. Katz, "Synthesis fidelity and time-varying spectral

change in vowels", *The Journal of the Acoustical Society of America*, Vol. 117, No. 2, pp. 886–895, 2005.

63. Rao, K. S. and B. Yegnanarayana, "Prosody modification using instants of significant excitation", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 14, No. 3, pp. 972–980, 2006.

64. Griffin, D. W. and J. S. Lim, "Multiband excitation vocoder", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 36, No. 8, pp. 1223–1235, 1988.

65. Holmes, W., *Speech synthesis and recognition*, CRC press, 2002.

66. Chen, K., B. Chen, J. Lai and K. Yu, "High-quality Voice Conversion Using Spectrogram-Based WaveNet Vocoder.", *INTERSPEECH*, pp. 1993–1997, 2018.

67. Mizuno, H. and M. Abe, "Voice conversion based on piecewise linear conversion rules of formant frequency and spectrum tilt", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 1, pp. I–469, 1994.

68. Helander, E., T. Virtanen, J. Nurminen and M. Gabbouj, "Voice conversion using partial least squares regression", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 18, No. 5, pp. 912–921, 2010.

69. Lee, K.-S., "Statistical approach for voice personality transformation", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 2, pp. 641–651, 2007.

70. Soong, F. and B. Juang, "Line spectrum pair (LSP) and speech data compression", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 9, pp. 37–40, 1984.

71. Erro, D., A. Moreno and A. Bonafonte, "Voice conversion based on weighted frequency warping", *IEEE Transactions on Audio, Speech, and Language Processing*,

Vol. 18, No. 5, pp. 922–931, 2009.

72. Rabiner, L., "Fundamentals of speech recognition", *Fundamentals of Speech Recognition*, 1993.

73. Furui, S., "Research of individuality features in speech waves and automatic speaker recognition techniques", *Speech Communication*, Vol. 5, No. 2, pp. 183–197, 1986.

74. Rentzos, D., S. Vaseghi, Q. Yan and C.-H. Ho, "Voice conversion through transformation of spectral and intonation features", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. I–21, 2004.

75. Funahashi, K.-I., "On the approximate realization of continuous mappings by neural networks", *Neural Networks*, Vol. 2, No. 3, pp. 183–192, 1989.

76. Ruhmelhart, D., G. Hinton and R. Wiliams, "Learning representations by back-propagation errors", *Nature*, Vol. 323, pp. 533–536, 1986.

77. Alpaydin, E., *Introduction to machine learning*, MIT press, 2009.

78. Deng, L., D. Yu *et al.*, "Deep learning: methods and applications", *Foundations and Trends® in Signal Processing*, Vol. 7, No. 3–4, pp. 197–387, 2014.

79. Kingma, D. P. and M. Welling, "Auto-encoding variational bayes", *arXiv preprint arXiv:1312.6114*, 2013.

80. Doersch, C., "Tutorial on variational autoencoders", *arXiv preprint arXiv:1606.05908*, 2016.

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

82. "Common Problems — Generative Adversarial Networks — Google Developers", `https://developers.google.com/machine-learning/gan/problems`, 2020, (Accessed on 01/01/2020).

83. Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin and A. C. Courville, "Improved training of wasserstein gans", *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

84. Cleeremans, A., D. Servan-Schreiber and J. L. McClelland, "Finite state automata and simple recurrent networks", *Neural Computation*, Vol. 1, No. 3, pp. 372–381, 1989.

85. Pascanu, R., T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks", *International Conference on Machine Learning*, pp. 1310–1318, 2013.

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

87. Cho, K., B. Van Merriënboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", *arXiv preprint arXiv:1409.1259*, 2014.

88. Zhou, C., M. Horgan, V. Kumar, C. Vasco and D. Darcy, "Voice conversion with conditional SampleRNN", *arXiv preprint arXiv:1808.08311*, 2018.

89. Lee, C.-Y., P. W. Gallagher and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree", *Artificial Intelligence and Statistics*, pp. 464–472, 2016.

90. van den Oord, A., N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals and A. Graves, "Conditional Image Generation with PixelCNN Decoders", *Advances in Neural Information Processing Systems*, pp. 4790–4798, 2016.

91. Dauphin, Y. N., A. Fan, M. Auli and D. Grangier, "Language modeling with gated convolutional networks", *International Conference on Machine Learning*, pp. 933–941, JMLR. org, 2017.

92. Huang, X. and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization", *ICCV*, pp. 1501–1510, 2017.

93. Shi, W., J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network", *CVPR*, pp. 1874–1883, 2016.

94. Ioffe, S. and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *arXiv preprint arXiv:1502.03167*, 2015.

95. Bjorck, N., C. P. Gomes, B. Selman and K. Q. Weinberger, "Understanding batch normalization", *Advances in Neural Information Processing Systems*, pp. 7694–7705, 2018.

96. Ulyanov, D., A. Vedaldi and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization", *arXiv preprint arXiv:1607.08022*, 2016.

97. Arjovsky, M., S. Chintala and L. Bottou, "Wasserstein gan", *arXiv preprint arXiv:1701.07875*, 2017.

98. Villani, C., *Optimal transport: old and new*, Vol. 338, Springer Science & Business Media, 2008.

99. Johnson, J., A. Alahi and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution", *ECCV*, pp. 694–711, Springer, 2016.

100. Wang, Y., R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis", *arXiv preprint arXiv:1703.10135*, 2017.

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

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

103. Lo, C.-C., S.-W. Fu, W.-C. Huang, X. Wang, J. Yamagishi, Y. Tsao and H.-M. Wang, "MOSNet: Deep Learning based Objective Assessment for Voice Conversion", *arXiv preprint arXiv:1904.08352*, 2019.

104. Rossum, G., *Python Reference Manual*, Tech. rep., NLD, 1995.

105. McFee, B., C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg and O. Nieto, "librosa: Audio and music signal analysis in python", *Proceedings of the 14th Python in Science Conference*, Vol. 8, 2015.

106. Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library", *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

107. Micikevicius, P., S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training", *arXiv preprint arXiv:1710.03740*, 2017.

108. "NVIDIA/apex: A PyTorch Extension: Tools for easy mixed precision and distributed training in Pytorch", `https://github.com/NVIDIA/apex`, 2019, (Accessed on 01/21/2020).

109. "Tensor Cores in NVIDIA Volta Architecture — NVIDIA", `https://www.nvidia.com/en-us/data-center/tensorcore/`, 2020, (Accessed

on 01/21/2020).

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

111. "lochenchou/MOSNet: Implementation of "MOSNet: Deep Learning based Objective Assessment for Voice Conversion"", `https://github.com/lochenchou/MOSNet`, 2019, (Accessed on 02/23/2020).

112. Ravanelli, M. and Y. Bengio, "Speaker recognition from raw waveform with sincnet", *IEEE Spoken Language Technology Workshop (SLT)*, pp. 1021–1028, 2018.