Disentanglement Learning for Text-Free Voice Conversion

Mingjie Chen

Supervisor: Thomas Hain

Department of Computer Science
University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

August 2023

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Mingjie Chen
August 2023

Acknowledgements

It is my pleasure to be a Phd student of Prof. Thomas Hain. Thomas is a kind and excellent supervisor. He is a supervisor full of knowledge and patience. He taught me how to start my research, as well as how to think logically and critically.

Many thanks to members in the MINI group and the SPandH group. Thanks for your kind suggestions and help with my work. I would especially like to thank Dr. Yanpei Shi and Dr. Qiang Huang for helping me at the early stage of my Phd journey.

Special thanks to my friends, Wanli Sun, Zehai Tu and Dr. Jisi Zhang. Thanks for your help in my study and daily life.

I would also like to thank my parents and my parents in law. Thanks for your support in my Phd journey.

Finally, I would like to say "thank you!" to the most special person ever in my life, my beautiful and lovely wife, Bingye Yi. She is the person who suggested and encouraged me to quit my job and start my Phd journey. Without her perfect cooking skills, I can not finish my studies. Her support and endless love are my greatest fortune.

Abstract

Voice conversion (VC) aims to change the perceived speaker identity of a speech signal from one to another, while preserving the linguistic content. Recent state-of-the-art VC systems typically are dependent on automatic speech recognition (ASR) models and they have gained great successes. Results of recent challenges show these VC systems have reached a level of performance close to real human voices. However, they are highly relying on the performance of the ASR models, which might experience degradations in practical applications because of the mismatch between training and test data.

VC systems independent of ASR models are typically regarded as text-free systems. They commonly apply disentanglement learning methods to remove the speaker information of a speech signal, for example, vector quantisation (VQ) or instance normalisation (IN). However, text-free VC systems have not reached the same level of performance as text-dependent systems. This thesis mainly studies disentanglement learning methods for improving the performance of text-free VC systems. Three major contributions are summarised as follows.

Firstly, in order to improve the performance of an auto-encoder based VC model, the information loss issue caused by the VQ of the model is studied. Two disentanglement learning methods are exploited to replace the VQ of the model. Experiments show that these two methods improve the naturalness and intelligibility performance of the model, but hurt the speaker similarity performance of the model. The reason for the degradation of the speaker similarity performance is studied in the further analysis experiments.

Next, the performance and the robustness of Generative Adversarial Networks (GAN) based VC models are studied. In order to improve the performance and the robustness of an GAN based VC model, a new model is proposed. This new model introduces a new speaker adaptation layer for alleviating the information loss issue caused by a speaker adaptation method based on IN. Experiments show that the proposed model outperformed the baseline models on VC performance and robustness.

The third contribution studies whether Self-Supervised Learning (SSL) based VC models can reach the same level of performance of the state-of-the-art text-dependent models. An encoder-decoder framework is established for experiments. In this framework, the performance of a VC systems implemented with a SSL model can be compared to a VC system implemented with an ASR model. Experiment results show that SSL based VC models can reach the same level of naturalness performance of the state-of-the-art text-dependent VC models. Also, SSL based VC models gained advantages on intelligibility performance when tested on out of domain target speakers. But they performed worse on speaker similarity.

Table of contents

List of figures

List of tables

Acronyms

AdaIN	adaptive instance normalisation. 35
ASR	Automatic Speech Recognition. 1
AUD	acoustic unit discovery. 5
BNE	Bottleneck. 27
CER	character error rate. 53
CIN	conditional instance normalisation. 6
CMVN	Cepstral Mean Variance Normalisation. 6
CNN	convolution neural network. 72
DNN	Deep Neural Network. 23
DPGMM	Dirichlet Process Gaussian Mixture Model. 85
DTW	Dynamic Time Warping. 22
EER	equal error rate. 54
EM	Expectation Maximisation. 70
F0s	fundamental frequencies. 31
FMLLR	Feature space Maximum Likelihood Linear Regression. 6
GAN	Generative Adversarial Network. 3
GMM	Gaussian Mixture Model. 22
IN	instance normalisation. 3

IN-WAE	instance normalisation WaveNet auto-encoder. 8
JD-GMM	Joint-Density Gaussian Mixture Model. 24
LSTM	Long Short Term Memory. 23
MCD	Mel-Cepstral Distortion. 55
MCEP	Mel-cepstral coefficients. 19
MFCCs	Mel-Frequency Cepstral Coefficients. 62
MOS	Mean Opinion Score. 41
PCC	Pearson Correlation Coefficient. 55
PPGs	Phoneme Posterior Grams. 26
PWG	Parallel WaveGAN. 21
Seq2seq	Sequence-to-sequence. 23
SSL	Self-Supervised Learning. 3
SVQ	sliced vector quantisation. 8
SVQ-WAE	sliced vector quantised WaveNet auto-encoder. 8
TTS	Text-to-Speech. 1
VAE	Variational Auto-Encoder. 33
VC	Voice Conversion. 1
VQ	vector quantisation. 3
VQ-WAE	vector quantised WaveNet auto-encoder. 5
VQVAE	vector quantised variational auto-encoder. 4
VTLN	Vocal Tract Length Normalisation. 25
WAdaIN	weight adaptive instance normalisation. 9
WAGAN-VC	weight adaptive instance normalisation voice conversion. 9
WER	word error rate. 55
ZS	Zero Speech. 5, 50

Notations

General Notations

λ	A weight.
$\mathbb{R}$	Real number.
$\mathbf{s}$	A bold lowercase letter is used to denote a vector.
$\mathbf{X}$	A bold uppercase letter is used to denote a matrix.
$\mathcal{L}$	Training objective.
$\mathcal{X}$	Data distribution
$f(\cdot)$	A function.
s	A plain lowercase letter is used to denote a scalar.

Mathematical Notations

ε	A constant.
$\mathbb{1}$	The indicator function.
$\mathbb{E}$	The expectation symbol.
$\mu = \{\mu_i\}$	A mean vector and its element.
$\sigma = \{\sigma_i\}$	A standard deviation vector and its element.
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	A Gaussian distribution.
$argmin(\cdot)$	The argmin function.
log	The logarithm.

$sg(\cdot)$	The stop gradient function.

Evaluation Metrics Notations

$B(\mathbf{U})$	The bit-rate of a sequence of representations $\mathbf{U}$.
ABX	ABX score.
$F0RMSE()$, $F0CORR()$	F0 root mean square error and F0 Pearson Correlation Coefficient.
$MCD()$	Mel-cepstral distortion function.

Auto-Encoder based VC Models

$\mathbf{Z}$	A latent Representation Matrix.
$\mathbf{z}_e$, $\mathbf{z}_q$	The latent representation and the quantised latent representation of a VQVAE model.
$\mathcal{L}_{ae}$	The training objective of the auto-encoder model.
$\mathcal{L}_{vae}$	The training objective of the VAE model.
$\mathcal{L}_{vqvae}$	The training objective of the VQVAE model.
$\mathcal{L}_{vqwae}$	The training objective of the VQ-WAE model.
$\mathcal{L}_{wav}$	The reconstruction objective of the VQ-WAE model.
θ	The parameters of the encoder of a VAE model.
$Dec()$	The decoder of an auto-encoder model.
$Enc()$	The encoder of an auto-encoder model.
$VQ()$	The VQ block of the VQ-WAE model.

GAN based VC Models

λ_{cyc}, λ_{id}	The Weights of a cycle-consistency objective and an identity-mapping objective.
λ_{cyc}^{G}, $\mathcal{L}_{cyc}^{D}$	The overall training objectives of the generator and the discriminator of the CycleGAN-VC model.
λ_{star2}^{G}, $\mathcal{L}_{star2}^{D}$	The overall training objectives of the generator and the discriminator of the StarGAN-VC2 model.

λ^{G}_{star}, $\mathcal{L}^{D}_{star}$, $\mathcal{L}^{C}_{star}$ The overall training objectives of the generator, the discriminator and the speaker classifier of the StarGAN-VC model.

$\mathbf{s}^{src}$, $\mathbf{s}^{tgt}$ Speaker identity one-hot vectors of a source speaker and a target speaker.

$\mathbf{W}$ The kernels of a 1D-convolution layer.

$\mathbf{X}^{src}$, $\mathbf{X}^{tgt}$ Speech features of a source speaker and a target speaker.

$\mathbf{z}$, $\hat{\mathbf{x}}$ A Gaussian noise variable and a generated data sample in the GAN model.

$\mathcal{L}^{G}_{wa}$, $\mathcal{L}^{G}_{wa}$ The overall training objectives of the generator and the discriminator of the WAGAN-VC model.

$\mathcal{L}^{G}_{cls}$, $\mathcal{L}^{C}_{cls}$ The speaker classifier objective of the generator and the speaker classifier of the StarGAN-VC model.

$\mathcal{L}^{D}_{cyc\_adv}$, $\mathcal{L}^{G}_{cyc\_adv}$ The adversarial training objective of the discriminator and the generator of the CycleGAN model.

$\mathcal{L}_{cyc}$, $\mathcal{L}_{id}$ The cycle-consistency objective and the identity-mapping objective of the CycleGAN-VC model.

$\mathcal{L}_{D}$, $\mathcal{L}_{G}$ The training objectives of the discriminator and the generator of the GAN model.

$\mathcal{L}^{G}_{star2\_adv}$, $\mathcal{L}^{D}_{star2\_adv}$ The adversarial training objectives of the generator and the discriminator of the StarGAN-VC2 model.

$\mathcal{L}^{G}_{star2\_cyc}$, $\mathcal{L}^{G}_{star2\_id}$ The Cycle-consistency objective and the identity-mapping objective of the StarGAN-VC2 model.

$\mathcal{L}^{G}_{star\_adv}$, $\mathcal{L}^{D}_{star\_adv}$ The adversarial training objectives of the generator and the discriminator of the StarGAN-VC model.

$\mathcal{L}^{G}_{star\_cyc}$, $\mathcal{L}^{G}_{star\_id}$ The cycle-consistency objective and the identity-mapping objective of the StarGAN-VC model.

$\mathcal{L}^{G}_{wa\_adv}$, $\mathcal{L}^{D}_{wa\_adv}$ The adversarial training objectives of the generator and the discriminator of the WAGAN-VC model.

$\mathcal{L}^{G}_{wa\_cyc}$, $\mathcal{L}^{G}_{wa\_id}$, $\mathcal{L}^{G}_{spk}$ The cycle-consistency, identity-mapping and speaker embedding reconstruction objectives of the WAGAN-VC model.

$C(\cdot)$ The speaker classifier of the StarGAN-VC model.

$G(\cdot), D(\cdot)$	The Generator and the discriminator of the GAN model.

SSL based VC Models

$\mathbf{Z}$, $\mathbf{C}$	Representations and aggregated representations in SSL models.
$\mathcal{L}_{apr}$	The autoregressive predictive reconstruction objective in SSL models.
$\mathcal{L}_{arp}$	The autoregressive predictive training objective in SSL models.
$\mathcal{L}_{mock}$	The training objective of the Mockingjay model.
$\mathcal{L}_{mp}$	The masked predictive training objective in SSL models.
$\mathcal{L}_{mr}$	The masked predictive reconstruction objective in SSL models.
$I(\cdot,\cdot)$	Mutual information.

Others

$\boldsymbol{\beta}, \boldsymbol{\gamma}$	Two affine parameters.
$\mathbf{s}_{emb}$	A speaker embedding.
$\mathcal{L}_{ar}$	The training objective of autoregressive vocoder models.
$\mathcal{L}_{nonar}$	The training objective of non-autoregressive vocoder models.
$d()$, $s()$	The distance function and the search function of VQ.
$SEnc()$	A speaker encoder.

Chapter 1

Introduction

The objective of a Voice Conversion (VC) system is to change the perceived speaker identity of a speech signal to another speaker identity (Sisman et al., 2020), while the linguistic information of the speech signal is supposed to be unchanged (Childers et al., 1989).

VC techniques can be used in a wide range of areas. For example, they can be used for applications relevant to speaker identity transformation, such as transforming the voices of an actor to another actor (Mukhneri et al., 2020, Turk and Arslan, 2006) and transforming the singing voices of a singer to another singer (Turk et al., 2009, Villavicencio and Bonada, 2010). Besides, voice conversion techniques can be applied to other speech tasks. For example, they can be applied in speech synthesis systems (Kain and Macon, 1998, Gabryś et al., 2022, Zhang et al., 2019b), so that a synthetic speech signal can be transformed to a target speaker. Furthermore, VC techniques can be applied to generate data for enhancing the training of an Automatic Speech Recognition (ASR) system (Shahnawazuddin et al., 2020, Singh et al., 2021, McCarthy et al., 2020, Harvill et al., 2021) or a Text-to-Speech (TTS) system (Terashima et al., 2022, Ribeiro et al., 2022, Terashima et al., 2022). VC techniques can be exploited for speaker privacy preservation (Srivastava et al., 2020, Yoo et al., 2020) or employed in healthcare applications. For example, enhancing the intelligibility of a speech utterance spoken by a dysarthric speaker (Huang et al., 2022a, Zheng et al., 2022, Wang et al., 2020a, Zhao et al., 2019, Aihara et al., 2017).

Typically, researchers (Abe et al., 1990, Stylianou et al., 1998) name the two speakers in VC as source speaker and target speaker, respectively. A VC system converts the speaker identity of a speech signal from a source speaker to a target speaker. Research of VC was started by (Stevens et al., 1953). A type of method (Moulines and Charpentier, 1990, Valbret et al., 1992) is to directly modify the waveforms of a speech signal. As discussed in (Mohammadi and Kain, 2017), these methods produce high quality speech. However, these methods mainly modify the pitch information of a speech signal, which implies they can not flexibly modify the speaker identity of a voice (Mohammadi and Kain, 2017). In order to enhance the speaker similarity performance of VC systems, one solution is to simplify VC with constraints. For example, some VC systems (Kain and Macon, 1998, Stylianou et al., 1998, Abe et al., 1990, Shikano et al., 1991, Eide and Gish, 1996) applied constraints on the training data. They built their VC systems by learning a very local mapping function on parallel data. Parallel data means pairs of two samples containing the same linguistic information, where one is from a so-called source speaker and the other one is from a so-called target speaker. VC systems can be built up based on parallel datasets (Toda et al., 2016, Lorenzo-Trueba et al., 2018, Kominek and Black, 2004), however, parallel datasets are not always available (Sisman et al., 2020) in practical applications.

To open the opportunities for non-parallel data, many VC systems (Liu et al., 2021, 2020b, 2018, Kaneko and Kameoka, 2017, Kaneko et al., 2019b, Chou and Lee, 2019) have been developed. One obvious choice is to build a cascade system of an ASR model (Watanabe et al., 2017, Dong et al., 2018) and a TTS model (Shen et al., 2018, Li et al., 2019). In this thesis, for simplicity, these VC systems are named as ASR-TTS systems. In a VC system, apart from models, there could be other processes such as a feature extraction or a vocoder step. Hence, this thesis names the models of the ASR-TTS systems as ASR-TTS models. The ASR-TTS models employ an ASR model to transcribe an input speech signal into texts then use a TTS model to generate a converted signal.

The ASR-TTS models (Huang et al., 2020a, Zhang et al., 2020) and their extension models (Liu et al., 2020b, 2021, Zhao et al., 2021) are highly relying on good performing ASR models (Liu et al., 2020b, 2021, Dong et al., 2018, Watanabe et al., 2017). The training of these ASR models usually requires large datasets (Deng and Li, 2013) with text transcriptions. For simplicity, this thesis names these VC models as text-dependent models.

In the VC research community, challenges (Toda et al., 2016, Lorenzo-Trueba et al., 2018, Yi et al., 2020, Dunbar et al., 2019) usually were organised for comparing different techniques under the same evaluations, so that the differences of techniques can be shown. Text-dependent models gained successes in recent VC challenges (Lorenzo-Trueba et al., 2018, Yi et al., 2020). Some of them (Liu et al., 2020b, 2018) reached the level of quality close to real human voices (Yi et al., 2020).

ASR models might experience performance degradations in practical applications, because of the mismatch between the training and test data (Deng and Li, 2013), for example, multi-lingual or cross-lingual scenarios (Deng and Li, 2013) or children speech (Gerosa et al., 2009). In these text-dependent VC models, the performance degradations of the ASR model may cause mispronunciation issues of the converted speech signals. Because the errors of the output of the ASR models can be propagated to the converted speech signals, so that the linguistic information of the converted speech signal is different to the input speech signal.

Because of the performance issue of the ASR models, researchers started to build VC models (Kaneko and Kameoka, 2017, Qian et al., 2019, Chou et al., 2018, Chou and Lee, 2019, Kaneko et al., 2019a, 2020, Kameoka et al., 2018, Kaneko et al., 2019b, Huang et al., 2022b) without ASR. These VC models as typically named as text-free models.

As introduced above, in the context of VC, the model is supposed to keep the linguistic information of an input speech signal unchanged. A typical solution is to remove the speaker information from an input speech signal and then add the speaker information of another speaker. To achieve this, VC models usually apply disentanglement learning methods (Sisman et al., 2020), such as vector quantisation (VQ) (Gray, 1984) or instance normalisation (IN) (Huang and Belongie, 2017). Disentanglement learning aims to learn factorisation between different information. VQ is a data compression method and IN is a normalisation method. The objective of applying these disentanglement learning methods in VC models is to learn the disentanglement between the speaker-dependent information and the speaker-invariant information of a speech signal.

The information in a speech signal can be roughly categorised into three classes: the linguistic information, the speaker information and the prosodic information. The speaker-invariant information can be regarded as the representations of the linguistic information, which is expected to be preserved in a VC model. Because the speaker information and the prosodic information of a speech signal are relevant to the speaker identity (Sisman et al., 2020).

Hence, in text-free VC systems, the objective of the disentanglement learning methods is to learn the representations of the linguistic information of a speech signal, without depending on an ASR model. The text-free VC models can be categorised into the following three classes (Yi et al., 2020):

- Auto-encoder based models (Chou et al., 2018, Chou and Lee, 2019, Van Den Oord et al., 2017, Qian et al., 2019, Cho et al., 2019, Wu et al., 2020, Tang et al., 2022);
- Generative Adversarial Network (GAN) based models; (Kaneko and Kameoka, 2017, Kaneko et al., 2019a, 2020, Kameoka et al., 2018, Kaneko et al., 2019b);
- Self-Supervised Learning (SSL) based models (Huang et al., 2022b, 2021).

The disentanglement learning methods are implemented in VC models in various ways. For example, VQ has been mainly applied to the auto-encoder based VC models and the speech SSL based VC models. IN has been mainly exploited in the auto-encoder based VC models and the GAN based VC models.

However, text-free VC models generally can not reach the same level of performance as text-dependent VC models (Yi et al., 2020). From the results of the recent challenges (Yi et al., 2020, Dunbar et al., 2019), the performance of the text-free VC models (Cho et al., 2019, Kaneko and Kameoka, 2017, Kameoka et al., 2018) was still poor.

1.1 Research Objective

The objective of work in this thesis is to improve the performance of the text-free VC models. The disentanglement learning methods can remove the speaker information but usually cause the models to produce low quality speech. This is usually because the disentanglement learning methods cause a linguistic information loss issue.

The first two contributions of this thesis are mainly about solving the linguistic information loss issue caused by the disentanglement learning methods. The first part studies the linguistic information loss issue caused by VQ in the auto-encoder based models. Then the second work aims to solve the linguistic information loss issue caused by IN (and its extension methods) in the GAN based models. In the third work, this thesis studies whether the text-free SSL based models can reach the same level of performance of text-dependent models.

Solving the Information Loss Issue of the Auto-Encoder based Model

The auto-encoder model (Le Cun and Fogelman-Soulié, 1987, Bourlard and Kamp, 1988, Hinton and Zemel, 1993) have been found useful for representation learning (Bengio et al., 2013, Zhuang et al., 2015, Li et al., 2017) and VC (Mohammadi and Kain, 2015, 2016, Qian et al., 2019, Chou et al., 2018). The objective of representation learning is to learn useful representations from data, so that the representations can benefit the performance of other tasks (Bengio et al., 2013).

The auto-encoder model is composed of an encoder and a decoder. The encoder compresses data into a latent representation and the decoder reconstructs the input data from the latent representation. The vector quantised variational auto-encoder (VQVAE) model (Van Den Oord et al., 2017) is an extension of the auto-encoder model. The VQVAE model applied VQ in the auto-encoder model. VQ is a data compression technique (Gray, 1984). The objective of applying VQ in the auto-encoder model is to compress redundant information in the latent representations.

In the context of VC, the VQVAE model and its extension models (Wu et al., 2020, Tang et al., 2022, Cho et al., 2019) found that VQ can compress the speaker information in the latent representations. Because of this, the VQVAE based models have gained successes in VC, especially in a special test scenario where source and target speakers are outside of training data.

The VQVAE model has also been found effective for learning representations from speech data (Chorowski et al., 2019, Tjandra et al., 2019, Cho et al., 2019). For example, the acoustic unit discovery (AUD) task (Versteegh et al., 2015, Dunbar et al., 2017, 2019) is an unsupervised speech representation learning task. The objective of this task is to learn useful phonetic representations from speech without text transcriptions. The learned representations are supposed to be invariant to speaker information. This task is closely relevant to the objective of the text-free VC models, because both of them are learning the disentanglement.

In the recent Zero Speech (ZS) 2019 challenge (Dunbar et al., 2019), these two tasks were set up together in a multi-task scenario. In this challenge, models (Cho et al., 2019, Tjandra et al., 2019, Eloff et al., 2019, Liu et al., 2019) built upon the VQVAE model obtained competitive performance. The vector quantised WaveNet auto-encoder (VQ-WAE) model (Cho et al., 2019, Chorowski et al., 2019) was among these models. It is an extension of the VQVAE model.

The challenge evaluated the two tasks separately. In the evaluations of the challenge, the VQ-WAE model performed well on the AUD task, however, poorly on the VC task. The reason for the poor VC performance of the VQ-WAE model has been discussed in the summary paper of the ZS 19 challenge (Dunbar et al., 2019). The VQ in the VQ-WAE models were commonly configured with high discretisation, which caused too much information being compressed. This can effectively remove the speaker information from the latent representations but also the linguistic information. Hence, the high discretisation of the VQ in the VQ-WAE model causes a linguistic information loss issue. This issue might be the reason for the poor VC performance of the VQ-WAE model.

Solving the Information Loss Issue of the GAN based Model

GAN based VC models (Kaneko and Kameoka, 2017, Kaneko et al., 2019a, 2020, Kameoka et al., 2018, Kaneko et al., 2019b) applied GAN models proposed for image style transfer tasks to VC. For example, the CycleGAN model (Zhu et al., 2017) and the StarGAN model (Choi et al., 2018).

GAN based VC models typically exploited IN (Huang and Belongie, 2017) and its extension method (Dumoulin et al., 2017) for removing speaker information. The IN layer is one of normalisation layers (Huang and Belongie, 2017, Ioffe and Szegedy, 2015, Wu and He, 2018) commonly used in neural networks. These normalisation layers can be applied in neural networks as an intermediate layer. The objective of applying the IN layer in GAN based models is to remove the speaker information from intermediate features. The IN layer normalises intermediate features through a mean variance normalisation across time. In the context of the disentanglement learning, the normalised intermediate features can be regarded as the representations of the speaker-invariant information. Furthermore, the IN layer can be extended to adaptation layers (Dumoulin et al., 2017). They are used in GAN based models for adapting intermediate features to a target speaker.

The StarGAN-VC2 model (Kaneko et al., 2019b) is one of the recently proposed GAN based VC models. The StarGAN-VC2 model applied the IN layer, as well as an adaptation layer, the conditional instance normalisation (CIN) layer (Dumoulin et al., 2017). The CIN layer is an extension method of the IN layer. This layer normalises intermediate features through a mean variance normalisation then it adapts the normalised intermediate features through a linear transform.

Normalisation methods (Viikki and Laurila, 1998, Eide and Gish, 1996, Prasad and Umesh, 2013, Liu et al., 1993) and adaptation methods (Huang et al., 2005, Gales, 1998, Tibrewala and Hermansky, 1997, Varadarajan et al., 2008) have been widely studied in speech tasks such as speech recognition. The objective of applying the normalisation methods in speech recognition is to reduce the mismatch of training and test data for ASR models (Prasad and Umesh, 2013). The adaptation methods were exploited to enhance the performance of ASR models on the data of a target speaker.

Some normalisation methods such as Cepstral Mean Variance Normalisation (CMVN) (Viikki and Laurila, 1998, Rehr and Gerkmann, 2015) are similar to the IN layer. Because the CMVN can remove the speaker information from speech data (Rehr and Gerkmann, 2015). Another common attribute of the IN layer and the CMVN is that both conduct the mean variance normalisation across time. Hence, the IN layer might suffer from the same issue that the CMVN suffered in speech recognition. It has been found that the CMVN suffered from degradations when applied on a short utterance, because the data is insufficient for estimating the mean and the variance of the utterance (Prasad and Umesh, 2013). This issue could lead to an information loss of the normalised data (Prasad and Umesh, 2013).

In speech recognition, many adaptation methods have been used. One type of adaptation methods, such as Feature space Maximum Likelihood Linear Regression (FMLLR) (Gales, 1998, Varadarajan et al., 2008), adapts speech data through a linear transform. The FMLLR is similar to the CIN layer as both conduct adaptation through a linear transform. Hence, the issue of the FMLLR in speech recognition might happen when the CIN layer is applied in GAN based VC models. The FMLLR has been found causing performance degradations of ASR models when the adaptation data is insufficient (Joy et al., 2018, Senior and Lopez-Moreno, 2014, Parthasarathi et al., 2015). Because when not enough data is available, the FMLLR might cause an over-fitting issue.

On the other hand, the StarGAN-VC2 model has gained attention because it is a text-free model. This can be an advantage in practical applications because the datasets with text transcriptions are not always available. One drawback of the StarGAN-VC2 model is that it has only been studied on one dataset (Lorenzo-Trueba et al., 2018). The dataset only contains four speakers with 5 minutes data per speaker. Hence, it is still not clear whether the performance of the StarGAN-VC2 model can be maintained when trained on various datasets.

As discussed above, in speech recognition, it has been found that the performance of the adaptation methods is dependent on whether data is sufficient. It is still a question whether the CIN layer of the StarGAN-VC2 model will cause a linguistic information loss issue when data is insufficient. In other words, how to improve the performance and the robustness of the StarGAN-VC2 model is to be studied.

Study the Performance of SSL based Models

As mentioned before, most text-dependent VC models rely on ASR. A common approach of these models is to employ an ASR model to generate the text transcriptions (Zhang et al., 2020, Huang et al., 2020a) or the representations of the linguists information (Liu et al., 2018, 2020c, 2021, Zhao et al., 2021) of an input speech signal. Then a TTS model is used to generate a speech signal from the text transcriptions or the linguistic representations.

One approach for text-free models is to replace the ASR model with an unsupervised equivalent, for example, a SSL model. The SSL models (Schneider et al., 2019, Baevski et al., 2019, 2020, Hsu et al., 2021, Chung and Glass, 2020, Chung et al., 2020, Liu et al., 2020a) aim to learn representations from speech data without text transcriptions. The representations from SSL models have been found useful in many speech tasks, such as ASR (Baevski et al., 2020, Hsu et al., 2021), speaker recognition (Chen et al., 2022) and emotion recognition (Pepino et al., 2021).

S3PRL-VC (Huang et al., 2022b) found that SSL models can be useful in VC. The objective of the S3PRL-VC is to study whether SSL models can be used as the replacement of the ASR model in a VC system. For simplicity, this thesis names the VC model that applies a SSL model as SSL based VC model.

To study the performance of SSL based VC models, they conducted experiments on the dataset of a challenge (Yi et al., 2020), so that the performance of SSL based VC models can be directly compared with the best-performing text-dependent VC models (Liu et al., 2018, 2020b) in the challenge. In the S3PRL-VC, SSL based VC models were trained and then tested on the challenge dataset. They found that the performance of SSL based VC models were still behind the level of the best-performing models.

A drawback of the S3PRL-VC is that, when comparing SSL based VC models to the best-performing models, it was not a fair comparison. The SSL based VC models were trained on a dataset with only 2 hours data. This dataset is smaller than the training datasets of the best-performing models, for example, in (Liu et al., 2020b) the internal dataset contains 130 hours data. Hence, it is still not clear whether text-free SSL based VC models can reach the same level of performance as the best-performing text-dependent VC models, when using the same training dataset.

In summary, this thesis aims to study the following three research questions:

- How can the linguistic information loss issue of the VQ be solved in the VQ-WAE model?
- Will the CIN layer in the StarGAN-VC2 model cause a linguistic information issue? If so, how can one improve the performance and the robustness of the StarGAN-VC2 model?
- Can SSL based VC models reach the same level of performance as text-dependent VC models, when training on the same dataset?

1.2 Contribution

The following lists the contributions of this thesis:

1. To study the linguistic information loss issue caused by the VQ of the VQ-WAE model, two alternative disentanglement learning methods are introduced as the replacement of VQ. The first method is using IN as the disentanglement learning method. The second method is to replace VQ with an extension method of VQ, sliced vector quantisation (SVQ) (Kaiser et al., 2018, Jegou et al., 2010). This work proposes two models, namely instance normalisation WaveNet auto-encoder (IN-WAE) and sliced vector quantised WaveNet auto-encoder (SVQ-WAE). The two proposed disentanglement learning methods can alleviate the linguistic information loss issue, so that they can improve the naturalness and the intelligibility of VC. However, the two disentanglement learning methods cause a degradation of speaker similarity performance. This work further hypotheses the phonetic information and the speaker-dependent information in these two models are entangled. In the further analysis, it has been found that these two methods cause the entanglement. Because of the entanglement, the IN-WAE model suffered from a poor VC performance. But the SVQ-WAE model performed well on VC. In addition, it has been found that IN suffered from a linguistic information loss

issue when input utterance is too short. Part of this work has been published in the INTERSPEECH 2020 conference (Chen and Hain, 2020).

2. In this work, the StarGAN-VC2 model has been evaluated in various training data situations. These training data situations are set up with varying numbers of speakers and varying number of utterances per speaker. It has been found that the StarGAN-VC2 model experiences performance degradations when decreasing the amount of data per speaker. In order to improve the performance and the robustness of the StarGAN-VC2, a new model is proposed, namely weight adaptive instance normalisation voice conversion (WAGAN-VC). This new model introduces a new speaker adaptation layer, namely weight adaptive instance normalisation (WAdaIN). It has been found that the WAGAN-VC model outperforms the StarGAN-VC2 model on the VC performance and the robustness. It has been found that the WAdaIN layer can alleviate the linguistic information loss issue. This work leads to a publication (Chen et al., 2021a) in the ICASSP 2021 conference.

3. To analyse the VC performance of SSL based VC models and text-dependent VC models, a framework has been set up for experiments. This framework allows a fairly comparison between a VC system implemented with an ASR model and a VC system implemented with a SSL model. This framework also allows researchers to compare the performance of a VC system implemented with an autoregressive TTS model and the performance of a VC system implemented with a non-autoregressive TTS model. This work chooses the VQ-Wav2vec model (Baevski et al., 2019) as a representative of SSL models. In the experiments, three VC tasks are set up to cover various test scenarios of VC, including a many-to-many VC task, an intra-lingual VC task and a cross-lingual VC task. It has been found that VC systems implemented with the VQ-Wav2vec model can obtain better naturalness performance, however, poorer speaker similarity. In addition, VC systems implemented with the VQ-Wav2vec mode can obtain better performance when tested on out-of-domain target speakers. A speech re-synthesis task has been conducted as an analysis. It has been found that the VQ-Wav2vec model can preserve more prosodic information in representations. Apart from this, the performance gap between the autoregressive TTS model and the non-autoregressive TTS model has been clearly shown. The autoregressive TTS model obtains better performance because it can better model the prosodic information of a speech signal.

Audio demos of this thesis can be found in https://mingjiechen.github.io/thesis/index.html

1.3 Thesis Outline

The remainder of the thesis is structured as follows:

- Chapter 2 reviews VC techniques. It introduces a workflow of VC techniques and categorisations of VC models. It reviews the performance of various VC models in the literature.
- Chapter 3 reviews recent unsupervised techniques for learning disentangled phonetic representations from speech. It introduces the disentanglement learning methods and the models used for this task. It also introduced the relevance of this task to VC.
- Chapter 4 studies how to solve the linguistic information loss issue of the VQ-WAE model.
- Chapter 5 investigates whether the StarGAN-VC2 model can maintain the performance on varying training data situations and proposes a new model for improving the performance and the robustness of the StarGAN-VC model.
- Chapter 6 studies whether the performance of SSL based VC models reach the same level of text-dependent VC models.
- Chapter 7 summarises the main conclusions and presents potential future research directions.

Chapter 2

Voice Conversion

2.1 Introduction

The objective of a VC system is to change the perceived speaker identity of a speech signal to another speaker (Sisman et al., 2020). An illustration of the use of a VC system can be found in Figure 2.1. If a speaker A produces a sentence, for example "Nice to meet you", the VC system will convert the waveform into an equivalent with the same sentence but spoken in the style of a speaker B. For this purpose, the VC system requires speaker information from the speaker B, which can be encoded from their data resources. For a VC system, speaker A and speaker B are regarded as the source speaker and the target speaker, respectively. A listener should perceive that the converted speech signal sounds like the speaker B saying "Nice to meet you". In principle, the converted speech signal is expected to sound natural and similar to the target speaker.

The information about a speaker is encoded in complex ways in a speech signal. It reflects the physical characteristics of a speaker, but also other elements such as their biological sex, social gender, socioeconomic or regional background and age (Schweinberger et al., 2014). Some of these characteristics are highly related to other information in speech signals, such as emotion, accent and even thereby the content of the speech signal. There is no known computational model that will allow to encode speaker information perfectly for this purpose.

Although speaker information is related to various other information, to better understand a VC system, the information in a speech signal can still be roughly categorised into three classes:

- linguistic information, such as phonetic information;
- para-linguistic information, such as emotion, accent, prosodic information;

- speaker information, such as the spectrum and formants of a speech signal.

Prosodic information usually is seen as a composition of three types of information: pitch, energy and duration. Among these, pitch, duration are commonly treated as speaker-dependent information (Sisman et al., 2020).

Based on this categorisation, a VC system is to change the speaker information and preserve the linguistic information. But for para-linguistic information, it is complicated what a VC system is expected to do. A VC may need to change the prosodic information and accent of a speech signal, to ensure the converted speech signal sounds similar to the target speaker. However, whether a VC system should change the emotion information of a speech signal is still a question.

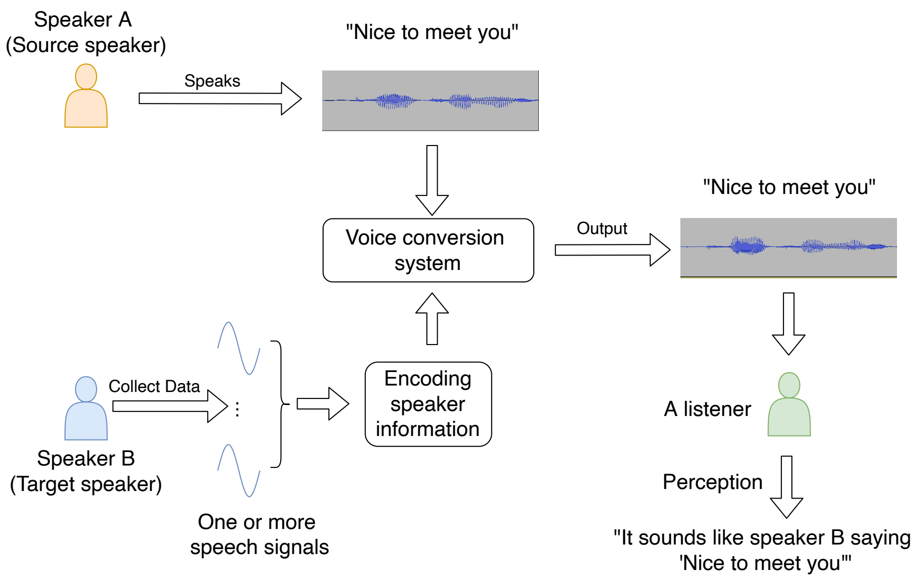

Fig. 2.1 An illustration of the use of a VC system.

2.1.1 A Workflow of VC Systems

A simple solution for VC is to directly modify the waveform of a speech signal. There have been methods (Moulines and Charpentier, 1990, Valbret et al., 1992) that directly act on waveforms. However, as discussed in (Sisman et al., 2020) and (Mohammadi and Kain, 2017), these methods can produce high quality speech signals but are not flexible for modifying the speaker information (Mohammadi and Kain, 2017). Instead, VC systems follow a workflow that depends on speech analysis and reconstruction methods. The objective of the speech analysis methods is to extract features from a speech signal. Different from methods that directly modify waveforms, the speech analysis methods output frame-wise features, which

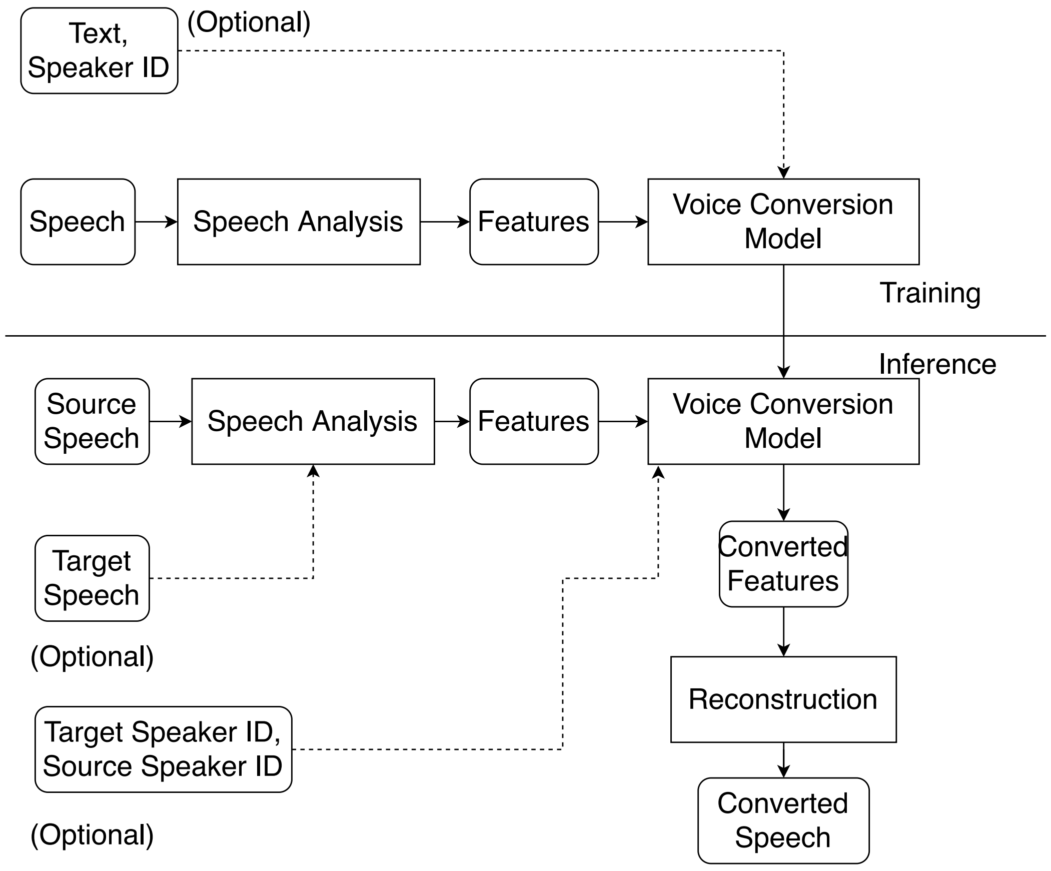

Fig. 2.2 A typical workflow of VC models (Sisman et al., 2020), where "speaker ID" denotes speaker identity.

can be regarded as "some form of intermediate representation for effective manipulation or modification with respect to the acoustic properties of speech" (Sisman et al., 2020). The objective of the reconstruction methods is to reconstruct a speech signal from the extracted features. Further details of the speech analysis and reconstruction methods will be introduced in Section 2.2.

An illustration of a workflow (Sisman et al., 2020) can be found in Figure 2.2. The workflow is composed of two phases, a training phase and an inference phase. The objective of the training phase is to train a VC model. At the inference phase, the trained model is applied for VC tasks. The training phase is composed of two steps: speech analysis and VC model training. Specifically, the speech analysis step extracts speech features from speech signals. Then the speech features are used for training the VC model. Besides, the training of the VC models might require other resources such as text transcriptions or speaker identities. At the inference phase, given a source speech signal, the VC system is to convert the source speech signal to a target speaker. Similar to the training phase, various optional inputs are needed, such as a target speech signal or the speaker identities of the source and target speakers.

2.1.2 Categorisations of VC Systems

Practical VC systems have evolved dramatically over time and many different categorisations of such systems exist. Generally speaking, given a speech signal of any target speaker, a perfect VC system is supposed to convert a speech signal from any source speaker to the target speaker. However, due to the complexity of speech data, achieving this is difficult. Thus, in order to simplify the VC task, researchers have applied various constraints on VC systems and these constraints lead to various categorisations of VC systems. These categorisations distinguish systems in the way they process data, in the way they need data for training of models and in the information they need in order to perform the conversion. The following presents a list of these categorisations then further details are introduced below.

- Parallel data vs. Non-parallel data;
- Intra-lingual vs. Cross-lingual;
- Source-to-target mappings: one-to-one vs. many-to-one vs. one-to-many vs. many-to-many vs. any-to-any;
- Text-dependent vs. Text-free.

Parallel Data vs. Non-Parallel Data

Early VC systems (Abe et al., 1990, Shikano et al., 1991, Kain and Macon, 1998, Stylianou et al., 1998, Eichner et al., 2004) applied a constraint on the form of training data. The training of these systems requires paired speech signals, one from a source speaker and the other one from a target speaker, with the two samples containing the same linguistic information. In this chapter, the datasets (Toda et al., 2016, Lorenzo-Trueba et al., 2018, Kominek and Black, 2004) that provide paired speech signals are named as parallel data. This chapter names these systems as parallel data VC systems, or parallel data systems for simplicity. Correspondingly, systems that do not depend on parallel data are named as non-parallel VC systems, or non-parallel systems for simplicity.

By constraining the training data as parallel pairs, the VC task can be seen as a feature mapping task. Speech features are extracted from a source speech signal and a target speech signal. The VC model maps a frame of source features to a frame of target features. Parallel data VC systems provide a solution for VC when parallel data is available. More details of parallel data VC models will be introduced in Section 2.3.

However, in practice, parallel datasets are not always available (Sisman et al., 2020). Hence this constraint might become a limitation in real applications. In order to release this constraint, many non-parallel data VC models (Liu et al., 2018, 2020b, 2021, Kaneko and Kameoka, 2017, Kameoka et al., 2018) have been studied. A categorisation of non-parallel data VC models and further details will be introduced later.

Intra-Lingual vs. Cross-Lingual

This categorisation distinguishes the languages of the source speaker and target speaker in VC. Intra-lingual VC systems (Abe et al., 1990, Shikano et al., 1991, Kain and Macon, 1998, Stylianou et al., 1998, Eichner et al., 2004, Liu et al., 2018, 2020b, 2021) only supports source speakers and target speakers that speak the same language. In contrast, cross-language VC systems (Zhou et al., 2019, Mohammadi and Kim, 2018, Zhou et al., 2019, Erro and Moreno, 2007, Qian et al., 2011, Turk et al., 2009) support source speakers and target speakers that speak any languages.

Early VC systems such as parallel data systems applied a constraint that the language of the source and target speakers should be the same. Because parallel data for two speakers that speak different languages is not always available. There have been several studies (Qian et al., 2011, Turk et al., 2009) on cross-lingual VC by using parallel data models. Most cross-lingual systems (Zhou et al., 2019, Mohammadi and Kim, 2018, Zhou et al., 2019) are built upon non-parallel VC models.

Moreover, this categorisation can also distinguish VC tasks into intra-lingual VC tasks and cross-lingual VC tasks. In intra-lingual VC tasks, a source speaker and a target speaker should speak the same language. In cross-lingual VC tasks, a source speaker and a target speaker should speak different languages.

Source-to-Target Mappings

In VC, a source-to-target mapping means a unique pair of a source speaker and a target speaker (Kaneko et al., 2019b). This chapter follows the concept of the source-to-target mappings and use it to distinguish VC systems. To distinguish how many source and target speakers that a VC system can support, the source-to-target mappings includes four cases: one-to-one (Abe et al., 1990, Shikano et al., 1991, Stylianou et al., 1998, Kain and Macon, 1998, Sundermann and Ney, 2003, Huang et al., 2020a, Kaneko and Kameoka, 2017, Kaneko et al., 2019a, 2020), one-to-many (Toda et al., 2007, Saito et al., 2011), many-to-one (Sun et al., 2016b, Huang et al., 2021, Chen et al., 2019, Saito et al., 2020), many-to-many (Liu et al., 2018, 2020b, Kaneko et al., 2019b, Kameoka et al., 2018). For example, the one-to-one mapping means the system only supports one source speaker and one target speaker and the one-to-many mapping means the system supports only one source speaker but multiple target speakers. This is an implicit constraint on these four cases, that is all of them only focus on test speakers that appear in training data.

Furthermore, researchers (Liu et al., 2021, Lin et al., 2021b, Lu et al., 2019, Chou and Lee, 2019, Qian et al., 2019, Wu et al., 2020, Tang et al., 2022) followed this categorisation and named another case, any-to-any. Any-to-any VC systems should support "any" source speakers and "any" target speakers, whether or not they appear in training data. Speakers that appear in training data are called seen speakers in VC. Speakers that are outside of training data are called unseen speakers.

Early VC systems, such as parallel data systems, usually are regarded as one-to-one systems. There are also several non-parallel VC systems (Kaneko and Kameoka, 2017, Kaneko et al., 2019a, 2020, Huang et al., 2020a,b,c) that only support one-to-one mapping. In a lot of VC application scenarios, it is desirable to obtain a VC systems supporting more than one mapping (Kameoka et al., 2018). Many attempts (Toda et al., 2007, Saito et al., 2011, Sun et al., 2016b, Huang et al., 2021, Chen et al., 2019, Saito et al., 2020, Liu et al., 2018, 2020b, Kaneko et al., 2019b, Kameoka et al., 2018) have been made to allow VC systems to support this, for example, one-to-many, many-to-one or many-to-many mappings.

Apart from the above cases, there exists a specific case within any-to-any VC systems, one-shot VC systems (Chou and Lee, 2019, Wu et al., 2020, Tang et al., 2022, Qian et al., 2019, Lu et al., 2019, Chen et al., 2021b, Wang et al., 2020b, Tang et al., 2022, Ishihara and Saito, 2020, Wang et al., 2021). The objective of these systems is to allow a VC system to convert a speech signal to an unseen target speaker that only one test sample of this target speaker is available. Correspondingly, they named a type of VC tasks as one-shot VC tasks.

Text-Dependent vs. Text-Free

Early VC systems,such as parallel data systems, usually do not require text transcriptions for training. As introduced before, the parallel data constraint simplifies the VC task to a local feature mapping task. In this task, A feature vector from a source speaker is mapped to an aligned feature vector of a target speaker. Each feature frame is modeled independently, without considering contextual information in neighbouring feature frames. The VC models learn a mapping function from a source speaker feature vector to a target speaker feature vector through supervised training.

Without parallel data, it is difficult to train a feature mapping function in a supervised form. Therefore a new training paradigm of VC models is needed. An obvious solution is to build a cascade system (Huang et al., 2020a,c, Zhang et al., 2020) of an ASR model (Watanabe et al., 2017, Dong et al., 2018) and a TTS model (Li et al., 2019, Shen et al., 2018). The ASR model transcribes an input speech signal into texts and the TTS generates a speech signal from the texts. Many VC systems (Liu et al., 2018, 2020b, 2021, Zhao et al., 2021) have been developed following this solution. This thesis names VC systems that are built with ASR models as text-dependent systems. One drawback of these systems is that they are highly dependent on the good performance of the ASR models. ASR models usually require large training datasets (Deng and Li, 2013) and might suffer from degradation because of the mismatch of training and test data.

To release the constraints on text transcriptions, many text-free VC systems (Kaneko and Kameoka, 2017, Kameoka et al., 2018, Qian et al., 2019, Chou and Lee, 2019) have been studied. Text-free VC models can be categorised into three classes: auto-encoder based models (Chou et al., 2018, Chou and Lee, 2019, Van Den Oord et al., 2017, Qian et al., 2019, Cho et al., 2019, Wu et al., 2020, Tang et al., 2022), GAN based models (Kaneko and Kameoka, 2017, Kaneko et al., 2019a, 2020, Kameoka et al., 2018, Kaneko et al., 2019b) and SSL based models (Huang et al., 2022b, 2021). Further details will be introduced in Section 2.3 and Section 2.4.

2.1.3 A Categorisation of Non-Parallel VC Models

This thesis mainly focuses on non-parallel data VC models. A categorisation (Yi et al., 2020) of non-parallel data VC models is illustrated in Figure 2.3. Non-parallel data VC models can be categorised into two classes: encoder-decoder based models (Huang et al., 2020a, Liu et al., 2018, 2020b, 2021, Zhao et al., 2021) and GAN based models (Kaneko and Kameoka, 2017, Kaneko et al., 2019a, 2020, Kameoka et al., 2018, Kaneko et al., 2019b). Encoder-decoder based models will be introduced in Section 2.4 and GAN based models will be introduced in Section 2.5.

The remainder of this chapter is organised as follows:

- Section 2.2 reviews speech analysis and reconstruction methods for VC.
- Section 2.3 reviews parallel data based VC models.
- Section 2.4 reviews encoder-decoder based VC models.
- Section 2.5 reviews GAN based VC models.

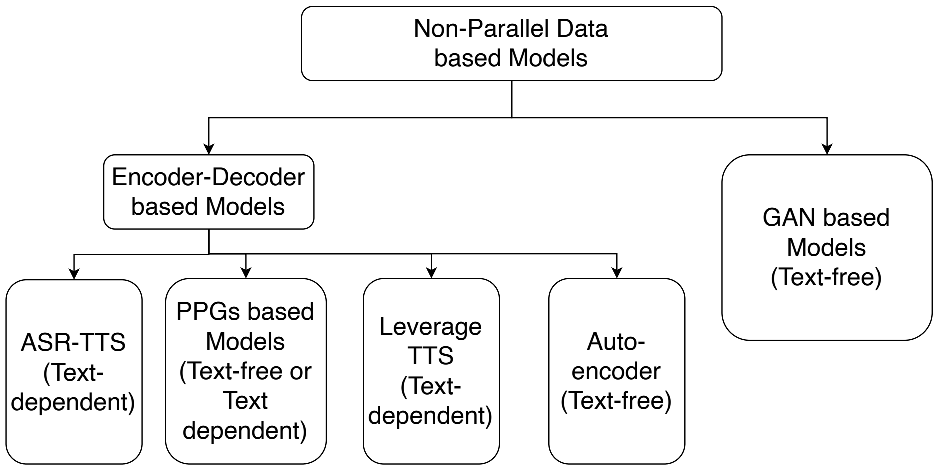
Non-Parallel Data based Models
Encoder-Decoder based Models
GAN based Models (Text-free)
ASR-TTS (Text-dependent)
PPGs based Models (Text-free or Text dependent)
Leverage TTS (Text-dependent)
Auto-encoder (Text-free)

Fig. 2.3 A categorisation of non-parallel data based VC models (Yi et al., 2020), where "PPGs" denotes phoneme posterior grams.

- Section 2.6 introduces datasets, challenges, evaluation metrics and current challenges for VC.

2.2 Speech Analysis and Reconstruction

This section introduces methods used for the speech analysis step and the reconstruction step in the workflow illustrated in Figure 2.2. Speech analysis and reconstruction methods are also named as vocoder methods in (Kawahara et al., 1999, 2008, Morise et al., 2016). They have been studied widely in speech communication and speech synthesis (Sisman et al., 2020).

Reconstruction methods can be further categorised into two types (Sisman et al., 2020), parametric vocoder methods (Kawahara et al., 1999, 2008, Morise et al., 2016) and neural vocoder models (van den Oord et al., 2016, Yamamoto et al., 2020, Kalchbrenner et al., 2018, Kong et al., 2020, Kumar et al., 2019). As discussed in (Sisman et al., 2020) that "the parameters of parametric vocoders are manually tuned according to some over-simplified assumptions in signal processing". Because of this, parametric vocoder methods are likely to suffer from a low voice quality issue. Neural vocoder models apply neural networks to generate waveforms from speech features. Typically neural vocoder models are independently trained and they are employed for the reconstruction step in VC systems.

2.2.1 Speech Analysis and Parametric Vocoder Methods

The objective of the speech analysis step is to extract frame-wise features from speech signals, which can be regarded as "some form of intermediate representation for effective manipulation or modification with respect to the acoustic properties of speech" (Sisman et al., 2020).

Most speech analysis methods are based on the source filter model (Chiba and Kajiyama, 1958). The source filter model is a physical model that assumes speech signals are produced by passing a source signal through a filter. The source filter model assumes the independence between the source signal and the filter. As discussed in (Mohammadi and Kain, 2017), from the perspective of the human body, the source signal is produced from vocal cord movements and the vocal tract is regarded as a filter.

The source-filter model can be implemented in many ways. STRAIGHT (Kawahara et al., 1999) and its extension TANDEM-STRAIGHT (Kawahara et al., 2008) are two commonly used speech analysis and reconstruction methods. They extract spectral features, fundamental frequency, and aperiodicity from a speech signal (Kawahara et al., 2008, Mohammadi and Kain, 2017). Based on STRAIGHT, WORLD (Morise et al., 2016) further improved the computational efficiency of STRAIGHT.

As introduced in (Sisman et al., 2020), further feature extraction processes can be used to enhance features extracted from speech analysis methods. Logarithm and Mel-scale are two further processes that are commonly used for spectral features. Spectral features can also be transformed into cepstral space, for example, Mel-cepstral coefficients (MCEP) are commonly used in VC systems (Kaneko and Kameoka, 2017, Kameoka et al., 2018, Liu et al., 2018).

The STRAIGHT and WORLD methods have been shown effective in many early VC works (Abe et al., 1990, Shikano et al., 1991, Zen et al., 2008) and were used in several recent VC works (Kameoka et al., 2018, Kaneko et al., 2019b,a, 2020, Kaneko and Kameoka, 2017). However, as discussed in (Sisman et al., 2020), as a reconstruction method, they typically suffer from a low voice quality issue.

2.2.2 Neural Vocoder Methods

Instead of using parametric vocoder methods, many VC systems applied neural vocoder models to improve the voice quality. In VC systems, neural vocoder models are applied as a reconstruction method. Before this, the neural vocoder models need to be trained independently on a speech corpus (Ito.o, 2017, Yamagishi et al., 2019, Kominek and Black, 2004) with high quality recordings. From the perspective of model architecture, neural vocoder models can be roughly categorised into two types: autoregressive models (van den Oord et al., 2016, Kalchbrenner et al., 2018) and non-autoregressive models (Kong et al., 2020, Yamamoto et al., 2020, Kumar et al., 2019, Ping et al., 2020, Prenger et al., 2019). Further details of these two types of neural vocoder models are introduced in the following.

Autoregressive Neural Vocoder Models

Given a sequence of waveform samples $\mathbf{x} = \{x_t | t = 1, ..., T\}$, where $\mathbf{x} \sim \mathcal{X}$. The training objective of neural vocoder models is to maximise the log-likelihood of data $logp(\mathbf{x}|\mathbf{Z})$, where $\mathbf{Z}$ denotes speech features, such as log-mel-spectrograms. Autoregressive neural vocoder models (van den Oord et al., 2016, Kalchbrenner et al., 2018) further factorise $p(\mathbf{x})$ as a product of conditional probabilities $p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1})$. The training objective can be written as follows.

$$\mathcal{L}_{ar} = \mathbb{E}_{\mathcal{X}}\{-\sum_{t=1}^{T} logp(x_t | x_1, ..., x_{t-1}, \mathbf{Z})\}. \tag{2.1}$$

An example of autoregressive neural vocoder models (van den Oord et al., 2016, Kalchbrenner et al., 2018) is WaveNet (van den Oord et al., 2016). It has become one of the popular neural vocoder models, because it can produce high quality speech signals. It has been found outperforming a parametric vocoder method, STRAIGHT, in a subjective evaluation (van den Oord et al., 2016).

Non-Autoregressive Neural Vocoders

Although autoregressive neural vocoder models can obtain good performance, they are likely to suffer from an issue of their low decoding speeds (Oord et al., 2018). In order to speed up the decoding process, there have been several attempts to use non-autoregressive models (Kong et al., 2020, Yamamoto et al., 2020, Kumar et al., 2019, Ping et al., 2020, Prenger et al., 2019, Oord et al., 2018).

Non-autoregressive vocoder models directly map speech features $\mathbf{Z}$ to waveform samples $\mathbf{x}$. Different from autoregressive vocoder models that generate one waveform sample conditioning on previously generated samples, non-autoregressive vocoder models generate waveform samples only conditioning on speech features. The objective can be written as follows:

$$\mathcal{L}_{nonar} = \mathbb{E}_{\mathcal{X}}\{-logp(\mathbf{x}|\mathbf{Z})\}. \tag{2.2}$$

Non-autoregressive neural vocoder models usually build models upon generative models such as GAN (Goodfellow et al., 2014) and Flow (Rezende and Mohamed, 2015). Several

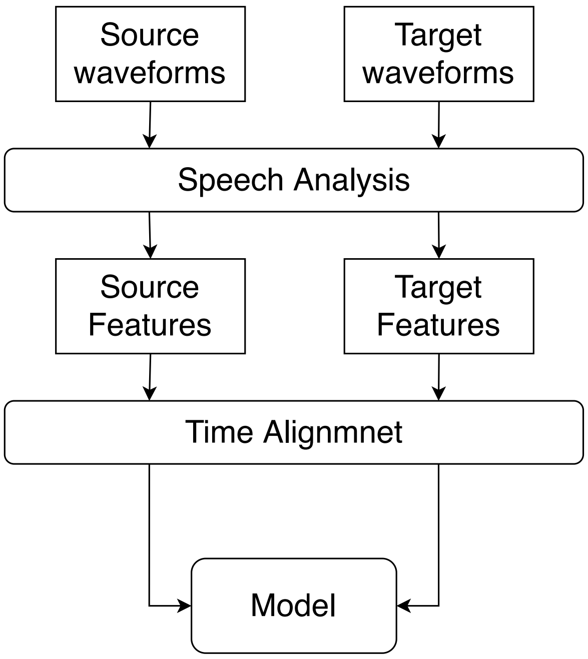

Fig. 2.4 A typical workflow of the training phase of a parallel data VC system (Mohammadi and Kain, 2017).

examples of GAN based vocoder models are Parallel WaveGAN (PWG) (Yamamoto et al., 2020), HifiGAN (Kong et al., 2020) and MelGAN (Kumar et al., 2019). Two examples of Flow based vocoder models are WaveFlow (Ping et al., 2020) and WaveGlow (Prenger et al., 2019).

It is worthy to note that HifiGAN has become a widely used vocoder model. In (Kong et al., 2020), HifiGAN has been compared with WaveNet and WaveGlow from three aspects: naturalness, decoding speeds and model sizes. HifiGAN has been found outperforming the other two vocoder models on naturalness on the LJSpeech dataset (Ito.o, 2017). It has a higher decoding speed and a smaller model size than the other two models.

2.3 Parallel Data VC Models

The VC model is a part of the workflow, which has been illustrated in Figure 2.2. This section briefly introduces models that have been used in parallel data VC systems. For simplicity, these models are to be referred to as parallel data models.

Although these models require parallel data for training, parallel speech samples are not illustrated in Figure 2.2. For a better illustration of the training of parallel data VC models, Figure 2.4 illustrates a workflow of the training phase of these systems. The parallel data

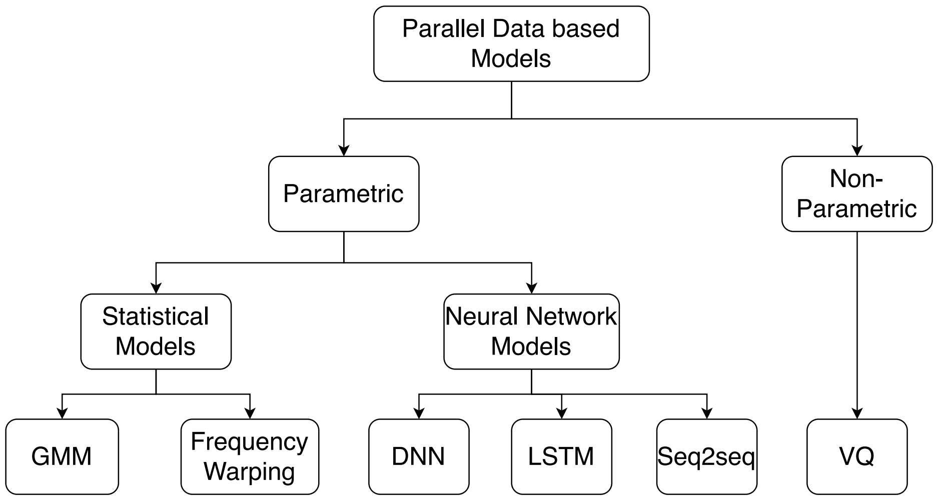

Fig. 2.5 A categorisation of parallel data VC models.

pair is composed of a source speech signal and a target speech signal. Speech features are extracted from both signals through the speech analysis process.

The source features are commonly a sequence of feature vectors, as well as the target features. As discussed in (Mohammadi and Kain, 2017), they usually have different durations. Thus a time alignment method is needed to align corresponding feature frames between them. Dynamic Time Warping (DTW) (Bellman and Kalaba, 1959) is commonly used in parallel data VC systems (Abe et al., 1990, Kain and Macon, 1998). After time alignment, the aligned source and target features are fed to the VC model for training.

Parallel data VC models can be implemented in various ways. A categorisation of them is illustrated in Figure 2.5. Generally, as summarised in (Sisman et al., 2020), these models can be categorised into two classes: parametric models (Kain and Macon, 1998, Stylianou et al., 1998, Toda et al., 2001, Zeng and Yu, 2010, Sundermann and Ney, 2003, Sundermann et al., 2004, Eide and Gish, 1996, Erro et al., 2012, Desai et al., 2009, Valentini-Botinhao et al., 2015, Xie et al., 2014, Sun et al., 2015, Ming et al., 2016) and non-parametric models (Abe et al., 1990, Shikano et al., 1991, Mouchtaris et al., 2007). The parametric models "make assumptions about the underlying statistical distributions of speech features and their mapping" (Sisman et al., 2020).

Non-parametric models typically utilise VQ to built a mapping function. Parametric models can be further categorised into two types (Sisman et al., 2020): statistical models (Kain and Macon, 1998, Stylianou et al., 1998, Toda et al., 2001, Zeng and Yu, 2010, Sundermann and Ney, 2003, Sundermann et al., 2004, Eide and Gish, 1996, Erro et al., 2012) and neural network models (Desai et al., 2009, Valentini-Botinhao et al., 2015, Xie et al., 2014, Sun et al., 2015, Ming et al., 2016). Gaussian Mixture Model (GMM) based models

(Kain and Macon, 1998, Stylianou et al., 1998, Toda et al., 2001, Zeng and Yu, 2010) and frequency warping based models are two commonly used statistical models. As for neural network models, several commonly used methods are Deep Neural Network (DNN) (Desai et al., 2009, Valentini-Botinhao et al., 2015), Long Short Term Memory (LSTM) (Xie et al., 2014, Sun et al., 2015, Ming et al., 2016) and Sequence-to-sequence (Seq2seq) (Zhang et al., 2019a, Tanaka et al., 2019).

2.3.1 Non-parametric Models

VQ, as a data compression technique (Gray, 1984), has been introduced in several early VC models (Abe et al., 1990, Shikano et al., 1991, Arslan and Talkin, 1997). VQ represents data with a finite set of vectors, this set is named as the codebook of VQ. Given an input feature vector, VQ finds the nearest neighbour codebook vector based on the Euclidean distance function. Then the nearest neighbour codebook vector is to represent the input feature vector.

Specifically, codebooks are built for both source and target speakers from data of source and target speakers (Abe et al., 1990). Let the codebook of the source speaker be $\mathbf{E}_{src} = \{\mathbf{e}^i_{src} \in \mathbb{R}^D | i = 1,...,K\}$ and the codebook of the target speaker be $\mathbf{E}_{tgt} = \{\mathbf{e}^i_{tgt} \in \mathbb{R}^D | i = 1,...,K\}$. The codebook of the source and target speakers share the same number of vectors K and the dimensionality of vectors D. The codebook vectors of the two speakers are aligned by histograms (Abe et al., 1990), which means a source codebook vector $\mathbf{e}^i_{src}$ is aligned with the ith vector $\mathbf{e}^i_{tgt}$ in the target codebook.

The process of the mapping function is introduced as follows. Given a source feature vector $\mathbf{x} \in \mathbb{R}^D$, VC models find the index m of nearest neighbour vector from the source codebook.

$$m = \underset{i \in [1,K]}{\operatorname{argmin}} ||\mathbf{x} - \mathbf{e}^i_{src}||_2. \tag{2.3}$$

Then the mth codebook vector in the codebook of target speaker is used as the output $\mathbf{o}$ of the model.

$$\mathbf{o} = \mathbf{e}^m_{tgt}. \tag{2.4}$$

The VQ based VC model (Abe et al., 1990) is simple. However, it is likely to produce discontinuous speech samples (Mohammadi and Kain, 2017). In order to alleviate the discontinuity issue, fuzzy VQ based models (Shikano et al., 1991, Arslan and Talkin, 1997) have been proposed as an extension method of the VQ based model. Instead of only use one target codebook vector as the output of the model (referring to Equation 2.4), the fuzzy VQ based models use a weighted sum over all target codebook vectors as the output of the model.

$$\mathbf{o} = \sum_{m=1}^{K} w_m \mathbf{e}_{tgt}^{m}, \tag{2.5}$$

where w_m is a weight parameter, which can be produced through a distance function. For the distance function, the Euclidean distance (Shikano et al., 1991) or an exponential decay function (Arslan, 1999) can be used.

2.3.2 Parametric Models

Statistical Models

GMM based VC models (Kain and Macon, 1998, Stylianou et al., 1998, Toda et al., 2001, Zeng and Yu, 2010) applied a GMM model to build a mapping function. GMM models are utilised to fit the distribution of speech features. The parameters of trained GMM models can be used to estimate the parameters of the mapping function.

Joint-Density Gaussian Mixture Model (JD-GMM) (Kain and Macon, 1998) is a commonly used GMM based model. Given a feature vector of source speaker $\mathbf{x}$ and the aligned feature vector of target speaker $\mathbf{y}$. The VC model is defined as a mapping function $\mathbf{o}_t = f(\mathbf{x}_t)$, where $\mathbf{o}_t$ and $\mathbf{x}_t$ are the output and input of the model. The mapping function $f(\mathbf{x}_t)$ can be written as:

$$f(\mathbf{x}) = \sum_{m=1}^{M} w_m^x (\mathbf{A}_m \mathbf{x} + \mathbf{b}_m), \tag{2.6}$$

where w_m^x, $\mathbf{A}_m$ and $\mathbf{b}_m$ are the parameters of the mapping function, M is the number of components in the GMM model.

JD-GMM used a GMM to fit the joint distribution $p(\mathbf{z}), \mathbf{z} = [\mathbf{x}^\top, \mathbf{y}^\top]^\top$ of the source features and target features. The JD-GMM model can be written as follows.

$$p(\mathbf{z}) = \sum_{m=1}^{M} w_m^z \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}_m^z, \boldsymbol{\Sigma}_m^z), \tag{2.7}$$

where

$$\boldsymbol{\mu}_m^z = \begin{bmatrix} \boldsymbol{\mu}_m^x \\ \boldsymbol{\mu}_m^y \end{bmatrix}, \tag{2.8}$$

$$\boldsymbol{\Sigma}_m^z = \begin{bmatrix} \boldsymbol{\Sigma}_m^{xx} & \boldsymbol{\Sigma}_m^{xy} \\ \boldsymbol{\Sigma}_m^{yx} & \boldsymbol{\Sigma}_m^{yy} \end{bmatrix}. \tag{2.9}$$

The JD-GMM model is trained using the EM algorithm (Dempster et al., 1977) with a maximisation likelihood estimation objective. The parameters of the trained JD-GMM model are used to estimate the parameters of the mapping function in a closed form.

GMM based VC models are likely to suffer from a over-smoothing issue, which means the converted speech signals of the models have a poor quality and being "buzzy-sounding" (Sisman et al., 2020), because they ignore the "time dynamic and spectral dynamic" (Sisman et al., 2020) information of speech features.

Vocal Tract Length Normalisation (VTLN) (Sundermann et al., 2004, Eide and Gish, 1996) is a commonly used frequency warping based model in VC systems (Sundermann and Ney, 2003, Eichner et al., 2004, Přibilová and Přibil, 2006). As discussed in (Mohammadi and Kain, 2017), frequency warping based VC models are difficult to "mimic very different voices" from source speakers. Further details of VTLN will be introduced in Section 3.2.

Neural Network Models

DNN based VC models (Desai et al., 2009, Valentini-Botinhao et al., 2015, Xie et al., 2014) employ a DNN model as a mapping function. They take in a source feature vector and map it to a target feature vector. The training objective is to minimise the L2 distance between the mapped features and the target features. Similar to GMM based models, DNN based models map the frames of the source features independently. They still ignore the dynamic information of speech features. In order to model time dynamic information, LSTM (Hochreiter and Schmidhuber, 1997) based VC models have been utilised. Compared to DNN based models, LSTM based models allow to learn the contextual information of frames of the source features. LSTM based VC models (Sun et al., 2015, Ming et al., 2016) have been shown outperforming DNN based VC models (Sun et al., 2015, Ming et al., 2016). Seq2seq based VC models (Zhang et al., 2019a, Tanaka et al., 2019) are built upon the encoder-attention-decoder architecture (Parikh et al., 2016). This architecture allows the models to map a sequence of source features to a sequence of features with a different duration. Hence, the Seq2seq based VC models do not require a time alignment process.

2.4 Encoder-Decoder based VC Models

As illustrated in Figure 2.3, encoder-decoder based VC models can further be categorised into four classes:

- ASR-TTS models (Huang et al., 2020a,c, Zhang et al., 2020);

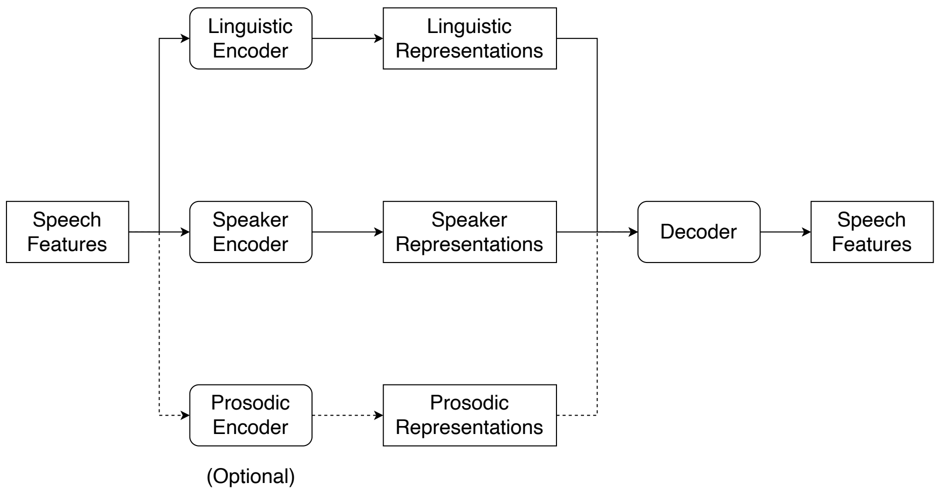

Fig. 2.6 A framework of encoder-decoder based VC models, based on the discussions in (Yi et al., 2020).

- Phoneme Posterior Grams (PPGs) based models (Liu et al., 2020b, 2021, Huang et al., 2022b);
- Leverage TTS models (Park et al., 2020, Huang et al., 2020b, 2021);
- Auto-Encoder based models (Qian et al., 2019, Chou and Lee, 2019, Chou et al., 2018, Wu et al., 2020, Tang et al., 2022, Cho et al., 2019).

For simplicity, this chapter names encoder-decoder based VC models as encoder-decoder based models. To better understand encoder-decoder based models, based on discussions in (Yi et al., 2020), this chapter summarises a common framework of encoder-decoder based models and illustrates it in Figure 2.6.

As introduced before, the objective of a VC system is to preserve the linguistic information and modify the speaker information and the prosodic information. The idea of encoder-decoder based models is to utilise encoders to extract intermediate representations of the linguistic information, the speaker information and the prosodic information from speech features, respectively. The encoders used for extracting these three types of information are named as linguistic encoder, speaker encoder and prosodic encoder, respectively. Then a decoder is employed to generate speech features from extracted representations.

Encoder-decoder based models can be implemented in various ways. Specifically, Table 2.1 presents a summary of recent encoder-decoder based models. This table shows the implementations for the three encoders and the decoder, respectively. Also it is worthy to note that some models do not employ a speaker encoder or a prosodic encoder. For example, among the listed models, the prosodic encoder is only employed in three models. It is

Table 2.1 A comparison of recent encoder-decoder based models, where "Linguistic", "Speaker" and "Prosodic" three columns show what implementations are used for the linguistic encoder, the speaker encoder and the prosodic encoder, respectively. "ASR-BNE" denotes automatic speech recognition bottleneck, "CNN" denotes convolution neural network, "PPGs based" denotes PPGs based VC model.

Model	Type	Linguistic	Speaker	Prosodic	Decoder
Taco-AR	PPGs based	PPGs	d-vector	No	Tacotron-2
BNE-VC	PPGs based	ASR-BNE	d-vector	Yes	LSTM
FastSpeech-VC	PPGs based	PPGs	None	Yes	FastSpeech
ASR-TTS	ASR-TTS	ASR	x-vector	No	TTS
S3PRL-VC	PPGs based	SSL	d-vector	No	Tacotron-2
AutoVC	Auto-Encoder	LSTM	x-vector	No	LSTM
AdaIN-VC	Auto-Encoder	CNN	CNN	No	CNN
VQ-WAE	Auto-Encoder	CNN	1hot	No	WaveNet

optional whether or not to use a prosodic encoder, because whether or not modeling the prosodic information by learning explicit representations is still a question, which will be discussed later. There are some models that do not use a speaker encoder, just using a speaker identity one-hot vector as the representations of the speaker information.

Auto-encoder based VC models have two main differences from the other encoder-decoder based VC models. Firstly, auto-encoder based VC models are text-free models but most of the other models are text-dependent models. Secondly, the encoder and the decoder of auto-encoder based models are trained together but most of the linguistic encoders of the other models are trained independently. Because of these differences, this section introduces auto-encoder based VC models independently, which will be at the end of this section. Before an introduction of auto-encoder based VC models, this section begins with introductions of various implementations of the three encoders. Then this section introduces developments of the implementations of the decoder. Next, this section introduces the training and inference of encoder-decoder based models.

2.4.1 Linguistic Encoders

From discussions in (Yi et al., 2020), this chapter summarises four ways to implement the linguistic encoder:

1. ASR models (Zhang et al., 2020, Huang et al., 2020a);

2. PPGs extractor (Liu et al., 2018, 2020c, Zhao et al., 2021);

3. Bottleneck (BNE) extractor (Liu et al., 2021);

4. SSL models (Huang et al., 2022b, 2021).

Except SSL models, all above implementations of the linguistic encoder depend on ASR models. Hence, these VC systems can be regarded as text-dependent systems.

ASR Models

ASR-TTS models (Zhang et al., 2020, Huang et al., 2020a) are a cascade system consisting of an ASR model and a TTS model. ASR-TTS models usually utilise an ASR model as the linguistic encoder, which transcribes an input speech signal to texts. Hence, transcribed texts are regarded as linguistic representations.

The Transformer ASR model (Dong et al., 2018) is commonly used in ASR-TTS models (Zhang et al., 2020, Huang et al., 2020a). It applied the Transformer model (Vaswani et al., 2017) to ASR. One drawback of applying an ASR model as the linguistic encoder is that the errors of the outputs of the ASR model might be propagated to converted speech signals, causing a mispronunciation issue.

Phoneme Posterior Grams Extractors

In order solve the mispronunciation issue, PPGs based VC models (Sun et al., 2016a, Liu et al., 2020b, Zhao et al., 2021, Sun et al., 2016a) replaced the ASR model with a PPGs extractor. The PPGs extractor extracts phoneme posterior grams from speech features. The duration of PPGs is equal to input speech features. Compared to texts, PPGs are frame-level phonetic representations and PPGs are supposed to contain more local phonetic information than texts. The PPGs extractor model is usually an acoustic model of a speaker-independent speech recognition system (Lee, 1988, Maas et al., 2017, Graves et al., 2013). One drawback of the PPGs extractor is that the training of PPG extractors usually depends on "the HMM-GMM/DNN-based phonetic alignments with acoustic features" (Liu et al., 2021). The errors of the alignments also could cause converted speech signals having a mispronunciation issue.

Bottleneck Representations Extractors

In order to solve the issue of PPGs, BNE-VC (Liu et al., 2021) proposed to utilise the bottleneck features of a Seq2seq ASR model (Watanabe et al., 2017) as the linguistic representations. For simplicity, this chapter names the bottleneck features of a Seq2seq ASR model as "ASR-BNE". As an extension model of PPGs based models, BNE-VC (Liu et al., 2021) can still be categorised into this class.

The Seq2seq ASR model (Watanabe et al., 2017) has an encoder-attention-decoder architecture (Parikh et al., 2016). The Seq2seq ASR model is trained using the hybrid CTC-Attention objectives (Watanabe et al., 2017). BNE-VC (Liu et al., 2021) used the encoder of a Seq2seq ASR model as the linguistic encoder. Different from the PPGs extractor that outputs posterior grams, the ASR-BNE extractor produces frame-wise dense feature vectors. It has been found that BNE-VC outperformed other PPGs based models (Liu et al., 2021).

Self-Supervised Learning Models

Different from above linguistic encoder implementations that require text transcriptions for supervised training, SSL models (Schneider et al., 2019, Baevski et al., 2019, 2020, Hsu et al., 2021) can be trained unsupervisedly. SSL models aim to learn representations from speech data unsupervisedly. The representations from SSL models usually contain various types of information of a speech signal, such as the linguistic information and the speaker information. As one SSL model, VQ-Wav2vec (Baevski et al., 2019) has been the first to be introduced to VC by (Huang et al., 2021). More recently, many SSL models have been exploited for VC (Polyak et al., 2021, Huang et al., 2022b, van Niekerk et al., 2022).

SSL models encode speech signals into frame-wise representations. As introduced before that text-free models typically apply disentanglement learning to remove the speaker information of a speech signal. In the encoder-decoder framework mentioned above, the linguistic encoder is supposed to preserve phonetic information and remove speaker information. Hence, whether a SSL model can preserve phonetic information and remove speaker information is relevant to its VC performance when applied as the linguistic encoder. This chapter refers the performance of preserving phonetic information and meanwhile removing speaker information to disentanglement learning performance.

SSL models can be implemented in various ways. VQ is a commonly used disentanglement learning method in SSL models (Baevski et al., 2019, Chung et al., 2020, Baevski et al., 2020). The disentanglement learning performance of SSL models have been studied in the SUPERB benchmark (Yang et al., 2021). This section presents part of the results from (Yang et al., 2021).

SUPERB benchmark (Yang et al., 2021) established various down-stream tasks for evaluating representations from SSL models. In this benchmark, representations are applied as the inputs to various down-streaming tasks, such as phone classification and speaker classification. The disentanglement learning performance of SSL models can be shown through the accuracy of the phone classification task and the speaker classification task.

Table 2.2 presents the phone classification and the speaker classification results of several SSL models. It is clear that SSL models can encode linguistic information and speaker

Table 2.2 The phone and speaker classification results from the SUPERB benchmark (Yang et al., 2021), where the bold result denotes the lowest speaker classification result.

SSL Model	Phone(%) ↑	Speaker(%) ↓
Wav2vec (Schneider et al., 2019)	71.31	75.47
VQ-Wav2vec (Baevski et al., 2019)	63.38	**24.76**
Wav2vec 2.0 (Baevski et al., 2020)	56.39	82.53
APC (Chung and Glass, 2020)	72.76	79.08
VQ-APC (Chung et al., 2020)	71.92	68.56
Mockingjay (Liu et al., 2020a)	67.51	97.22

information into representations. Among SSL models, the lowest speaker classification result was obtained by VQ-Wav2vec, which was only 24.76 %, which implies the VQ-Wav2vec model performed better on removing speaker information than other SSL models. Also, the VQ-Wav2vec model obtained a 63.38 % phone accuracy, which was not the highest result, but was close to the highest result 72.76%.

The objective of the S3PRL-VC (Huang et al., 2022b) is to study whether SSL models can be used as the linguistic encoder in VC systems and which SSL model can obtain the best VC performance. They set up an encoder-decoder framework based on the Taco-AR model (Liu et al., 2020b), which was the best-performing model in the VCC 2020 challenge (Yi et al., 2020). As shown in Table 2.1, the Taco-AR model utilised a PPGs extractor as the linguistic encoder and a Tacotron-2 model (Shen et al., 2018) as the decoder. The S3PRL-VC replaced the PPGs extractor with a SSL model and used the Tacotron-2 model as the decoder. The SSL based VC models were trained and tested on the VCC 2020 challenge dataset, following the evaluation setup of the challenge. The S3PRL-VC presented a comparison of the VC performance of various SSL based VC models.

From the results (Huang et al., 2022b), among SSL models, the VQ-Wav2vec model outperformed other SSL models on naturalness and speaker similarity on the VCC 2020 benchmark. Since the experiments conducted the same evaluation as the VCC 2020 challenge, the performance of SSL based VC models can be compared to the performance of the best-performing model, Taco-AR. It was found that SSL based VC models can not reach the same level of performance of the Taco-AR model.

The S3PRL-VC found that SSL models can be used as the linguistic encoder in VC systems. But several drawbacks of the S3PRL-VC can be summarised as follows.

- They compared SSL models with a weak linguistic encoder, which was a PPGs extractor model (Huang et al., 2022b). The PPGs extractor was trained on the TIMIT dataset (Garofolo et al., 1993), which contains only 5 hours data.

- The training datasets in the experiments and the training datasets used for training the Taco-AR model were different. The Taco-AR model was trained on an internal dataset with 130 hours data but the VCC 2020 dataset only contains 2 hours data.
- The S3PRL-VC only considered one decoder implementation, the Tacotron-2 model.

2.4.2 Speaker Encoders

As shown in Table 2.1, d-vector (Wan et al., 2018) and x-vector (Snyder et al., 2018) are two commonly used implementations for the speaker encoder. Besides, the one-hot vector of a speaker identity can be used as a speaker representation. The one-hot speaker identity vector can only be used for speakers within training data. Specifically, Taco-AR, BNE-VC and S3PRL-VC used d-vector and ASR-TTS (Huang et al., 2020a) used x-vector. There has been little study on the difference between implementations of the speaker encoder for VC.

2.4.3 Prosodic Encoders

The prosodic encoder has been utilised in several models (Liu et al., 2021, Zhao et al., 2021, Zhang et al., 2020). The ways they implemented the prosodic encoder are different. BNE-VC and FastSpeech-VC (Zhao et al., 2021) used fundamental frequencies (F0s) of a speech signal as the prosodic representations, the WORLD vocoder was used to extract F0s. (Zhang et al., 2020) used a neural network as the prosodic encoder. The neural network was trained together with the decoder. There has little been study on the difference of implementations of the prosodic encoder for VC.

2.4.4 Decoders

In VC systems, it is common to employ a TTS model (Shen et al., 2018, Li et al., 2019, Ren et al., 2019, 2020) as the decoder. ASR-TTS models (Huang et al., 2020a, Zhang et al., 2020) employed a TTS model as the decoder. The Transformer-TTS model (Li et al., 2019) is commonly used in these models. In addition, Tacotron-2 (Shen et al., 2018) is commonly used in PPGs based VC models (Liu et al., 2018, 2020b, 2021).

From the perspective of model architecture, TTS models can be categorised into two types: autoregressive and non-autoregressive. In TTS, there have been various comparison studies (Ren et al., 2019, 2020, Miao et al., 2020, Elias et al., 2021) on autoregressive and non-autoregressive models, in terms of their voice quality, decoding speeds and model sizes. FastSpeech (Ren et al., 2019) has become a popular non-autoregressive model. The FastSpeech model is built upon a stack of Transformer layers (Vaswani et al., 2017). As a

TTS model, the FastSpeech model has shown an advantage on the decoding speed over the Tacotron-2 model. However, it has been found that FastSpeech obtained worse naturalness performance than Tacotron-2 (Ren et al., 2019).

VC systems mainly focused on applying autoregressive TTS models, such as the Tacotron-2 model or the Transformer-TTS model. There have been a few works (Zhao et al., 2021, Toda et al., 2001) that applied non-autoregressive TTS models for the decoder. The FastSpeech model is commonly used in these VC systems. FastSpeech-VC (Zhao et al., 2021) built a VC model utilising the FastSpeech model. In the context of VC, a comparison study between the autoregressive decoders and the non-autoregressive decoders is needed.

2.4.5 Training and Inference

The training and the inference of encoder-decoder based VC models can be implemented in various ways. Generally speaking, the training objective is to reconstruct a sequence of speech features.

Most of the linguistic encoders were trained independently on large datasets such as the LibriSpeech dataset (Panayotov et al., 2015). Besides, the speaker encoders were usually independently trained on the VoxCeleb 1 dataset (Nagrani et al., 2017) or the VoxCeleb 2 dataset (Nagrani et al., 2020).

There are mainly two ways of implementations of the prosodic encoder. One is to use a parametric vocoder such as the WORLD method (Morise et al., 2016) to extract F0s (Liu et al., 2021, Zhao et al., 2021). The WORLD method does not require training. Another way is to build up a prosodic encoder neural network (Zhang et al., 2020), the neural network was trained with the prosodic encoder neural network together with the decoder, with the same training objective as the decoder.

The training of the decoder usually happens after the independent training of the encoders. The decoder training is usually conducted on a VC dataset such as VCC 2020 (Yi et al., 2020), LibriTTS (Zen et al., 2019) or VCTK (Yamagishi et al., 2019). The trained encoders are used to extract representations from speech signals on the VC datasets. These representations are to be used as the inputs to the decoders. The decoder takes in these representations and reconstructs a sequence of speech features. The training objective is usually the L2 or L1 distance between reconstructed speech features and input speech features.

At the inference phase, the inputs to the model are a sequence of source speech features. For a target speaker, the inputs can be a speaker identity one-hot vector or the speech features from the target speaker. The linguistic encoder and the prosodic encoder are used to extract representations from source speech features. The speaker encoder is used to extract the speaker representations from target features. The decoder takes in representations and generates a sequence of speech features.

2.4.6 Auto-Encoder based VC Models

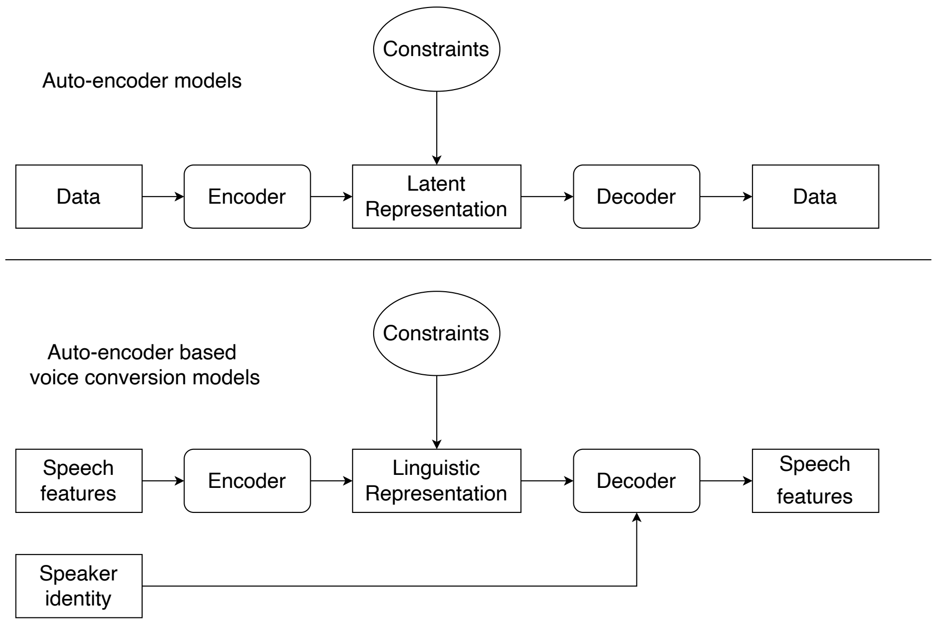

Fig. 2.7 A framework of auto-encoder models and a framework of auto-encoder based VC models.

The auto-encoder model (Le Cun and Fogelman-Soulié, 1987, Bourlard and Kamp, 1988, Hinton and Zemel, 1993) is a machine learning model that can be used for various applications, such as representation learning (Bengio et al., 2013, Zhuang et al., 2015, Li et al., 2017, Chorowski et al., 2019) and VC (Mohammadi and Kain, 2015, 2016, Qian et al., 2019, Chou et al., 2018). As introduced in (Bengio et al., 2013), the objective of the auto-encoder model is to learn representations through data compression, so that the redundant information of data can be removed in the latent representation.

The Variational Auto-Encoder (VAE) model (Kingma and Welling, 2014) and the VQVAE model (Van Den Oord et al., 2017) are two extension models of the auto-encoder model. The common attributes of these three auto-encoder models is that they applied constraints for compressing information. The upper part of Figure 2.7 illustrates a framework of various auto-encoder models, including an encoder and a decoder. The constraints are applied to the latent representation.

In the context of VC, the three auto-encoder models (Chou et al., 2018, Chou and Lee, 2019, Van Den Oord et al., 2017, Qian et al., 2019, Cho et al., 2019, Wu et al., 2020, Tang et al., 2022) have been widely studied. Most auto-encoder based VC models are regarded as text-free models. A framework of auto-encoder based VC models is illustrated in the lower part of Figure 2.7. As discussed in Chapter 1 that text-free models typically apply disentanglement learning methods for removing speaker information. The constraints in auto-encoder models can be regarded as disentanglement learning methods. The encoder of the auto-encoder plays a similar role as the above linguistic encoder, which is supposed to preserve linguistic information and remove speaker information.

Auto-Encoder

The auto-encoder model is composed of an encoder $Enc()$ and a decoder $Dec()$. For an input data $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{x} \sim \mathcal{X}$, the encoder encodes the input data to a latent representation $\mathbf{z} = Enc(\mathbf{x})$, where $\mathbf{z} \in \mathbb{R}^N$. Then the decoder takes in the latent representation and generates a reconstruction $\hat{\mathbf{x}} = Dec(\mathbf{z})$. The training objective is to minimise the L2 distance between the reconstruction and the input.

$$\mathcal{L}_{ae} = \mathbb{E}_{\mathcal{X}}\{||\hat{\mathbf{x}} - \mathbf{x}||_2\}. \tag{2.10}$$

The constraint applied in the auto-encoder model is to reduce the dimensionality of the latent representation. By applying this constraint, the information in the latent representation is compressed. In the context of VC, the latent representation is supposed to be invariant to the speaker-dependent information. There have been many works that built VC models (Qian et al., 2019, Chou et al., 2018) based on the auto-encoder model. AutoVC (Qian et al., 2019) applied an auto-encoder model for VC and explored the model on a one-shot VC task. It has been found that the latent dimensionality reduction can be used for removing speaker information (Qian et al., 2019), so that the model can convert a speech signal to an unseen target speaker. However, it requires careful tuning on the configurations of the latent dimensionality, which can be time-consuming and less robust (Chan et al., 2022).

Variational Auto-Encoder

The VAE model is an extension model of the auto-encoder model. The VAE model assumes the latent representation as a latent variable $\mathbf{z}$. The VAE model further assumes the prior distribution $p(\mathbf{z})$ and the posterior distribution $q(\mathbf{z}|x)$ of the latent variable are both a Gaussian distribution.

The constraint of VAE is to assume the prior distribution of $\mathbf{z}$ is a unit Gaussian distribution, $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$. The parameters of the posterior distribution can be estimated through the encoder. The encoder outputs the mean and standard deviation of the posterior distribution $q_\theta(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_\theta, \sigma_\theta)$, where μ_θ and σ_θ are the outputs of the encoder, θ denotes the parameters of the encoder. The training objective of VAE can be written as follows.

$$\mathcal{L}_{vae} = \mathbb{E}_{\mathcal{X}}\{||\hat{\mathbf{x}} - \mathbf{x}||_2 + D_{KL}(q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\}. \tag{2.11}$$

This training objective is composed of two terms, one is the reconstruction term, the other one is the KL divergence between the prior and the posterior.

When minimising the training objective, the KL divergence term is minimised. This implies that the posterior distribution is forced to move towards the prior distribution, which is a unit Gaussian distribution. Because the unit Gaussian distribution can be regarded as containing no information, this KL divergence term can be regarded as a constraint for compressing redundant information in the latent variable (Burgess et al., 2018).

There have been many works (Hsu et al., 2016, Chou and Lee, 2019) appling the VAE model in VC. However, their models usually suffered from an over-smoothing issue (Sisman et al., 2020), which means converted speech signals have a poor quality and being "buzzy-sounding" (Sisman et al., 2020).

In order to enhance the disentanglement learning performance of VAE based VC models, AdaIN-VC (Chou and Lee, 2019) further introduced the IN layer and the adaptive instance normalisation (AdaIN) layer (Huang and Belongie, 2017) to a VAE model. Specifically, they applied the IN layer in the encoder and the AdaIN layer in the decoder. The IN layer is a speaker normalisation method and the AdaIN layer is a speaker adaptation method. In their paper (Chou and Lee, 2019), applying these two layers was regarded as a disentanglement learning method. Further details of the IN layer and the AdaIN layer will be introduced in Chapter 3. The AdaIN-VC model gained success in a one-shot VC task.

Vector Quantised Variational Auto-Encoder

The VQVAE model is another extension model of the auto-encoder model. The VQVAE model assumes the latent representation to be discrete. The discretisation is implemented through introducing VQ into the model. VQ has been used for VC in early parallel data models (Abe et al., 1990, Shikano et al., 1991). Figure 2.8 illustrates a framework of the VQVAE based VC models. Since VQ is a data compression technique, the objective of using VQ in an auto-encoder model is to compress information in the latent representation.

Similar to the auto-encoder model, the VQVAE model has an encoder $Enc()$ and a decoder $Dec()$. In addition, the VQVAE model utilised a VQ block $VQ()$ as the bottleneck block. The latent representation can be obtained as $\mathbf{z}_e = Enc(\mathbf{x})$, then it is discretised by a

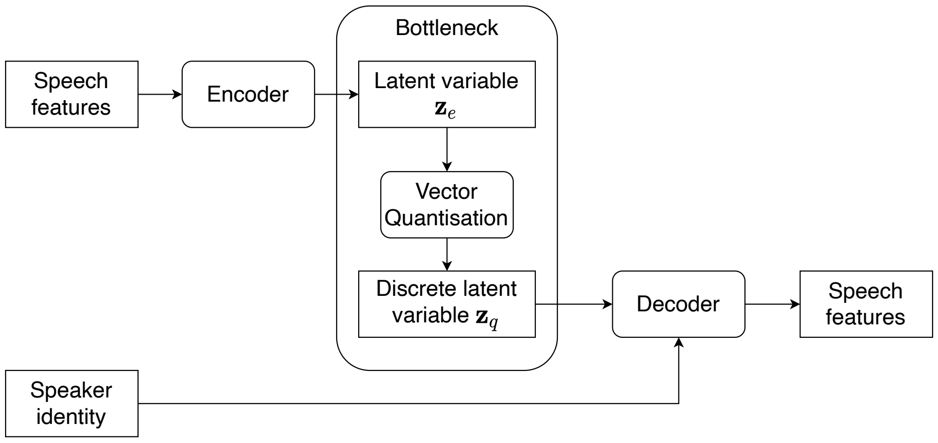

Fig. 2.8 A framework of VQVAE based VC models.

VQ block. The discrete latent representation can be obtained $\mathbf{z}_q = VQ(\mathbf{z}_e)$. The VQ process can be formulated as follows.

The VQ block has a codebook $\{\mathbf{e}_i | i = 1, ..., K\}$ containing K codebook vectors. VQ represents $\mathbf{z}_e$ with its nearest neighbour codebook vector $\mathbf{e}_m$ based on an Euclidean distance function. The index m of the nearest neighbour codebook vector can be obtained as follows.

$$m = \underset{j \in [1,K]}{\text{argmin}} ||\mathbf{z}_e - \mathbf{e}_j||_2. \tag{2.12}$$

Then the VQ block outputs the mth codebook vector, which is the discrete latent variable.

$$\mathbf{z}_q = \mathbf{e}_m. \tag{2.13}$$

The decoder takes in $\mathbf{z}_q$ and a speaker identity one-hot vector $\mathbf{s}$ then generates speech features $\hat{\mathbf{x}}$, which can be written as follows.

$$\hat{\mathbf{x}} = Dec(\mathbf{z}_q, \mathbf{s}). \tag{2.14}$$

The training objective of the VQVAE model can be written as follows.

$$\mathcal{L}_{vqvae} = \mathbb{E}_{\mathcal{X}}\{||\mathbf{x} - \hat{\mathbf{x}}||_2 + ||sg(\mathbf{z}_e) - \mathbf{z}_q||_2 + \lambda ||\mathbf{z}_e - sg(\mathbf{z}_q)||_2\}, \tag{2.15}$$

where λ is a weight and the $sg()$ function is the stop gradient function. The stop gradient function stops the backward gradient propagation at a tensor in a neural network training, which can be written as:

$$sg(\mathbf{x}) = \begin{cases} \mathbf{x}, forward \\ 0, backward \end{cases} \tag{2.16}$$

For VC, the discrete latent representation can be regarded as the linguistic representations. The VQ block is supposed to remove the speaker information in the latent representation $\mathbf{z}_e$.

In the VQVAE paper (Van Den Oord et al., 2017), the VQVAE model was evaluated on a VC task and it was found effective for VC. Following on from the VQVAE paper, many VQVAE based VC models (Wu and Lee, 2020, Wu et al., 2020, Tang et al., 2022, Wang and Borth, 2021) have been developed. The VQVAE based VC models obtained better performance on one-shot VC tasks than AutoVC and AdaIN-VC. However, the VQVAE based VC models were still not able to achieve the same level of performance of text-dependent VC models. In the VCC 2020, the best performing VQVAE based model was (Zhang, 2020b). (Zhang, 2020b) built a model upon the Vector Quantised WaveNet Auto-Encoder (VQ-WAE) model (Chorowski et al., 2019). The VQ-WAE model was proposed for learning phonetic representation from speech data and further was found effective for VC (Cho et al., 2019). Further details of the VQ-WAE model will be introduced later.

Not only for VC, the VQVAE model can also be used for unsupervised phonetic representation learning tasks. The objective of the phonetic representation learning is to learn representations for the phonetic information from speech data. The phonetic representations are supposed to be invariant to speaker-dependent information. The introduction of unsupervised phonetic representation learning will be the main focus of Chapter 3.

2.5 GAN based VC Models

For simplicity, this chapter names GAN based VC models as GAN based models. As a non-parallel data VC model, parallel samples are not available. Without parallel samples, the supervised training of a feature mapping model is difficult. To train a feature mapping model without parallel data samples, one solution is to employ the adversarial training of the GAN model (Goodfellow et al., 2014).

The objective of the GAN model is to train a generator that generates good quality data samples. The GAN model is composed of a generator and a discriminator. The objective of the generator is to generate data samples from a Gaussian noise. The objective of the discriminator is to distinguish whether a data sample is a real data sample or a generated (fake) data sample. The training process of the GAN model can be regarded as an adversarial game between the discriminator and the generator. The generator is trained to fool the discriminator that the generated sample is a real data sample. And the discriminator is trained

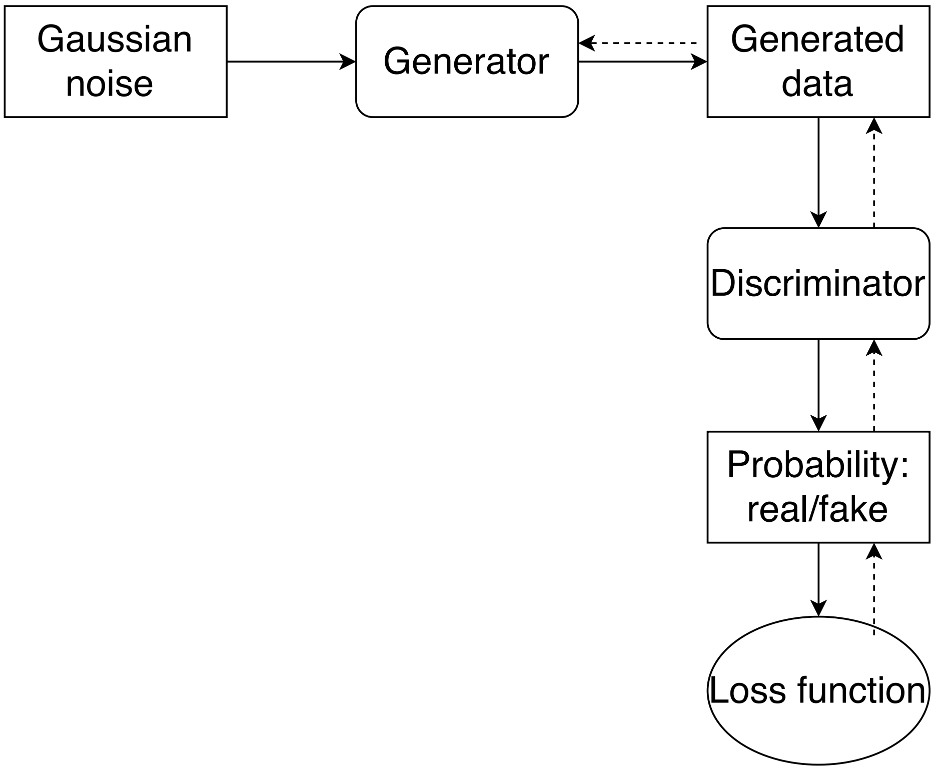

Fig. 2.9 An illustration of the adversarial training between the generator and the discriminator in GAN, where line arrows denote the forward process of the training and the dashed line arrows denote the backward process of the training.

not to be fooled. The adversarial training between the generator and the discriminator is illustrated in Figure 2.9.

The forward process of the adversarial training can be described as follows. The generated data is generated from the generator then is fed to the discriminator. The discriminator outputs a probability of it being a real sample. The training objective of the generator is to maximise this probability. In the backward process of the training, the loss is passed backward to the generator through the discriminator. The parameters in the generator are trained through the output of the discriminator. From the perspective of the generator, the discriminator can be regarded as an objective function.

Many GAN models have been developed for transfer the style of an image to another style, for example, the CycleGAN model (Zhu et al., 2017) and the StarGAN model (Choi et al., 2018). CycleGAN-VC (Kaneko and Kameoka, 2017), CycleGAN-VC2 (Kaneko et al., 2019a) and CycleGAN-VC3 (Kaneko et al., 2020) are three models that applied CycleGAN to VC. Besides, StarGAN-VC (Kameoka et al., 2018) and StarGAN-VC2 (Kaneko et al., 2019b) are two models that applied StarGAN to VC.

In GAN based VC models (Kaneko and Kameoka, 2017, Kaneko et al., 2019a, 2020, Kameoka et al., 2018, Kaneko et al., 2019b), the objective of the generator is to convert a sequence of source features to a target speaker. The discriminator is used as an objective function for the training of the generator. The CycelGAN-VC models and the StarGAN-VC models can be easily distinguished through the source-to-target mappings. The CycleGAN-VC models are one-to-one VC models. The StarGAN-VC models are many-to-many VC models.

This section begins with a brief introduction of the adversarial training objectives of the GAN model (Goodfellow et al., 2014). Because these adversarial training objectives were commonly used in GAN based VC models. Then this section introduces the CycleGAN-VC models and the StarGAN-VC models, respectively.

2.5.1 Adversarial Training Objectives of GAN

As introduced before, the GAN model (Goodfellow et al., 2014) is composed of a generator $G()$ and a discriminator $D()$. Let a data sample be $\mathbf{x}$, which can be sampled from a real data distribution $p(\mathbf{x})$. The generator generates a data sample $\hat{\mathbf{x}} = G(\mathbf{z})$ from a unit Gaussian variable $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. The discriminator is trained to distinguish between the generated data sample $\hat{\mathbf{x}}$ and a real data sample $\mathbf{x}$. The adversarial objective of the discriminator can be written as follows.

$$\mathcal{L}_D = \mathbb{E}_{p(\mathbf{x})}\{-D(\mathbf{x})\} - \mathbb{E}_{p(\mathbf{z})}\{1 - D(\hat{\mathbf{x}})\}. \tag{2.17}$$

Then the output of the discriminator can be used in an objective function to train the generator, so that the generator is forced to generate samples that fit well to the real data distribution. The objective function of the generator can be written as follows.

$$\mathcal{L}_G = \mathbb{E}_{p(\mathbf{z})}\{-D(\hat{\mathbf{x}})\}. \tag{2.18}$$

2.5.2 CycleGAN-VC Models

CycleGAN-VC (Kaneko and Kameoka, 2017) was the first to apply the CycleGAN model to VC. However, the CycleGAN-VC model suffered from a low voice quality and a low intelligibility. In order to improve the voice quality and the intelligibility, CycleGAN-VC2 (Kaneko et al., 2019a) and CycleGAN-VC3 (Kaneko et al., 2020) mainly focused on improving the model architecture of the generator. The three CycleGAN-VC models shared almost the same training objectives.

CycleGAN-VC Training Objectives

The adversarial training objectives were reused in the three CycleGAN-VC models. Moreover, in order to preserve the phonetic information of source features, CycleGAN-VC models followed the CycleGAN model that used a cycle-consistency objective and an identity-mapping objective.

A CycleGAN-VC model is composed of a generator and a discriminator, which supports one source-to-target mapping. Let two speakers be $\mathbf{s}_x$ and $\mathbf{s}_y$, respectively. Let the speech features of the two speakers be $\mathbf{X} \sim \mathcal{X}$ and $\mathbf{Y} \sim \mathcal{Y}$, respectively. Let two generators be $G_{x2y}() : \mathcal{X} \rightarrow \mathcal{Y}$ and $G_{y2x}() : \mathcal{Y} \rightarrow \mathcal{X}$. Let two discriminators be $D_{x2y}() : \mathcal{Y} \rightarrow \mathbb{R}$ and $D_{y2x}() : \mathcal{X} \rightarrow \mathbb{R}$.

The generator $G_{x2y}()$ takes in $\mathbf{X}$ and converts it to $\hat{\mathbf{Y}}$, this process can be written as follows.

$$\hat{\mathbf{Y}} = G_{x2y}(\mathbf{X}). \tag{2.19}$$

Similarly, the generator $G_{y2x}()$ takes in $\mathbf{Y}$ and converts it to $\hat{\mathbf{X}}$, the process can be written as follows.

$$\hat{\mathbf{X}} = G_{y2x}(\mathbf{Y}). \tag{2.20}$$

Following the adversarial objectives (Equation 2.18 and 2.17) of the GAN model, the adversarial objective functions of CycleGAN-VC can be written as:

$$\mathcal{L}^{D}_{cyc\_adv} = \mathbb{E}_{\mathcal{X}}\{-D_{x2y}(\mathbf{X})\} + \mathbb{E}_{\mathcal{Y}}\{-D_{y2x}(\mathbf{Y})\} - \mathbb{E}_{\mathcal{X}}\{1 - D_{x2y}(\hat{\mathbf{Y}}\} - \mathbb{E}_{\mathcal{Y}}\{1 - D_{y2x}(\hat{\mathbf{X}})\}, \tag{2.21}$$

$$\mathcal{L}^{G}_{cyc\_adv} = \mathbb{E}_{\mathcal{X}}\{-D_{x2y}(\hat{\mathbf{Y}})\} + \mathbb{E}_{\mathcal{Y}}\{-D_{y2x}(\hat{\mathbf{X}})\}. \tag{2.22}$$

Apart from the adversarial training objectives, the three CycleGAN-VC models used the cycle-consistency objective, which is from the CycleGAN model. The cycle-consistency objective feeds the converted speech features $\hat{\mathbf{Y}}$ as inputs to the $G_{y2x}()$. The generator $G_{y2x}()$ back converts the converted speech features $\hat{\mathbf{Y}}$ to $\mathbf{s}_x$. The back converted speech features can be obtained by $\mathbf{X}' = G_{y2x}(\hat{\mathbf{Y}})$. The cycle-consistency objective computes the L1 distance between the back converted speech features $\mathbf{X}'$ and the real speech features $\mathbf{X}$. The same process can be repeated for the generator $G_{x2y}()$. In total, the cycle consistency training objective can be written as follows.

$$\mathcal{L}_{cyc} = \mathbb{E}_{\mathcal{X},\mathcal{Y}}\{||\mathbf{X} - \mathbf{X}'||_1\} + \mathbb{E}_{\mathcal{X},\mathcal{Y}}\{||\mathbf{Y} - \mathbf{Y}'||_1\} \tag{2.23}$$

In addition, CycleGAN-VC used the identity-mapping objective. The identity-mapping objective uses $\mathbf{X}$ as the inputs to $G_{y2x}()$. The generator $G_{y2x}()$ reconstructs the input speech features $\mathbf{X}$. So that the reconstructed speech features can be obtained as $\mathbf{X}'' = G_{y2x}(\mathbf{X})$. The identity-mapping objective computes the L1 distance between $\mathbf{X}''$ and $\mathbf{X}$. The same process can be repeated for the generator $G_{x2y}()$. In total, the identity-mapping training objective can be written as follows.

$$\mathcal{L}_{id} = \mathbb{E}_{\mathcal{X}}\{||\mathbf{X}-\mathbf{X}''||_1\} + \mathbb{E}_{\mathcal{Y}}\{||\mathbf{Y}-\mathbf{Y}''||_1\}. \tag{2.24}$$

Overall, the whole training objectives of CycleGAN-VC can be written as follows.

$$\mathcal{L}^{G}_{cyc} = \mathcal{L}^{G}_{cyc\_adv} + \lambda_{cyc}\mathcal{L}_{cyc} + \lambda_{id}\mathcal{L}_{id}, \tag{2.25}$$

$$\mathcal{L}^{D}_{cyc} = \mathcal{L}^{D}_{cyc\_adv}, \tag{2.26}$$

where $\mathcal{L}_G$ and $\mathcal{L}_D$ are objective functions for the generators and the discriminators, respectively. λ_{cyc} and λ_{id} are hyper-parameters.

The Developments of the Generator Model Architectures

The generator model architecture of CycleGAN-VC (Kaneko and Kameoka, 2017) is fully based on 1D-convolution layers. As shown in Figure 2.10, the model architecture of CycleGAN-VC can be regarded as a cascade of two down-sampling convolution blocks, six bottleneck convolution blocks and two up-sampling convolution blocks. Each down-sampling block down-samples intermediate features by 2 times over time dimension. Each up-sampling block up-samples intermediate features by 2 times over time dimension.

CycleGAN-VC found that the CycleGAN model can be effectively adapted to VC. However, from the results of the CycleGAN-VC2 paper (Kaneko et al., 2019a), the Mean Opinion Score (MOS) of CycleGAN-VC stayed at 2.0. The voice quality of CycleGAN-VC was still poor. As discussed in the CycleGAN-VC2 paper (Kaneko et al., 2019a), one issue of CycleGAN-VC was that converted speech signals were over-smoothed (Kaneko et al., 2019a).

In order to solve this issue, CycleGAN-VC2 proposed a new generator model architecture, which was named as "2-1-2D". As shown in Figure 2.10, the "2-1-2D" model architecture implies the model uses 2D-convolution layers for the down-sampling and the up-sampling blocks. It used 1D-convolution layers for the bottleneck blocks. An insight from the CycleGAN-VC2 paper was that 2D-convolution layers would be more suitable for preserving

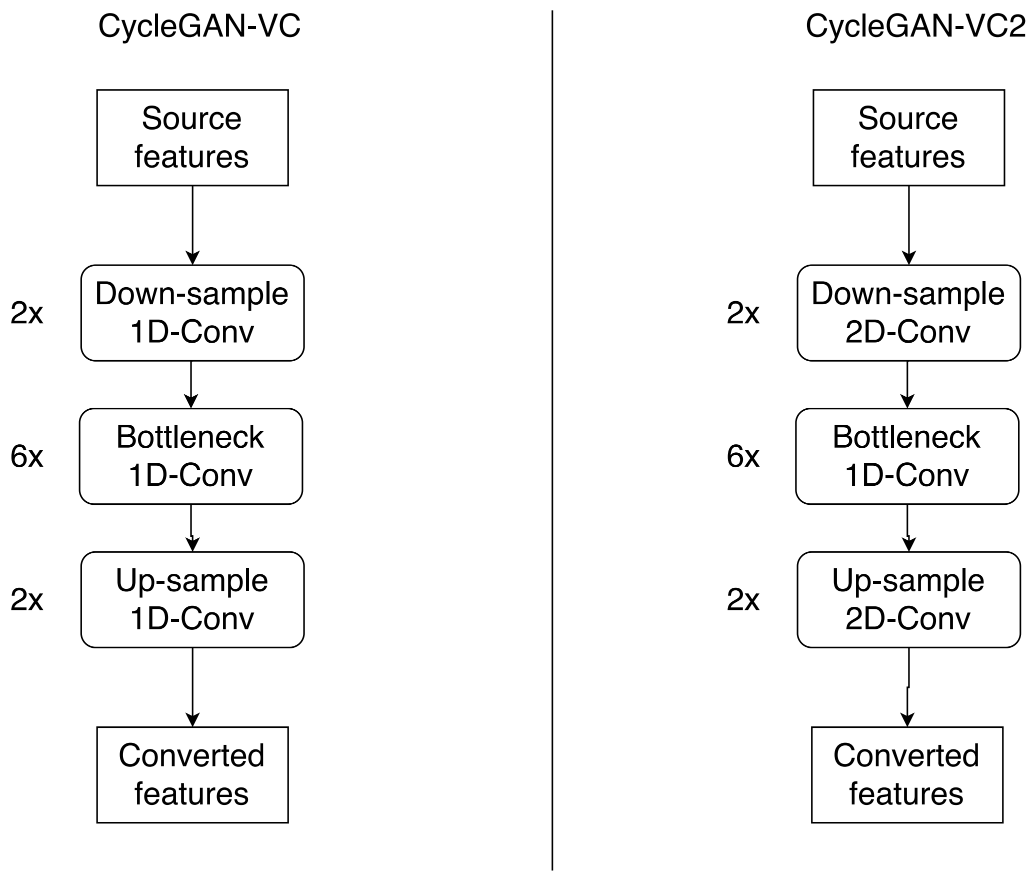

Fig. 2.10 A comparison of the generator model architecture of CycleGAN-VC and CycleGAN-VC2.

phonetic information than 1D-convolution layers. By using the "2-1-2D" generator model architecture, CycleGAN-VC2 successfully improved the MOS to 2.6.

In order to further improve performance, CycleGAN-VC3 proposed a new generator model architecture. Based on the generator model architecture of CycleGAN-VC2, CycleGAN-VC3 introduced the SPADE layer (Park et al., 2019). The SPADE layer was originally proposed for image style transfer. CycleGAN-VC3 kept the "2-1-2D" architecture and applied the SPADE layers into the generator. As shown in Figure 2.11, both the bottleneck blocks and up-sampling blocks apply the SPADE layers.

The SPADE layer is an extension method of the IN layer. It first normalises intermediate features. The normalised features are then adapted by a linear transform. The affine parameters of this linear transform are produced from the input speech features. The objective of applying the SPADE layers to the generator is to enhance the phonetic information preservation in the generator. However, since the affine parameters are produced from the input speech features, which could contain speaker-dependent information, CycleGAN-VC3

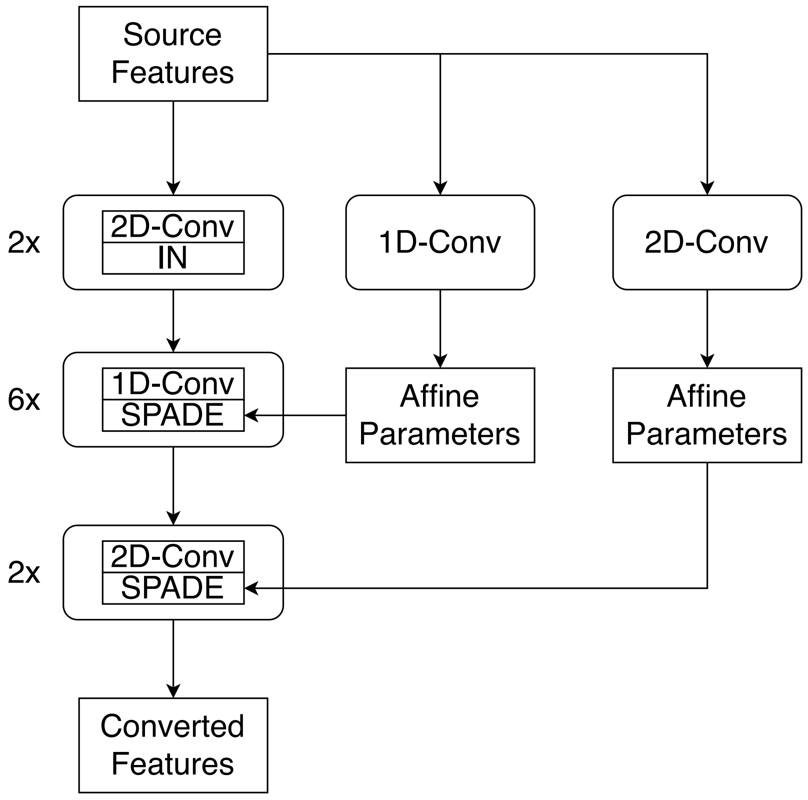

Fig. 2.11 An overview of the generator model architecture of CycleGAN-VC3.

might suffer from a low speaker similarity issue. From the results in the CycleGAN-VC3 paper (Kaneko et al., 2020), CycleGAN-VC3 obtained a 3.5 MOS. However, compared to CycleGAN-VC2, it could hardly improve the speaker similarity results.

2.5.3 StarGAN-VC Models

StarGAN-VC models (Kameoka et al., 2018, Kaneko et al., 2019b) support many-to-many mappings. Compared to CycleGAN-VC models, StarGAN-VC models can be more flexible in practical applications. Because of this, StarGAN-VC models can not depend on a specific source speaker or a specific target speaker. The speaker information of a target speaker is needed to be part of the inputs to the model. The generator takes in source features and the information of a target speaker and adapts intermediate features to the target speaker. Hence, StarGAN-VC models typically applied speaker adaptation methods.

Training Objectives of StarGAN-VC

The StarGAN-VC model is composed of a generator $G()$, a discriminator $D()$ and a speaker classifier $C()$. The generator takes in source speech features and a target speaker one-hot vector then generates converted speech features. Let source speech features be $\mathbf{X} \sim \mathcal{X}$, the source speaker one-hot vector be $\mathbf{s}_{src} \sim \mathcal{S}$ and the target speaker one-hot vector be $\mathbf{s}_{tgt} \sim \mathcal{S}$, the generator generates the converted speech features $\hat{\mathbf{X}} = G(\mathbf{X}, \mathbf{s}_{tgt})$.

The adversarial training objectives are similar to the above Equation 2.17 and 2.18. The adversarial objectives for the generator and the discriminator can be written as follows.

$$\mathcal{L}^{G}_{star\_adv} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{-D(\hat{\mathbf{X}}, \mathbf{s}_{tgt})\}, \tag{2.27}$$

$$\mathcal{L}^{D}_{star\_adv} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{-D(\mathbf{X}, \mathbf{s}_{src})\} - \mathbb{E}_{\mathcal{X},\mathcal{S}}\{1 - D(\hat{\mathbf{X}}, \mathbf{s}_{tgt})\}. \tag{2.28}$$

The objective of using the speaker classifier is to enhance the speaker similarity of the converted speech features $\hat{\mathbf{X}}$. The speaker classifier forces the converted speech features to get close to the target speaker. The speaker classifier objective function for the generator can be written as follows.

$$\mathcal{L}^{G}_{cls} = -\mathbb{E}_{\mathcal{X},\mathcal{S}}\{p(\mathbf{s}_{tgt}|\hat{\mathbf{X}})\}. \tag{2.29}$$

Also, the speaker classifier is trained with the source speech features:

$$\mathcal{L}^{C}_{cls} = -\mathbb{E}_{\mathcal{X},\mathcal{S}}\{p(\mathbf{s}_{src}|\mathbf{X})\}. \tag{2.30}$$

Apart from the adversarial objectives and the speaker classifier objectives, StarGAN-VC also used the cycle-consistency objective and the identity-mapping objectives that have been used in CycleGAN-VC models. For the cycle-consistency objective, the converted speech features $\hat{\mathbf{X}}$ are converted back to the source speaker. The back converted speech features can be obtained as $\mathbf{X}' = G(\hat{\mathbf{X}}, \mathbf{s}_{src})$. The cycle-consistency objective minimises the L1 distance between the back converted speech features $\mathbf{X}'$ and the source speech features $\mathbf{X}$.

$$\mathcal{L}^{G}_{star\_cyc} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{||\mathbf{X} - \mathbf{X}'||_1\}. \tag{2.31}$$

For the identity-mapping objective, the generator takes in the source speech features $\mathbf{X}$ and the source speaker one-hot vector $\mathbf{s}_{src}$ then reconstructs the source speech features $\mathbf{X}'' = G(\mathbf{X}, \mathbf{s}_{src})$. The identity-mapping objective function minimises the L1 distance between the reconstructed source speech features and the source speech features.

$$\mathcal{L}^{G}_{star\_id} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{||\mathbf{X}'' - \mathbf{X}||_1\}. \tag{2.32}$$

Overall, the training objectives of StarGAN-VC for the generator, discriminator and speaker classifier can be written as follows, respectively.

$$\mathcal{L}^{G}_{star} = \mathcal{L}^{G}_{adv} + \lambda_{cyc}\mathcal{L}^{G}_{star\_cyc} + \lambda_{id}\mathcal{L}^{G}_{star\_id} + \lambda_{cls}\mathcal{L}^{G}_{cls}, \tag{2.33}$$

$$\mathcal{L}^{D}_{star} = \mathcal{L}^{D}_{star\_adv}, \tag{2.34}$$

$$\mathcal{L}^{C}_{star} = \mathcal{L}^{C}_{cls}, \tag{2.35}$$

where λ_{cyc}, λ_{id} and λ_{cls} are hyper-parameters

Training Objectives of StarGAN-VC2

Although StarGAN-VC used a speaker classifier to enhance the speaker similarity, StarGAN-VC still suffered from a low speaker similarity issue (Kaneko et al., 2019b). It has been discussed in (Kaneko et al., 2019b), the speaker classification objective forces converted speech features to move far away from the decision boundaries between speakers. At the inference phase, for the source features that are close to the decision boundaries, StarGAN-VC would perform badly on speaker similarity.

In order to solve this issue, StarGAN-VC2 removed the speaker classifier. Then StarGAN-VC2 proposed to use not only the target speaker one-hot vector $\mathbf{s}_{tgt}$ but also the source speaker one-hot vector $\mathbf{s}_{src}$ as the inputs to the generator and the discriminator.

The adversarial training objectives of the StarGAN-VC2 model can be written as follows.

$$\mathcal{L}^{G}_{star2\_adv} = \mathbb{E}_{\mathcal{X},\mathcal{S}_{src}}\{-D(G(\mathbf{X},\mathbf{s}_{src},\mathbf{s}_{tgt}))\}, \tag{2.36}$$

$$\mathcal{L}^{D}_{star2\_adv} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{-D(\mathbf{X},\mathbf{s}_{tgt},\mathbf{s}_{src})\} - \mathbb{E}_{\mathcal{X},\mathcal{S}}\{1 - D(G(\mathbf{X},\mathbf{s}_{src},\mathbf{s}_{tgt}),\mathbf{s}_{src},\mathbf{s}_{tgt}))\}. \tag{2.37}$$

The cycle-consistency objective can be written as follows.

$$\mathcal{L}^{G}_{star2\_cyc} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{||\mathbf{X} - G(G(\mathbf{X},\mathbf{s}_{src},\mathbf{s}_{tgt}),\mathbf{s}_{tgt},\mathbf{s}_{src})||_1\}. \tag{2.38}$$

The identity-mapping objective can be written as follows.

$$\mathcal{L}^{G}_{star2\_id} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{||G(\mathbf{X},\mathbf{s}_{src},\mathbf{s}_{src}) - \mathbf{X}||_1\}. \tag{2.39}$$

Overall, the training objectives of StarGAN-VC2 for the generator and discriminator can be written as follows, respectively.

$$\mathcal{L}^{G}_{star2} = \mathcal{L}^{G}_{star2\_adv} + \lambda_{cyc}\mathcal{L}^{G}_{star2\_cyc} + \lambda_{id}\mathcal{L}^{G}_{star2\_id}, \tag{2.40}$$

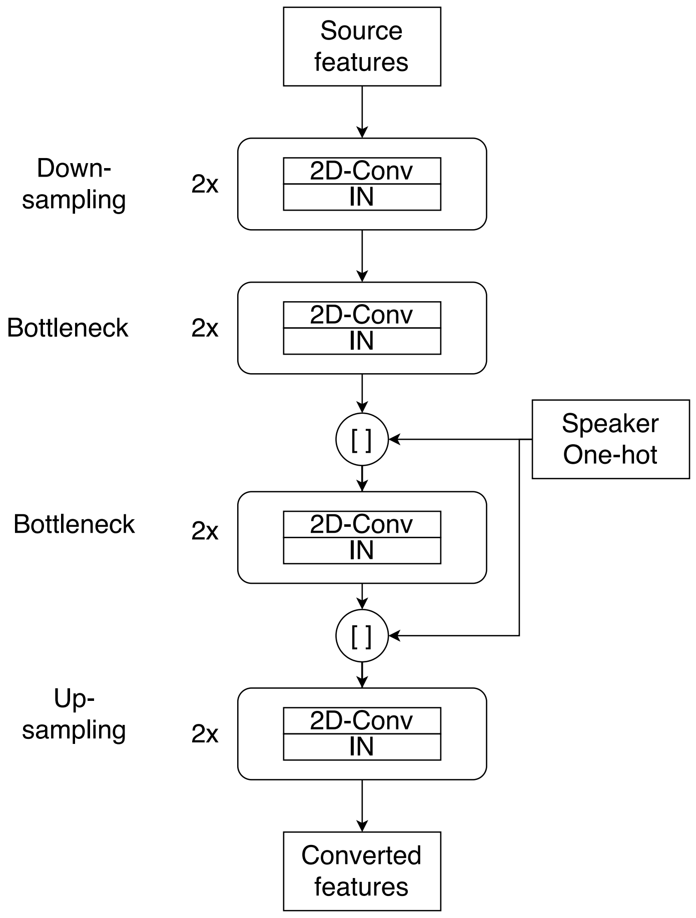

Fig. 2.12 An illustration of the generator of StarGAN-VC, where "[ ]" denotes concatenation.

$$\mathcal{L}^{D}_{star2} = \mathcal{L}^{D}_{star2\_adv}, \tag{2.41}$$

where λ_{cyc} and λ_{id} are hyper-parameters.

Speaker Adaptation Methods for StarGAN-VC Models

The generator of the StarGAN-VC model applied the IN layer as a speaker normalisation method and concatenated a speaker one-hot vector with intermediate features. Figure 2.12 illustrates the model architecture of the generator of StarGAN-VC. The generator is composed of two down-sampling 2D-convolution blocks, four bottleneck 2D-convolution blocks and two up-sampling 2D-convolution blocks. Each block uses an IN layer. For the last two bottleneck blocks and two up-sampling blocks, the output intermediate features of each block are concatenated with a speaker one-hot vector. The concatenation can be regarded as an additive speaker adaptation method (Kim and Yook, 2004).

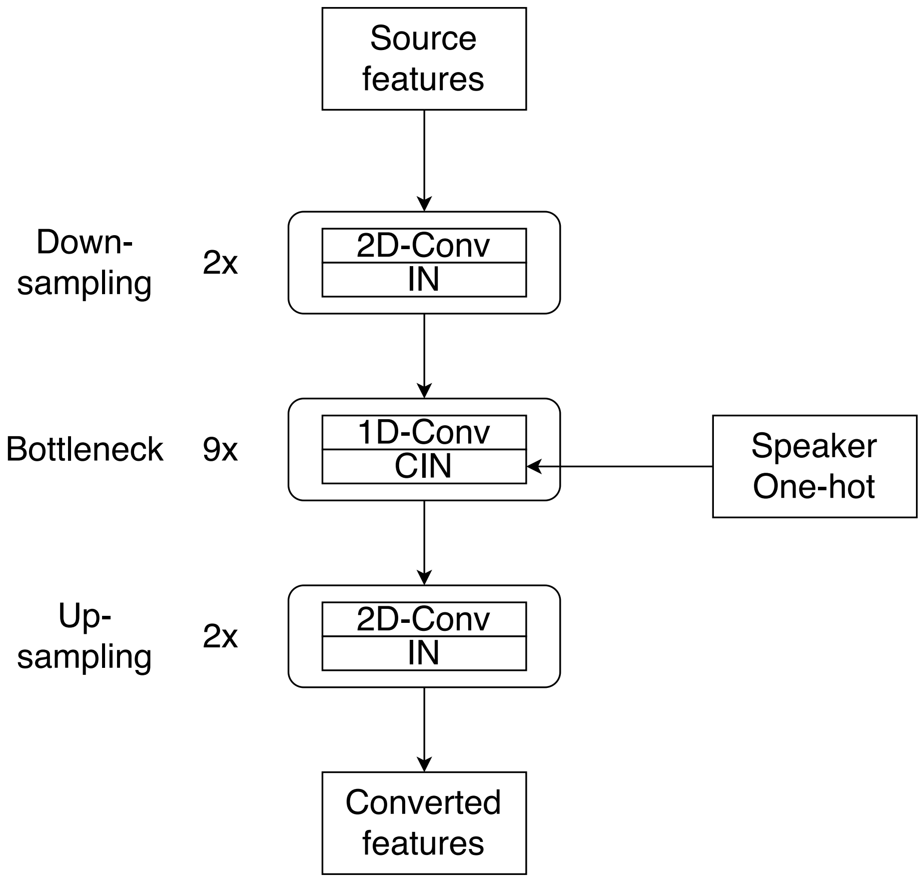

Fig. 2.13 An illustration of the generator of StarGAN-VC2.

As mentioned before, StarGAN-VC suffered from a low speaker similarity issue. StarGAN-VC2 improved the speaker similarity of StarGAN-VC by introducing CIN to the generator. CIN is a speaker adaptation method, which is an extension of IN. The CIN layer normalises intermediate features over time dimension, then adapts the normalised intermediate features to a target speaker through a linear transform. The affine parameters of the linear transform are produced from the speaker one-hot vector of the target speaker.

Figure 2.13 illustrates the model architecture of the generator of StarGAN-VC2. The generator model architecture follows the "2-1-2D" model architecture of CycleGAN-VC2. StarGAN-VC2 introduced CIN into the bottleneck blocks. In a bottleneck block, the CIN layer is placed between a convolution layer and an activation layer. From results in (Kaneko et al., 2019b), StarGAN-VC2 obtained an obvious improvement on the speaker similarity over StarGAN-VC.

Table 2.3 A comparison of datasets in VC, where "Parallel" denotes whether parallel samples are provided. "Speakers" denotes the number of speakers, "Hours" denotes the number of hours.

Dataset	Speakers	Hours	Type	Parallel
VCTK	109	44	Studio Reading	No
LibriTTS	2456	586	Audio-book	No
VCC 2020	14	2	Studio Reading	Only Test
VCC 2018	12	2	Studio Reading	Yes
VCC 2016	10	2	Studio Reading	Yes
ZS 2019	2	20	Studio Reading	Partial

Table 2.4 The details of the VCTK dataset.

Total duration	46 hours
Speakers	109
Female speakers	62
Male speakers	47
Utterances per speakers	400
Minutes per speakers	25

2.6 Datasets, Challenges, Evaluation Metrics and Current Problems

2.6.1 Datasets

Table 2.3 presents a comparison of datasets commonly used for VC. These datasets are from various sources and can be distinguished from whether it is a multi-speaker dataset, from the type of speech recordings and from whether they provide parallel data. Some of these datasets are from the challenges that are relevant to VC.

VCTK

VCTK has been commonly used in VC works (Chou and Lee, 2019, Chou et al., 2018, Wang et al., 2021). The VCTK dataset contains read speech recordings from 109 native English speakers with various accents. The speech samples are recorded by studio reading. The sampling rate of speech samples is 48000 Hz. The content of speech samples are from a set of newspaper texts. The VCTK dataset provides text transcriptions. The details of the VCTK dataset are presented in Table 2.4.

Table 2.5 The details of the subsets of the LibriTTS dataset, where "Hours" denotes the number of hours, "Speakers" denotes the number of speakers.

subset	Hours	Speakers
dev-clean	8.97	40
test-clean	8.56	39
dev-other	6.43	33
test-other	6.69	33
train-clean-100	53.78	247
train-clean-360	191.29	904
train-other-500	310	1160

LibriTTS

The LibriTTS dataset (Zen et al., 2019) is extracted from the LibriSpeech dataset (Panayotov et al., 2015), with utterances that contains significant background noises being excluded. In order to enhance the quality of the data for speech synthesis tasks and VC tasks, the LibriTTS dataset made several changes to the LibriSpeech dataset. Different from the LibriSpeech dataset that contains samples with a sampling rate of 16000 Hz, the LibriTTS dataset changed the sampling rate to 24000 Hz. According to (Zen et al., 2019), the LibriTTS dataset has the same subset splits as the LibriSpeech dataset. Since noisy utterances have been excluded, the amount of data of the LibriTTS dataset is 586 hours. The details of the subsets of the LibriTTS dataset are presented in Table 2.5. Despite the above attempts for enhancing the quality of speech samples, the data quality of the LibriTTS dataset is still less than ideal (Sisman et al., 2020).

VCC 2020, VCC 2018 and VCC 2016

The Voice Conversion Challenge (VCC) 2016 (Toda et al., 2016), VCC 2018 (Lorenzo-Trueba et al., 2018) and VCC 2020 (Yi et al., 2020) are three successive VC challenges. The organisers provided a dataset for each challenge. The speech samples of the VCC 2016 are studio reading speech from 10 US English speakers. The sampling rate is 16000 Hz. The training data samples are parallel, there are 162 parallel training samples for each speaker. There are also 54 parallel test samples for each speaker.

In the VCC 2018 dataset the sampling rate became 22050 Hz. The VCC 2018 dataset contains 12 speakers. For each speaker, the dataset contains 81 training samples and 35 test samples. The challenge is composed of two sub-tasks, parallel task and non-parallel task. For the parallel task, the dataset provides parallel samples for four source and four target

Table 2.6 The detailed information of the VCC 2020 dataset, where " Training Utts" and " Testing Utts" denote the number of training utterances and testing utterances per speaker, respectively. "Female" and "Male" denote the number of female speakers and male speakers, respectively.

Language	Female	Male	Training Utts	Testing Utts
English (Source)	2	2	70	25
English (Target)	2	2	70	25
German	1	1	70	25
Finnish	1	1	70	25
Mandarin	1	1	70	25

speakers. For the non-parallel task, the dataset provides non-parallel samples for four source and four target speakers. Both sub-tasks share the same four target speakers.

The VCC 2020 dataset contains four languages, including English, Finnish, German, Mandarin. The VCC 2020 challenge has two sub-tasks, intra-lingual VC and cross-lingual VC. Specifically, the English dataset contains four source English speakers and four target English speakers. The four English source speakers include two female and two male. The four English target speakers include two female and two male. For the other languages, there are two target speakers for each language, including one male and one female. The details of the VCC 2020 dataset are shown in Table 2.6. The VCC 2020 dataset provides parallel test data for English speakers. As shown in Table 2.6, there are 25 test samples for each English speaker.

ZS 2019 and ZS 2020

The Zero Speech (ZS) 2019 challenge (Dunbar et al., 2019) and the ZS 2020 challenge (Dunbar et al., 2020) are two successive challenges. The ZS 2019 and the ZS 2020 denote the dataset from them. The ZS2019 dataset and the ZS2020 dataset are almost identical (Dunbar et al., 2020). The task setup of these two challenges is almost the same. The challenge is a multi-task challenge, including a VC task and an AUD task. The ZS 2020 dataset includes two languages: the development language (English) and the test language (Austronesian). The dataset contains studio reading speech samples with a 16000 Hz sampling rate.

Table 2.7 presents details of the ZS 2020 dataset. The dataset of each language is composed of three subsets: train voice, train unit and test. The train voice subset contains speech samples of the target speakers for the VC task. The train unit subset contains speech samples for the AUD task. The test subset is for testing.

Table 2.7 The detailed information of the ZS2020 dataset, where "Spk" denotes the number of speakers, "Dur" denotes data duration, "Utt" denotes the number of utterances, "h" denotes hour, "min" denotes minute.

Language	subset	Spk	Dur	Dur (per spk)	Utt
English	Train Voice	2	4h40min	2h20min	3533
English	Train Unit	100	15h40min	2-11min	5941
English	Test	24	28min	1-4min	455
Austronesian	Train Voice	1	1h30min	-	1862
Austronesian	Train Unit	112	15h	-	15340
Austronesian	Test	15	29min	1-3min	405

2.6.2 Subjective Evaluation Metrics

VC is highly relevant to human perception. The performance of a VC system is commonly evaluated by subjective evaluations. The subjective evaluations of VC are usually conducted through human listening tests, which assess the quality of speech samples produced from VC systems. Human listening tests can be conducted on-line or off-line. For off-line listening tests, human evaluators need to be recruited and organised. For on-line listening tests, a commonly used platform is the Amazon Mturk platform (Amazon, 2005).

The listening tests usually are conducted by a question-answering form. A listening test usually is composed of multiple sessions of question-answering tasks. One session contains one or more speech samples, a question and a few answers for choosing. In one session, the human evaluator needs to listen to the speech samples then answer the question. The human evaluator needs to choose one of the provided answers based on their listening perception.

In order to reduce variances and biases, one session can be allocated to multiple human evaluators. The results are usually reported by VC systems. For one system, the results it obtained may be analysed through statistical methods, such as mean, standard deviation, percentage or confidence interval. In VC, subjective evaluations usually focus on three aspects of converted speech signals: naturalness, speaker similarity and intelligibility.

Naturalness

For naturalness evaluation, MOS (Chu and Peng, 2001) is one of the most commonly used metrics. In one MOS session, a human evaluator listens to one speech sample then answers a question. An example of questions for naturalness MOS could be "How natural is this recording?". Then the human evaluator needs to choose one from five provided answers. A set of examples (Mohammadi and Kain, 2017) of five answers is ’5-excellent’, ’4-good’,

’3-fair’, ’2-poor’, ’1-bad’. The higher score means the better naturalness. The mean and the 95% confidence interval of MOS results of a VC system can be reported as results.

Apart from MOS, preference tests such as AB test can be used as a metric (Machado and Queiroz, 2010). AB test is a metric based on comparing a system with another system. A human evaluator needs to listen to two samples then decide which one has better naturalness. The two samples should be from two systems respectively. The percentage of options that a model has obtained can be reported as a result. The higher percentage means the better naturalness. Compared to MOS which evaluates the naturalness through absolute rating, AB test is through comparison. AB test might not be suitable to compare multiple systems at the same time (Sisman et al., 2020).

Speaker Similarity

For speaker similarity evaluation, a commonly used metric is the ’same/difference’ test. In one session, a human evaluator needs to listen two speech samples: a converted speech sample and a reference sample from the target speaker. Then the human evaluator needs to answer a question. An example of question could be "Do you think these two samples could have been produced by the same speaker? Some of the samples may sound somewhat degraded/distorted. Please try to listen beyond the distortion and concentrate on identifying the voice. Are the two voices the same or different? You have the option to indicate how sure you are of your decision" (Lorenzo-Trueba et al., 2018). Then the human evaluator needs to choose one from the provided answers. VC works (Huang et al., 2022b, Lorenzo-Trueba et al., 2018, Yi et al., 2020) usually set up four answers: ’4-same, absolutely sure’, ’3-same, not sure’, ’2-different, not sure’ and ’1-different, absolutely sure’. There have also been VC works (Toda et al., 2016) that only set up two options: ’same’ and ’different’. For one VC system, the percentage of the obtained answers that contain ’same’ can be reported as a result. The higher percentage means the better speaker similarity.

MOS can also be used for evaluating the speaker similarity performance of VC systems. The setup of MOS evaluations for speaker similarity is basically the same as the setup of MOS evaluations for naturalness. In one session, the human evaluator needs to listen to two samples: a converted sample and a reference sample. Then the human evaluator needs to answer a question. An example of a question could be "How similar are these two samples?" Then the human evaluator needs to choose one from five provided answers: ’5-excellent’, ’4-good’, ’3-fair’, ’2-poor’, ’1-bad’. For a VC system, the mean and the 95% confidence interval of MOS results of a VC system can be reported as results.

The ABX test can be used as a metric for speaker similarity evaluations. Similar to AB test, ABX Test is a metric based on comparing a system with another system. A human evaluator needs to listen to three samples, including two test samples from two models and a reference sample from a target speaker. The target speaker of the two samples should be the same. Then the human evaluator needs to decide which one from the two test samples has higher speaker similarity to the reference sample. The percentage of a system that has been chosen can be reported as a result. Similar to the AB test, the ABX test is not suitable for comparing multiple systems at the same time.

Intelligibility

For intelligibility evaluations, human transcribed character error rate (CER) can be used. It has been used in the ZS 2019 challenge for evaluating the intelligibility of converted speech samples. In one session, a human evaluator needs to listen to a converted sample and transcribe the sample into texts. The CER metric is a metric based on the Levenshtein distance (Levenshtein et al., 1966). For one system, the CER of the transcribed text sentences can be reported as a result. The lower CER implies better intelligibility.

2.6.3 Objective Evaluation Metrics

Subjective evaluations have been conducted in most VC works. However, subjective evaluations usually are time consuming and expensive. Because organisation works, infrastructure building works and possibly funding are needed. Also subjective evaluations can not be easily scaled due to expenses.

In order to simplify the evaluations of VC, objective evaluation metrics have been developed and applied in VC works (Li et al., 2021, Liu et al., 2021, Huang et al., 2022b). Similar to subjective evaluations, the main focus of the objective evaluations also is three aspects: naturalness, speaker similarity and intelligibility. In addition, if parallel test samples are available, objective evaluations can be conducted for evaluating prosodic information or overall performance.

Naturalness

MOSNet (Lo et al., 2019) provided a solution to build a neural network model to estimate the MOS results of speech samples. The objective of the MOSNet model is to predict the MOS result of a speech sample.

Please note that MOSNet sometimes shows inconsistent results to subjective human listening tests. It has an advantage of convenience, however, is sometimes divergent with real human perception.

The MOSNet model takes in a speech sample and outputs the estimated MOS result. The MOSNet model is trained on pairs of speech samples and human annotated MOS results. The training objective of the MOSNet model is to minimise the L2 distance between the predicted MOS results and human annotated MOS results. In evaluations, MOSNet takes in a converted speech sample and outputs a MOS result. The mean and standard deviation of MOS results can be reported as results. The higher MOSNet results implies better naturalness.

MOSNet has been used in several VC works (Li et al., 2021, Kobayashi et al., 2021, Lin et al., 2021a, Zhang, 2020a) and have been shown highly relevant to human listening MOSs (Huang et al., 2022b).

Speaker Similarity

In order to evaluate the speaker similarity of converted speech samples, there have been several VC works (Huang et al., 2022b, Li et al., 2021, Polyak et al., 2021) using speaker identification or speaker verification as an evaluation metric. A trained speaker recognition model can be used to conduct a speaker identification task or a speaker verification task on converted speech samples. Two commonly used speaker recognition models are the x-vector model (Snyder et al., 2018) and the ECAPA-TDNN model (Desplanques et al., 2020).

The speaker recognition models can be trained on large datasets such as VoxCeleb 1 (Nagrani et al., 2017) and VoxCeleb 2 (Nagrani et al., 2020). Also they can be trained on datasets that are used to train VC models.

In a speaker identity task, a speaker recognition model takes in a converted speech sample and predicts a speaker identity. Given a list of converted speech samples, a speaker identification accuracy can be obtained as a result. The higher speaker accuracy implies the higher speaker similarity

For the speaker verification task, a verification list needs to be set up. A verification list contains positive sample pairs and negative sample pairs. The positive sample pair is composed of a converted speech sample and a real sample from a target speaker. The negative sample pair is composed of a converted speech sample and a real sample from another speaker. A speaker recognition model takes in speech samples and extracts speaker embeddings. Then the speaker verification task can be conducted and an equal error rate (EER) can be obtained as a result. The lower EER implies better speaker similarity.

Intelligibility

To evaluate the intelligibility of converted speech samples, there have been several VC works (Huang et al., 2020a, 2021, 2022b, Polyak et al., 2021, Li et al., 2021, Liu et al., 2021) that use an ASR model and conduct an speech recognition task on converted speech samples. This evaluation metric requires ground-truth text transcriptions of converted speech samples. The Transformer ASR model (Dong et al., 2018) and the Wav2vec 2.0 based ASR model (Baevski et al., 2020) are commonly used as the ASR model. The ASR model takes in a converted speech sample and decoding it to texts. With the ground-truth text transcriptions, a word error rate (WER) result can be obtained and used as a result. The lower WER result implies better intelligibility.

Parallel Data based Objective Metrics

Some VC datasets (Yi et al., 2020, Toda et al., 2016, Lorenzo-Trueba et al., 2018) provide parallel test samples, this enables VC works to conduct evaluation based on ground-truth speech signals.

The Mel-Cepstral Distortion (MCD) metric (Kubichek, 1993) evaluates the overall performance of converted speech signals (Huang et al., 2022b). The MCD computes the cepstral space distances between a converted sample and a ground-truth sample. The MCD first extracts the MCEPs features of a converted sample and a ground-truth sample. Then the two MCEPs features are aligned by the DTW (Bellman and Kalaba, 1959) algorithm, so that the lengths of the two MCEPs features become equal. Let the aligned converted MCEPs as $\mathbf{X}^{cvt} \in \mathbb{R}^{T \times C}$ and the aligned ground-truth MCEPs as $\mathbf{X}^{grd} \in \mathbb{R}^{T \times C}$, where T and C denote the time length and the number of MCEP coefficients. The MCD score can be computed as follows.

$$MCD(\mathbf{X}^{cvt}, \mathbf{X}^{grd}) = \frac{1}{C}\frac{1}{T}\sum_{i=1}^{T}\sum_{j=1}^{C}\frac{10}{log10}\sqrt{2(x_{i,j}^{cvt} - x_{i,j}^{grd})^2}, \tag{2.42}$$

where $x_{i,j}^{cvt}$ and $x_{i,j}^{grd}$ denote the element of the $\mathbf{X}^{cvt}$ and $\mathbf{X}^{grd}$, respectively.

The prosodic information of a converted sample can be compared with a ground-truth sample. In order to evaluate prosodic information, F0-RMSE and F0-CORR can be used (Sisman et al., 2020). Given a converted sample and a ground-truth sample, F0s can be extracted. F0-RMSE compares F0s through the L2 distance, F0-CORR compares F0s through Pearson Correlation Coefficient (PCC) (Benesty et al., 2009).

Given the F0s of the converted speech $\mathbf{y}^{cvt} \in \mathbb{R}^T$ and the F0s of the ground-truth speech $\mathbf{y}^{grd} \in \mathbb{R}^T$. where T denotes the time length. So F0-RMSE score can be computed as follows.

$$F0RMSE(\mathbf{y}^{cvt}, \mathbf{y}^{grd}) = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(y_i^{grd} - y_i^{cvt})^2}, \tag{2.43}$$

where y_i^{cvt} and y_i^{grd} denote the element of $\mathbf{y}^{cvt}$ and $\mathbf{y}^{grd}$, respectively.

The F0-CORR can be computed as follows.

$$F0CORR(\mathbf{y}^{cvt}, \mathbf{y}^{grd}) = \frac{cov(\mathbf{y}^{grd}, \mathbf{y}^{cvt})}{\sigma(\mathbf{y}^{grd})\sigma(\mathbf{y}^{cvt})}, \tag{2.44}$$

where $cov()$ and $\sigma()$ denote covariance and standard deviation.

2.6.4 Evaluation Challenges in VC

This section introduces two challenge series that are relevant to VC. The first one is the VCC series, including the VCC 2016 (Toda et al., 2016), the VCC 2018 (Lorenzo-Trueba et al., 2018) and the VCC 2020 (Yi et al., 2020). The second one is the ZS challenge series, including the ZS 2019 (Dunbar et al., 2019) and the ZS 2020 (Dunbar et al., 2020).

VCC series: VCC 2016, VCC 2018 and VCC 2020

The details of the datasets of the three VCC challenges have been introduced before. Generally, the VCC series aims to set up common benchmarks that evaluate VC techniques under the same evaluations. Hence, the differences of VC techniques can be shown from the results of challenges. The challenge evaluations mainly used subjective evaluation and mainly focused on two aspects: naturalness and speaker similarity.

The tasks of the VCC challenges varied slightly. The VCC 2016 mainly focused on a parallel data VC task. The MOS was used as an evaluation metric for the naturalness evaluation and the 'same/different' test with two options was used for the speaker similarity evaluation. From the challenge summary paper (Toda et al., 2016), the GMM based model (Stylianou et al., 1998, Toda et al., 2007) was the most popular model. The best performing GMM based model could reach a MOS of 3.0 for naturalness and a 70 % 'same' percentage for speaker similarity. However, it was obvious that the voice quality of converted samples were far behind the quality level of real human voices.

The VCC 2018 became a challenge with two sub-tasks: a parallel data task and a non-parallel data task. Hence, the focus of the challenge was extended to non-parallel VC. The challenge used the MOS for the naturalness evaluation. The 'same/different' test with four options was used for the speaker similarity evaluation.

The performance of parallel data models and non-parallel data models can be compared. It was observed that the best performing non-parallel data model (Liu et al., 2018) significantly outperformed the best performing GMM based model (Kobayashi and Toda, 2018). (Liu et al., 2018) obtained a 4.1 MOS and the best performing GMM based model obtained a MOS of 3.5. (Liu et al., 2018) obtained 93% 'same' percentage. The best performing GMM based model obtained 71% 'same' percentage. It was also notable that, instead of the GMM based models, neural network based models have become popular, such as LSTM (Hochreiter and Schmidhuber, 1997) based models.

The VCC 2020 challenge has two sub-tasks: an intra-lingual VC task and a cross-lingual VC task. The challenge reused the same subjective metrics from the VCC 2016 for evaluations. In the VCC 2020, the most popular model became the encoder-decoder based models. From the evaluation results, the best performing model was Taco-AR (Liu et al., 2020b). Taco-AR is a PPGs based VC model, which has been introduced before. In the intra-lingual VC task, Taco-AR obtained a 4.3 MOS and 95% 'same' percentage. It was discussed in (Yi et al., 2020) that Taco-AR has reached a level of performance very close to human voices.

For the cross-lingual VC task, Taco-AR obtained a 4.3 MOS and 70% 'same' percentage. It was observed that compared to the level of performance on the intra-lingual VC task, most models experienced a performance degradation on the cross-lingual VC task.

The developments of the VCC series show that there has been a tendency to extend VC techniques to a wider range of applications. From the VCC 2016 to the VCC 2018, there have been attempts to release the limitations of dependencies on parallel data. In the VCC 2020, the limitation of the language of speakers was released.

ZS Challenge series: ZS 2019 and ZS 2020

The ZeroSpeech Challenge series have four episodes: ZS 2015 (Versteegh et al., 2015), ZS 2017 (Dunbar et al., 2017), ZS 2019 (Dunbar et al., 2019) and ZS 2020 (Dunbar et al., 2020). The original target of the ZS challenges was not VC. The ZS 2015 and ZS 2017 mainly focused on AUD, which aims to learn phonetic units or representations that have high phonetic discriminability, regardless of the speaker-dependent information of speech.

Then the ZS 2019 extended the challenge with another sub-task, VC. The ZS 2019 became a multi-task challenge, including an AUD task and a VC task. The learned phonetic units or representations are referred to the linguistic representations in the encoder-decoder VC framework. In the VC task, the objective is to generate a speech signal from representations learned from the AUD task. The ZS 2019 used subjective evaluations for the VC task. The MOS tests were used for the naturalness evaluations and the speaker similarity evaluations. The human transcription was used for the intelligibility evaluations.

The following briefly reviews the ZS 2019, including the models used in the challenge and their performance. As introduced before, the VQVAE model has been a popular auto-encoder based VC model. The VQVAE model applied the VQ block as a constraint, so that the latent representations can preserve phonetic information and be invariant to speaker information. The latent representations of the VQVAE model can also be used for the phonetic representation task.

In the ZS 2019, many models (Eloff et al., 2019, Tjandra et al., 2019, Liu et al., 2019, Cho et al., 2019) were developed based on the VQVAE model. From the challenge results, (Tjandra et al., 2019) obtained highly competitive performance on the VC task and the VQ-WAE model (Cho et al., 2019) performed well on the AUD task.

(Tjandra et al., 2019) built up a VQVAE model and they integrated it with a discriminator, which makes the model similar to the GAN model. Because the VQVAE model can be regarded as the generator and the VQVAE model is adversarially trained with the discriminator. The objective was to improve the voice quality of converted speech samples. The VQ-WAE model (Chorowski et al., 2019) was proposed for a phonetic representation learning task. The details of the VQ-WAE model will be introduced in Chapter 4. (Cho et al., 2019) applied the VQ-WAE model in the ZS 2019. The VQ-WAE model performed well on the AUD task, but it performed poorly on the VC task. It obtained a 1.23 naturalness MOS result and a 1.28 speaker similarity MOS result. Also it performed poorly on the intelligibility, it only obtained a 85% CER result in the human transcription test.

There has been a phenomenon in the ZS 2019 challenge that the models that performed well on the AUD would perform badly on the VC. As summarised in (Dunbar et al., 2019) that there existed a trade-off between the voice quality of VC and the phonetic discriminability of latent representations. The VQVAE based models can be seen as an information bottleneck, where the encoder and the VQ block compress information and the decoder reconstructs information. As introduced before, this data compression is achieved by applying constraints on the latent representations. So when this constraint is too strong, it might cause an information loss issue.

2.6.5 Current Problems of VC Techniques

The developments of VC techniques follow a general tendency that is to release the limitations of VC systems, for example, the dependencies on the parallel data, the number of source-to-target mappings. By releasing these limitations, VC techniques can be applied in a wider range of practical applications. In terms of the parallel data dependencies, in the recent VCC 2018 challenge, the best performing non-parallel VC model (Liu et al., 2018) outperformed the best performing GMM based VC model (Kobayashi and Toda, 2018).

This chapter summarises the following three challenges of VC techniques.

Text-Free VC Systems

As introduced before, text-dependent VC systems have gained successes in the recent VCC 2020 challenge. It can be observed from the results of the VCC 2020 challenge that the performance of text-free VC systems are far behind text-dependent VC systems. This usually is because of the information loss issue caused by the disentanglement learning methods used in the models.

Source-to-target Mappings

Most parallel data VC models (Abe et al., 1990, Shikano et al., 1991, Stylianou, 2001) are regarded as one-to-one models. Some non-parallel VC models (Huang et al., 2020a,b, Hayashi et al., 2021, Kaneko et al., 2019a, Kaneko and Kameoka, 2017, Kaneko et al., 2020) still only support one-to-one mapping.

In the recent VCC 2020 challenge, many-to-many VC models (Liu et al., 2020b, Zhang et al., 2020) have been widely studied. From the results of the VCC 2020 (Yi et al., 2020), the best-performing many-to-many model (Liu et al., 2020b) outperformed the best-performing one-to-one model (Huang et al., 2020c).

Furthermore, many any-to-any VC systems (Liu et al., 2021, Huang et al., 2022b, Qian et al., 2019, Chou and Lee, 2019, Huang et al., 2022b) have been developed. Generally speaking, it can be observed that, when extending testing speakers to unseen speakers, VC techniques usually experience a performance degradation, especially the speaker similarity performance (Huang et al., 2022b). In terms of one-shot VC systems (Chou and Lee, 2019, Wu et al., 2020, Tang et al., 2022), the performance of VC techniques is far behind the level of real human voices.

Cross-Lingual VC

Many works (Zhao et al., 2021, Huang et al., 2022b, Yi et al., 2020, Zhou et al., 2019, Mohammadi and Kim, 2018) have been developed for cross-lingual VC. In the VCC 2020 challenge, cross-lingual VC is a sub-task of the challenge. Generally speaking, compared to the performance of VC techniques on intra-lingual VC, the performance on cross-lingual VC is poorer.

2.7 Summary

This chapter briefly reviews the VC area. It begins with an introduction of a workflow of VC techniques. Based on this workflow, this chapter introduces several speech analysis and reconstruction methods commonly used for VC. In terms of VC models, this chapter first reviews parallel data VC models. For non-parallel data models, encoder-decoder based models are summarised and demonstrated, followed by introductions of several types of GAN based models. Finally, this chapter briefly reviews related datasets, challenges, evaluation metrics and current problems of VC.

Chapter 3

Unsupervised Phonetic Representation Learning

3.1 Introduction

The choice of data representations for machine learning models is important (Bengio et al., 2013). This is because data representations can entangle different explanatory factors of variation behind the data (Bengio et al., 2013). The objective of representation learning is to learn useful information from data, so that learned representations can be used and benefit down-stream tasks, such as image classification (Hinton et al., 2006, Bengio et al., 2006, Chen et al., 2020), natural language processing (Collobert et al., 2011, Kenton and Toutanova, 2019), ASR (Dahl et al., 2010, Deng et al., 2010, Hsu et al., 2021, Baevski et al., 2020). Data representations typically are applied in down-stream tasks as the inputs to models. Compared to using raw data, learned representations usually have several advantages.

- They can improve the performance of models (Dahl et al., 2010, Deng et al., 2010).
- They can be more robust to noises (Kenton and Toutanova, 2019, Chen et al., 2020).
- They can be used for multiple tasks (Kenton and Toutanova, 2019, Baevski et al., 2020).
- They can be generalised well to other domains (Baevski et al., 2020, Kenton and Toutanova, 2019).
- They can reduce the amount of labeled data needed (Baevski et al., 2020).

Speech signals are complex time series data. The performance of speech processing tasks, such as ASR (Deng and Li, 2013), are highly dependent on the quality of data. Speech

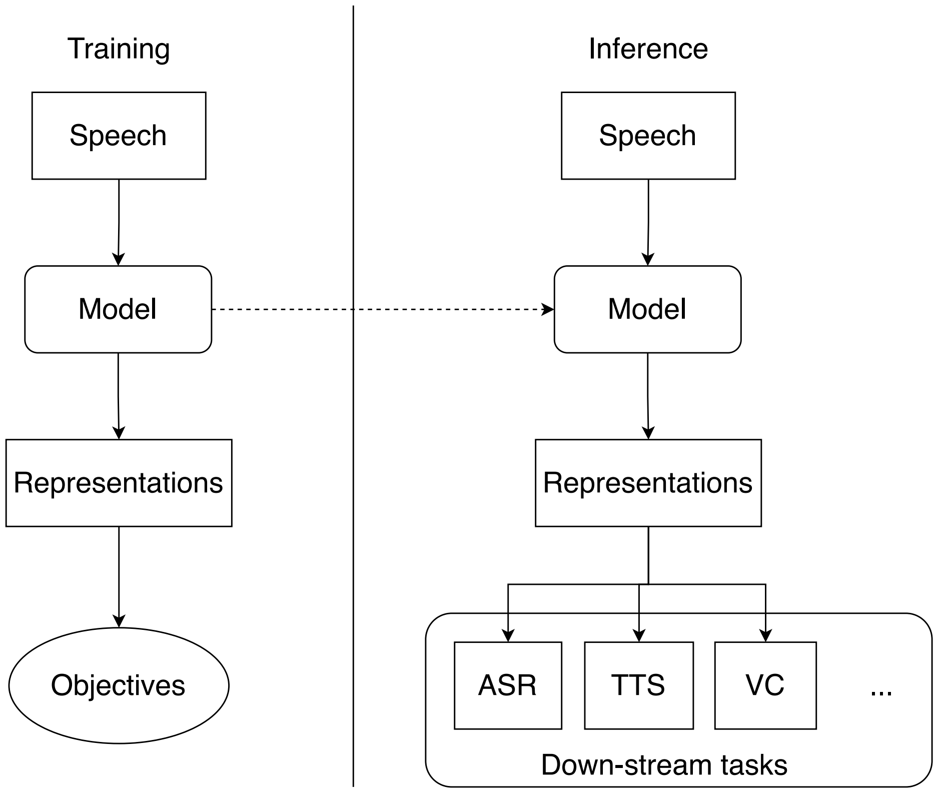

Fig. 3.1 A typical workflow of applying speech representations in down-stream speech tasks.

representation learning (Schneider et al., 2019, Baevski et al., 2020, Hsu et al., 2021, Liu et al., 2020a, Chung and Glass, 2020, Chung et al., 2020, Chorowski et al., 2019, Tjandra et al., 2019, Eloff et al., 2019) aims to learn useful information from speech signals.

A workflow of speech representation learning is illustrated in Figure 3.1. The workflow is composed of a training phase and an inference phase. In the training phase, the model takes in speech data and produces representations. The model can be trained with supervised objectives (Nguyen et al., 2014) or unsupervised objectives (Schneider et al., 2019, Hsu et al., 2021). In the inference phase, a representation learning model takes in speech data and outputs representations. The representations are typically frame-wise vectors, similar to commonly used speech features (e.g. Mel-Frequency Cepstral Coefficients (MFCCs)).

As introduced in Chapter 2, information in a speech signal can be roughly categorised into three classes: linguistic information, para-linguistic information and speaker information. Learning linguistic information from speech signals can benefit many speech tasks. For example, as introduced in Chapter 2.4 that encoder-decoder based VC models typically utilise a linguistic encoder for extracting the linguistic information of speech data.

An obvious solution for learning linguistic representations is to employ ASR models (Dong et al., 2018, Watanabe et al., 2017, Graves et al., 2013). ASR models are trained with text transcriptions, so that the intermediate representations of ASR models can be regarded as linguistic representations. For example, as introduced in Chapter 2.4, PPGs (Liu et al., 2020b,c, Sun et al., 2016a) or bottleneck features (Liu et al., 2021) have been applied in VC.

Another solution for learning linguistic information is through unsupervised models (Schneider et al., 2019, Baevski et al., 2020, Hsu et al., 2021, Chorowski et al., 2019). The objective of these models is to learn linguistic information from speech data without text transcriptions. Because no transcriptions are available, no word level information can be learned. These models can be regarded as unsupervised phonetic representation learning models. The corresponding task is called the unsupervised phonetic representation learning task.

In order to learn unsupervised phonetic representations, a solution is to remove the speaker-dependent information of speech, so that the preserved information would be only phonetic information. In order to remove speaker information, disentanglement learning methods (Chorowski et al., 2019, Baevski et al., 2019, Chung et al., 2020, Tjandra et al., 2019, Eloff et al., 2019) can be used.

Disentanglement learning is a popular research direction in machine learning. It has been studied widely in various applications, such as image style transfer (Choi et al., 2018, 2020), text generation (Cheng et al., 2020, Balasubramanian et al., 2021) and VC (Chou and Lee, 2019, Wu et al., 2020). However, the concept of disentanglement learning is complex currently. This chapter summarises two types of disentanglement learning methods by their objectives. Figure 3.2 illustrates the difference between these two types of methods. This chapter names the first type as factorisation learning methods and the second type as invariant information learning methods.

Specifically, the objective of the factorisation learning methods is to learn representations of independent explanatory factors from data. The number of factors can be one or more. The learning process usually is unsupervised, so that before training, what factors can be learned are not clear. The VAE model (Kingma and Welling, 2014) and several of its extension models (Burgess et al., 2018, Kim and Mnih, 2018) are examples of the factorisation learning.

In contrast, the objective of the invariant information learning methods is to learn disentanglement between two types of information. One type of information is dependent on a factor and the other type of information is invariant to the factor. For example, for learning phonetic representations, the task can be regarded as learning speaker-invariant representations. For simplicity, this chapter calls the speaker-invariant information learning methods as the disentanglement learning methods.

Many models (Chorowski et al., 2019, Baevski et al., 2020, Hsu et al., 2021, Baevski et al., 2019, Schneider et al., 2019, Tjandra et al., 2019, Eloff et al., 2019) have been developed

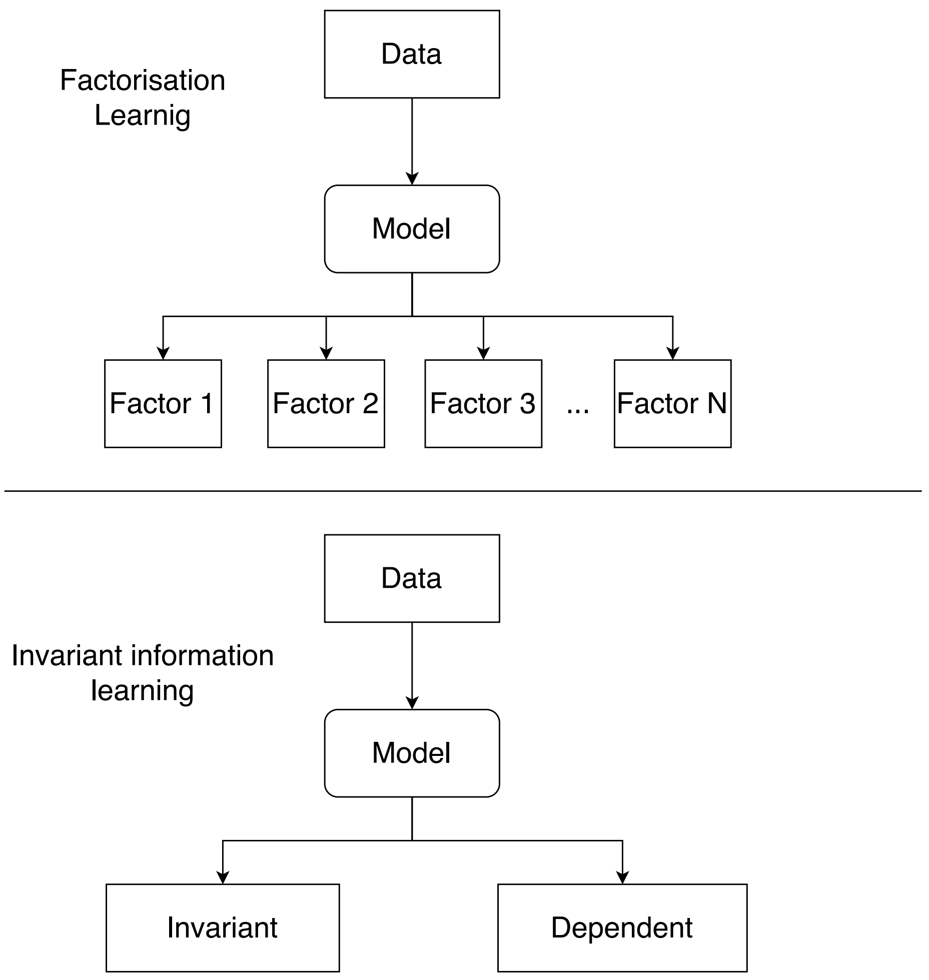

Fig. 3.2 The difference between the two types of disentanglement learning methods, factorisation learning methods and invariant information learning.

for the unsupervised phonetic representation learning task. There have been two types of commonly used models for this task, auto-encoder models (Chorowski et al., 2019, Vachhani et al., 2017, Tan and Sim, 2016, Karita et al., 2019) and SSL models (Schneider et al., 2019, Baevski et al., 2020, Hsu et al., 2021).

Disentanglement learning methods are applied to remove speaker information from data. As summarised in (Sisman et al., 2020), there have been two commonly used methods for this. One type is normalisation methods (Eide and Gish, 1996, Huang and Belongie, 2017, Gales, 1998), which remove speaker information through normalising data across time. The other type of methods is VQ (Van Den Oord et al., 2017, Chorowski et al., 2019, Baevski et al., 2019), which removes speaker information through information compression.

These two methods are simple but effective for learning disentanglement between the speaker information and the speaker-invariant information. They can be applied in both the auto-encoder models (Chorowski et al., 2019, Tjandra et al., 2019, Eloff et al., 2019) and the SSL models (Hsu et al., 2021, Schneider et al., 2019, Baevski et al., 2020, 2019).

The remainder of this chapter is organised as follows:

- Section 3.2 reviews speech normalisation and speaker adaptation methods in speech processing areas.
- Section 3.3 reviews VQ and its extension method, SVQ.
- Section 3.4 reviews auto-encoder based models for learning phonetic representations.
- Section 3.5 reviews SSL models for learning phonetic representations.
- Section 3.6 introduces datasets, evaluation metrics, challenges.

3.2 Normalisation and Adaptation

Normalisation methods typically aim to reduce the difference of data samples through mapping data samples into a common space. Speaker normalisation methods are applied to reduce the speaker difference between speech samples. These methods assume speaker information changes slowly over time. An obvious solution is to map speech samples into a common space where the utterance level means and variances of samples are the same.

Speaker normalisation methods have been found effective in many speech tasks. For example, they have been applied for reducing the difference of training data and test data in ASR (Liu et al., 1993, Viikki and Laurila, 1998, Eide and Gish, 1996). In the context of VC, these methods (Huang and Belongie, 2017) have been found effective for removing speaker information.

The objective of speaker adaptation methods (Huang et al., 2005, Gales, 1998, Tibrewala and Hermansky, 1997, Varadarajan et al., 2008, Gales et al., 1996) in ASR is to enhance the performance of ASR models on the data of a target speaker. The objective of adaptation methods (Dumoulin et al., 2017, Huang and Belongie, 2017) in VC is to adapt speech features of a source speaker to a target speaker. Some normalisation methods, such as IN, can be extended as an adaptation method.

3.2.1 Speaker Normalisation Methods

Speaker normalisation methods can be distinguished from what data they act on. One type of speaker normalisation methods (Liu et al., 1993, Viikki and Laurila, 1998, Eide and Gish, 1996) act on speech features, which are usually the inputs to ASR models. The other type of speaker normalisation methods, such as IN (Huang and Belongie, 2017), are commonly applied as an intermediate layer in neural networks. They act on intermediate features in neural networks.

Speaker Normalisation for Speech Features

Speech features based normalisation methods can further be divided into three types, which are listed as follows.

- mean normalisation (Liu et al., 1993);
- mean variance normalisation (Viikki and Laurila, 1998);
- frequency warping (Eide and Gish, 1996).

Given speech features $\mathbf{X} \in \mathbb{R}^{C\times T}$, where C and T denote the feature dimensionality and duration, respectively. $x_{i,j}$ denotes the element of $\mathbf{X}$. Mean normalisation computes the utterance level mean vector $\boldsymbol{\mu} \in \mathbb{R}^{C}$ of the speech features. Let the element of the mean vector be μ_i, which can be obtained as follows.

$$\mu_i = \frac{1}{T}\sum_{j=1}^{T} x_{i,j}. \tag{3.1}$$

Then the speech features $\mathbf{X}$ are normalised by the mean vector. Let $\hat{x}_{i,j}$ be the element of the normalised features $\hat{\mathbf{X}}$. $\hat{x}_{i,j}$ can be obtained as follows.

$$\hat{x}_{i,j} = x_{i,j} - \mu_i. \tag{3.2}$$

Mean variance normalisation computes the utterance level mean vector and the utterance level standard deviation vector of the speech features. Let μ_i be the element of the mean vector $\boldsymbol{\mu}$ and let σ_i be the element of the standard deviation vector $\boldsymbol{\sigma}$. μ_i and σ_i can be obtained as follows, respectively.

$$\mu_i = \frac{1}{T}\sum_{j=1}^{T} x_{i,j}, \tag{3.3}$$

$$\sigma_i = \sqrt{\frac{1}{T}\sum_{j=1}^{T} (x_{i,j} - \mu_i)^2}. \tag{3.4}$$

Then the element $\hat{x}_{i,j}$ of the normalised features $\hat{\mathbf{X}}$ can be computed as:

$$x_{i,j} = \frac{x_{i,j} - \mu_i}{\sqrt{(\sigma_i)^2 + \varepsilon}}, \tag{3.5}$$

where ε is a constant that avoids the denominator being zero.

Apart from mean normalisation and mean variance normalisation, another type of normalisation methods is frequency warping. VTLN (Eide and Gish, 1996, Kamm et al., 1995) is a commonly used frequency warping method. VTLN methods normalise speech data by taking account of vocal tract length. Because the speaker information of a speech signal is relevant to the length of the speaker's vocal tract (Eide and Gish, 1996). These methods aim to "compensate for the effect of speaker-dependent vocal tract lengths by warping the frequency axis of the phase and magnitude spectrum" (Sundermann and Ney, 2003).

Given a speech sample, VTLN methods typically generate one or more warping factors and normalise speech features through a warping function. The warping factors are applied in the warping function. As summarised in (Sundermann et al., 2004), VLTN methods can be distinguished from how many warping factors are used in the warping function, for example, one warping factor (Wegmann et al., 1996, Uebel and Woodland, 1999) or multiple warping factors (Sundermann and Ney, 2003). Many warping functions have been developed for VLTN. They can be divided into two types: linear functions (Eide and Gish, 1996, Acero and Stern, 1991, Pitz and Ney, 2005) and non-linear functions (Sundermann et al., 2004).

Speaker Normalisation for Intermediate Features

Many normalisation methods (Ioffe and Szegedy, 2015, Huang and Belongie, 2017, Wu and He, 2018) can be used as an intermediate layer in neural networks. For example, batch normalisation (Ioffe and Szegedy, 2015) was developed for enhancing the training stability of neural network models. Another example is the IN layer, it was developed for removing the style information in images, so that the performance of image style transfer models can be improved. In the context of VC, it has been introduced in Section 2.4 that IN can be used for removing the speaker information of speech features (Chou and Lee, 2019, Chen et al., 2021b, Kaneko et al., 2019b,a, 2020, Kaneko and Kameoka, 2017).

The following introduces the details of IN. IN conducts a mean variance normalisation across time on intermediate features in neural networks. Let the intermediate features be $\mathbf{X} \in \mathbb{R}^{C \times T}$, the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ can be computed as in Equation 3.3 and 3.4, respectively. Then the normalised features $\hat{\mathbf{X}}$ can also be computed as in Equation 3.5. IN is usually followed by a linear transform when applied in neural networks (Kaneko et al., 2019b,a, 2020).

3.2.2 Speaker Adaptation Methods

Similar to speaker normalisation methods, speaker adaptation methods can be distinguished from what they act on. The following lists a categorisation of adaptation methods.

- Speech features based adaptation (Varadarajan et al., 2008, Gales, 1998, Huang et al., 2005, Li et al., 2002, Huang and Sim, 2015) ;
- Intermediate features based adaptation (Dumoulin et al., 2017, Huang and Belongie, 2017) ;
- Model parameters based adaptation (Gales, 1998, Nguyen et al., 1999, Ferras et al., 2007, Gales et al., 1996).

Speech Features based Adaptation Methods

FMLLR (Varadarajan et al., 2008, Gales, 1998, Huang et al., 2005, Li et al., 2002, Huang and Sim, 2015) is an example of adaptation methods that act on speech features. In ASR speaker adaptation, a speaker independent ASR model usually has already been trained. Given a target speaker and its adaptation data, the speech features from the adaptation data are adapted through a linear transform, to enhance recognition performance on the adaptation data. The speech features are usually MFCCs or filter-bank features and are the inputs to models. For a sequence of speech features $\mathbf{X} = \{\mathbf{x}_t \in \mathbb{R}^C | t = 0, ..., T\}$, fMLLR adapts them through a linear transform. The linear transform can be written as follows.

$$\mathbf{x}'_t = \mathbf{W}\mathbf{x}_t + \mathbf{b}, \tag{3.6}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the transform weight and bias, respectively. The transform weight and bias can be updated iteratively to maximise the likelihood of adaptation data by using the EM algorithm (Dempster et al., 1977).

Intermediate Features based Adaptation Methods

Intermediate features based adaptation methods (Huang and Belongie, 2017, Dumoulin et al., 2017) can usually be applied as an intermediate layer in neural networks. These methods are usually the extension of IN, which has been introduced before. CIN (Dumoulin et al., 2017) and AdaIN (Huang and Belongie, 2017) are two commonly used speaker adaptation layers applied in VC models (Chou and Lee, 2019, Kaneko et al., 2019b).

The following introduces the details of CIN and AdaIN. Given intermediate features $\mathbf{X} = \{\mathbf{x}_t \in \mathbb{R}^C | t = 0, ...T\}$, they can be normalised by IN as in Equation 3.3, 3.4 and 3.5. Let a normalised feature vector be $\mathbf{x}'_t$. The normalised feature vector is adapted to a given target speaker by a linear transform. The linear transform can be written as follows.

$$\hat{\mathbf{x}}_t = \boldsymbol{\gamma} \odot \mathbf{x}'_t + \boldsymbol{\beta}, \tag{3.7}$$

where $\hat{\mathbf{x}}_t$ is the tth frame of the adapted features $\hat{\mathbf{X}}$.

The difference of CIN and AdaIN is how to produce the affine parameters. CIN uses a speaker one-hot vector to produce the affine parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. Given a one-hot vector $\mathbf{s}$ of the target speaker, CIN uses two linear layers to project $\mathbf{s}$ to $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, respectively.

$$\boldsymbol{\gamma} = \mathbf{W}_1\mathbf{s} + \mathbf{b}_1, \tag{3.8}$$

$$\boldsymbol{\beta} = \mathbf{W}_2\mathbf{s} + \mathbf{b}_2, \tag{3.9}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are parameters of linear layers.

Different from CIN that uses a speaker one-hot to produce affine parameters, AdaIN uses a speaker embedding $\mathbf{s}_{emb}$ to produce affine parameters. The speaker embedding is extracted from a speaker encoder $SEnc()$. The speaker encoder can be trained independently or trained together with the other parameters of the model. Given the speech features $\mathbf{X}^{tgt}$ of a target speaker, the speaker embedding could be obtained by

$$\mathbf{s}_{emb} = SEnc(\mathbf{X}^{tgt}). \tag{3.10}$$

Then the two affine parameters can be obtained by two linear layers.

$$\boldsymbol{\gamma} = \mathbf{W}_1\mathbf{s}_{emb} + \mathbf{b}_1, \tag{3.11}$$

$$\boldsymbol{\sigma} = \mathbf{W}_2\mathbf{s}_{emb} + \mathbf{b}_2, \tag{3.12}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are parameters of linear layers.

Model Parameters based Adaptation Methods

The speech features based adaptation methods and the model parameters based adaptation methods have been discussed in (Gales, 1998). It was concluded that "ML training of strict linear feature–space transformations may be shown to be inappropriate for ASR" (Gales, 1998). Instead of adapting speech features, MLLR (Leggetter and Woodland, 1995, Gales, 1998) adapts the parameters of a trained ASR model.

The adaptation process of MLLR is similar to FMLLR. It adapts the parameters of the model through linear transforms. The parameters of the linear transforms are estimated through maximising likelihood of target speaker data. The Expectation Maximisation (EM) algorithm (Dempster et al., 1977) can be used to update the parameters of the linear transforms.

3.3 Vector Quantisation and Sliced Vector Quantisation

An ideal method of modeling speaker-invariant phonetic information is to extract intermediate representations from an ASR model (Watanabe et al., 2017). When the ASR model is not available, one solution is to use a data compression method to remove speaker information. In history, there have been many data compression methods developed (Naqvi et al., 2011, Ahmed, 1991). As a data compression method, VQ has been widely used in speech tasks such as speech coding (Makhoul et al., 1985), ASR (Sonkamble and Doye, 2012, Furui, 1991) and VC (Cho et al., 2019, Wu et al., 2020, Tang et al., 2022, Wang et al., 2019). As has discussed in many papers (Pan et al., 1996, Van Den Oord et al., 2017) that VQ is highly relevant to clustering methods such as KMeans (MacQueen et al., 1967) and Gaussian Mixture Model (GMM) (Duda et al., 1973). The common attributes of VQ and clustering methods is that they assign an input data point to a finite set of codebooks or cluster centroids. This attribute helps to model dynamic information that changes frequently across time in speech, for example, phonetic information. Clustering methods have been proved useful for modeling speaker-invariant phonetic information in (Lee, 1988). The main purpose of applying VQ in these applications is model speaker-invariant phonetic information. Please note that, since VQ is an unsupervised method, it can not perfectly remove speaker information and preserve phonetic information. In practice, it might cause phonetic information loss when removing speaker information. This section introduces the detailed information of VQ and its extension SVQ.

3.3.1 Vector Quantisation

Let a feature vector be $\mathbf{x} \in \mathbb{R}^D$ and a D dimensional space be $\mathbb{R}^D$, which is divided into K regions. Let each region be $R_i, i = 1, ..., K$ and the centroid of each region be $\mathbf{e}_i \in \mathbb{R}^D$, where a centroid is used to represent the region. A clustering method aims to find the best match region R_* for the feature vector $\mathbf{x}$. A distance function $d()$ is required to calculate the distances between the centroid of each region and the feature vector. After computing the distances $\{l_1,...,l_K\}$, a search function $s()$ will be used to generate a representation of the feature vector. The representation could be the index of centroid or the centroid vector itself.

VQ defines a finite set of units named a codebook. The units are named as codebook vectors. Following the above framework, the codebook can be written as $\mathbf{E} = \{\mathbf{e}_i \in \mathbf{R}^D | i = 1,...,K\}$. VQ uses the Euclidean distance function as the distance function and the argmin function as the search function.

Following on the definition, the distance function $d()$ can be written as follows.

$$d(\mathbf{e}_i, \mathbf{x}) = ||\mathbf{e}_i - \mathbf{x}||_2. \tag{3.13}$$

Given the feature $\mathbf{x}$, VQ computes the distances $\mathbf{l}$ of the feature vector $\mathbf{x}$ to each codebook vector $\mathbf{e}_i$. The element l_k of the distance vector $\mathbf{l} = \{l_1,...,l_k,...,l_K\}$ can be obtained as follows.

$$l_k = d(\mathbf{x}, \mathbf{e}_k). \tag{3.14}$$

The argmin search function can be written as follows.

$$s(\mathbf{l}) = \underset{i\in[1,K]}{\operatorname{argmin}}\, l_i. \tag{3.15}$$

The output of VQ, $R_* = s(\mathbf{l})$, is the output of the search function, which is the index of the nearest neighbour codebook vector of the feature vector.

3.3.2 Sliced Vector Quantisation

As discussed in (Al-Mualla et al., 2002) that "compression in VQ is achieved by using a codebook with relatively few codebook vectors compared to the number of possible input vectors", there exists a trade-off between the information loss and the computation efficiency of VQ. Using a larger K, the information loss would be smaller. However, the computation efficiency would be increased. Because the computation complexity of VQ grows linearly as K increases.

In order to alleviate the information loss of VQ while maintaining a high computation efficiency, SVQ (Kaiser et al., 2018) was proposed. SVQ was also named as Product Quantisation (PQ) in (Jegou et al., 2010).

In order to increase the number of codebook vectors without hugely increasing the computation costs of VQ, one solution is to increase the number of codebooks. In VQ, there is only one codebook. SVQ can use multiple codebooks.

SVQ defines M codebooks $\{\mathbf{E}^j \in \mathbb{R}^{K\times D} | j = 0,...,M\}$. SVQ slices the feature vector $\mathbf{x}$ into M slices $\mathbf{x} = [\mathbf{x}^j \in \mathbf{R}^{D/M} | j = 0,..,M]$. For one slice $\mathbf{x}^j$ SVQ finds the nearest neighbour codebook vector from the jth codebook $\mathbf{E}^j$.

The distance function $d()$ is used to to compute the distances $\mathbf{l}^j = \{l_1^j,...,l_K^j\}$ with the codebook vectors in the jth codebook $\{\mathbf{e}_1^j,...,\mathbf{e}_K^j$. Let l_i^j be the element of $\mathbf{l}^j$. l_i^j can be computed as follows.

$$l_i^j = d(\mathbf{x}^j, \mathbf{e}_i^j). \tag{3.16}$$

After compute the distances for all slices $\{\mathbf{l}^j | j = 0,...,M\}$. The search function is used to find the indices of the nearest neighbour centroids in each codebook $\mathbf{E}^j$. Let R_*^j be the output of the search function, which is the indices of the jth nearest neighbour codebook vector. R_*^j can be obtained as follows.

$$R_*^j = s(\mathbf{l}^j). \tag{3.17}$$

The output of SVQ is a concatenation of the outputs of all indices $[R_*^j | j = 1,...,M]$.

3.4 Auto-Encoder based Models

As discussed in Chapter 2 that the auto-encoder model can be used for learning representations. The auto-encoder model and its two extension models, the VAE model (Kingma and Welling, 2014) and the VQVAE model (Van Den Oord et al., 2017), have been widely used for learning speech representations (Karita et al., 2019, Chorowski et al., 2019, Eloff et al., 2019, Tjandra et al., 2019, Liu et al., 2019). These auto-encoder models reduce the redundant information in the latent representation through applying constraints. These constraints can be regarded as methods for learning disentangled speaker-invariant information for VC. For the disentangled phonetic representation learning task, these constraints can also be used.

In a recent paper (Chorowski et al., 2019), these auto-encoder models have been analysed for an unsupervised phonetic representation learning task. In (Chorowski et al., 2019), the objective was to compare the constraints used in these auto-encoder models, because the latent representations learned from these three auto-encoder models would have different quality. In order to assess the quality of the latent representations, (Chorowski et al., 2019) used evaluation metrics that mainly focuses on two aspects: preserving phonetic information and removing speaker information.

The three auto-encoder models were built with encoder-decoder model architecture. The three models only differed from the constraints applied on the latent representation of the model. Specifically, (Chorowski et al., 2019) built up a fundamental framework with a convolution neural network (CNN) based encoder and a WaveNet (van den Oord et al.,

2016) decoder. The WaveNet directly reconstructs waveform speech signals. It was found in (Chorowski et al., 2019) that it can learn better phonetic representations than other decoders.

The VQVAE model with a WaveNet decoder in (Chorowski et al., 2019) is named as Vector Quantised WaveNet Auto-Encoder (VQ-WAE) in this thesis. To evaluate the learned latent representations, they (Chorowski et al., 2019) set up a phone classification task and a speaker classification task. They first trained the three auto-encoder models on the LibriSpeech dataset (Panayotov et al., 2015). Then the auto-encoder models are used to extract latent representations from samples of the LibriSpeech dataset. Then the extracted latent representations together with ground-truth phoneme labels and speaker labels are used for a phone classification task and a speaker classification task, respectively. In the classification tasks, a simple one linear layer model is used. The representations are the inputs to the models. After training the classification models on the representations, the test accuracy can be used as a metric of how much phonetic or speaker information remains in the representations. The higher phone classification accuracies and lower speaker classification accuracies imply the better phonetic representations.

From their results, the auto-encoder model obtained good phone classification accuracies, however, it also obtained high speaker classification accuracies. It implies that the auto-encoder performed poorly on removing speaker information. The VAE model obtained low phone classification accuracies and low speaker classification accuracies. The VQ-WAE model obtained high phone classification accuracies and low speaker classification accuracies. It was concluded that the VQ-WAE model performed better than the other two models on learning phonetic representations.

3.5 Speech Self-Supervised Learning Models

The objective of speech SSL models (Schneider et al., 2019, Baevski et al., 2020, 2019, Chung and Glass, 2020, Chung et al., 2020, Hsu et al., 2021, Liu et al., 2020a) is to learn useful speech representations from speech data. Speech representations learned from speech SSL models typically contain various information of speech signals, including phonetic information or speaker-dependent information.

Speech SSL models can be implemented in various ways. They can be distinguished from their input types, from what neural network layers they use or from what training objectives they use. This section summarises a framework of speech SSL models, which is illustrated in Figure 3.3. Generally speaking, this framework is composed of four steps, including a pre-process step, a feature encoder step, a feature aggregator step and a training step.

Table 3.1 Results from the SUPERB benchmark (Yang et al., 2021), where "Input" denotes the input type of the model. "Phone", "Speaker" and "WER" denote the results of the phone classification, the utterance level speaker classification and the ASR down-stream tasks. "Mel-Spec" denotes Mel-spectrograms. "FBank" denotes filter-bank features. "Params" denotes the number of parameters.

Representations	Input	Params	Phone (%) $\uparrow$	Speaker (%) $\downarrow$	WER (%) $\downarrow$
MFCCs	-	-	47.88	13,18	-
FBank	-	-	48.01	31.35	15.21
Mel-Spec	-	-	41.93	22.38	-
Wav2vec	Wav	32.53M	71.31	75.74	11.00
VQ-Wav2vec	Wav	34.15M	63.38	22.76	12.80
Wav2vec 2.0	Wav	95.04M	56.39	82.53	4.79
APC	Mel-Spec	4.11M	72.76	79.08	14.74
VQ-APC	Mel-Spec	4.63M	71.92	68.56	15.21
Mockingjay	Mel-Spec	85.12M	67.51	97.22	15.48

Specifically, this framework takes in waveforms and feeds the waveforms to a pre-process step. The pre-process step typically extracts speech features, such as Mel-spectrograms. Whether or not to use a pre-process step is optional for speech SSL models, since some models directly use waveforms as inputs.

The feature encoder encodes speech features or waveforms into representations $\mathbf{Z} = \{\mathbf{z}_i | i = 1, ..., T\}$. The main idea of speech SSL models is to maximise the mutual information of a frame $\mathbf{z}_i$ with its neighbouring frames. A common way to implement this idea is to use a feature aggregator to learn the contextual information of $\mathbf{z}_i$. The feature aggregator takes in the representations $\mathbf{Z}$ and outputs the aggregated representations $\mathbf{C} = \{\mathbf{c}_i | i = 1, ..., T\}$, where $\mathbf{c}_i$ represents the contextual information of $\mathbf{z}_i$. The training of the models is to maximise the mutual information of $\mathbf{c}_i$ and $\mathbf{z}_i$.

At inference time, some speech SSL models (Schneider et al., 2019, Baevski et al., 2019) output the representations $\mathbf{Z}$. There are also some speech SSL models (Baevski et al., 2020, Hsu et al., 2021, Liu et al., 2020a, Chung and Glass, 2020, Chung et al., 2020) that output the aggregated representations $\mathbf{C}$, or the output of an intermediate layer of the feature aggregator.

To evaluate the quality of representations, the SUPERB benchmark (Yang et al., 2021) established a benchmark with a number of down-stream speech tasks, including phone classification, speaker classification, ASR and so on. Table 3.1 presents the results of various speech SSL models on these down-stream speech tasks. This table also presents the differences of the input types and the model sizes of models. These results may be helpful to understand the differences of these models.

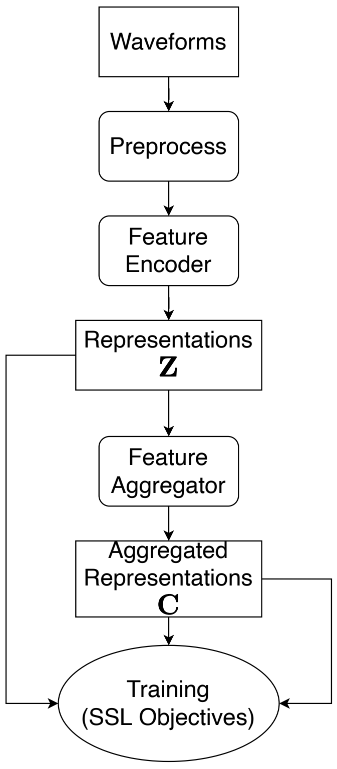

Fig. 3.3 An framework of speech SSL models.

3.5.1 Pre-processing

Speech SSL models can be categorised into two types by the input type of models. One type of models directly takes in waveforms, such as the three Wav2vec based models: Wave2vec (Schneider et al., 2019), VQ-Wav2vec (Baevski et al., 2019) and Wav2vec 2.0 (Baevski et al., 2020). In contrast, there is another type of model (Liu et al., 2020a, Chung and Glass, 2020, Chung et al., 2020) taking in speech features requires a feature extraction step.

For SSL models that take in waveforms, the pre-processing step is unnecessary. From the results in Table 3.1, it is clear that models taking in speech features generally obtained higher phone classification and speaker classification results, but worse WER results than models taking in waveforms.

3.5.2 Feature Encoder

The objective of the feature encoder is to encode a speech signal into frame-wise representations. Three types of implementations of the feature encoder have been developed: CNN based feature encoders, LSTM (Hochreiter and Schmidhuber, 1997) based feature encoders and Transformer (Vaswani et al., 2017) based feature encoders.

CNN based Feature Encoders

The three Wav2vec based models (Schneider et al., 2019, Baevski et al., 2019, 2020) and the HUBERT model (Hsu et al., 2021) used CNN based feature encoders. They typically built a feature encoder with 1D-convolution layers. 1D-convolution layers are usually configured with a large kernel size and a large stride size. This allows the feature encoder to down-sample waveforms into frame-wise representations.

As described in the Wav2vec paper (Schneider et al., 2019), "the output of the feature encoder is a low frequency feature representation which encodes about 30 ms of audio and the stride size of representations is 10ms.". Hence, the feature encoder of Wav2vec is more likely to encode the local information of a speech signal.

The Wav2vec model only used 1D-convolution layers in the feature encoder. It can encode phonetic information to representations, however, it can hardly remove speaker information. As shown in Table 3.1, Wav2vec obtained a 71.31% phone classification accuracy and a 75.74% speaker classification accuracy.

VQ-Wav2vec (Baevski et al., 2019) introduced SVQ into the feature encoder of Wav2vec. The objective of applying SVQ in the feature encoder is to compress speaker information, so that the quantised representations $\mathbf{Z}$ can be invariant to speaker information. The output of 1D-convolution layers are quantised by the SVQ. From the results in Table 3.1, when applying the SVQ block in the feature encoder, the VQ-Wav2vec obtained a 22.76% speaker classification accuracy, which was lower than the result of the Wav2vec model. This shows that the SVQ block can partially remove the speaker information of a speech signal.

LSTM based Feature Encoders

Apart from CNN based feature encoders, LSTM can be used for the feature encoder. The objective of applying LSTM in the feature encoder is different from apply CNN layers. As introduced before, the target of CNN based feature encoder is to learn the local information of a speech signal. In contrast, the target of applying LSTM is to learn the long-term dependency information of a speech signal. LSTM based feature encoder models are mainly used in models taking in speech features, such as APC (Chung and Glass, 2020) and VQ-APC (Chung et al., 2020).

APC used a stack of LSTM layers in the feature encoder. From the results of Table 3.1, APC obtained a 72.76 % phone classification accuracy and a 79.08 % speaker classification accuracy. These results may be explained by the ability of LSTM to learn long-term dependency. This ability enabled the APC model and the VQ-APC model to obtain good phone classification accuracies. This ability also caused high speaker classification accuracies of the APC model and the VQ-APC model.

VQ-APC, which is based on the APC model, applied a VQ block after the LSTM layers in the feature encoder. The target of applying VQ is to remove speaker information. From the results in Table 3.1, the phone classification accuracy and the WER of the VQ-APC model were slightly worse than the APC model. The speaker classification accuracy of the VQ-APC model was 68.56%, which was higher than VQ-Wav2vec. These results imply that the VQ-APC model performed better than the VQ-Wav2vec model on preserving phonetic information, but performed worse than the VQ-Wav2vec model on removing speaker information.

Transformer based Encoders

Following the bidirectional encoder representations from transformers (BERT) model (Kenton and Toutanova, 2019) that built a representation learning model based on a stack of Transformer layers (Vaswani et al., 2017), Mockingjay used a stack of Transformer layers in the feature encoder. It takes in Mel-spectrograms as the inputs to the model. The Transformer layers output the representations $\mathbf{Z}$. The Transformer layer has been shown having a strong ability to learn global level of dependencies in speech tasks such as ASR (Dong et al., 2018), TTS (Li et al., 2019) and VC (Huang et al., 2020b, 2021). From the results in Table 3.1, Mockingjay obtained a 97.22% speaker classification accuracy and a 67.51% phone classification accuracy. These results show that learning the global level of dependencies directly on Mel-spectrograms can encode most of the speaker-dependent information of a speech signal into the representations. Mockingjay obtained a 67.51% phone classification accuracy, which was worse than APC and VQ-APC. This means that learning the global level of dependencies might not be helpful for learning the phonetic information of a speech signal.

3.5.3 Feature Aggregator

The target of the feature aggregator is to learn the contextual information of the representations $\mathbf{Z}$. Three types of the feature aggregator have been developed: CNN based feature aggregators, LSTM based feature aggregators, Transformer based feature aggregators. In some speech SSL models, such as APC and Mockingjay, the implementations of the feature encoder and the feature aggregator are almost the same.

The three Wav2vec based models typically used a CNN based feature aggregator. For example, in the Wav2vec model, the feature aggregator is composed a stack of 13 1D-

Table 3.2 A brief summary of loss functions and training methods of speech SSL models, where "AR" denotes autoregressive, "L1" denotes the L1 distance.

SSL Model	Loss	Training Methods
Wav2vec	Contrastive	AR predictive
VQ-Wav2vec	Contrastive	AR predictive
Wav2vec 2.0	Contrastive	Masked predictive
APC	L1	AR reconstruction
VQ-APC	L1	AR reconstruction
Mockingjay	L1	Masked reconstruction

convolution layers, with the kernel sizes increasing from 1 to 13 and the stride sizes fixed to 1. By setting these configurations, the feature aggregator reached a receptive field size of 108 frames. It means that the feature aggregator can learn the contextual information of a frame $\mathbf{z}_i$ from its neighbouring 107 frames. From the results of Table 3.1, the WER results of Wav2vec and VQ-Wav2vec were 11.0% and 12.8%, respectively.

To enhance the ASR performance, Wav2vec 2.0 used a BERT model as the feature aggregator. Wav2vec 2.0 reused the feature encoder of VQ-Wav2vec. The representations $\mathbf{Z}$ of VQ-Wav2vec are discrete representations, which have been quantised by the SVQ block. A BERT model is used to model the global level of dependencies of these discrete representations. From the results in Table 3.1, Wav2vec 2.0 obtained an obvious improvement on ASR results. It got a 4.79 % WER result, which outperformed Wav2vec and VQ-Wav2vec. Wav2vec 2.0 obtained a 56.39% phone classification accuracy and a 82.53% speaker classification accuracy. Compared to VQ-Wav2vec, the Wav2vec 2.0 model performed worse on preserving phonetic information and also worse on removing speaker information.

APC and VQ-APC used LSTM based feature aggregators. From the ASR results in Table 3.1, they obtained a 14.74 % and a 15.21 % WER results, respectively. Both of these results were very close to the WER result of filter-bank features. Mockingjay used Transformer layers in the feature aggregator. Mockingjay could only obtain a 15.48% WER result.

3.5.4 Loss Functions and Training Methods

As introduced before, the training of SSL models usually follows a general idea to maximise the mutual information of a frame of the representations with its neighbouring frames. There have been many objective functions and training methods developed to implement this idea. Table 3.2 presents a brief summary of the loss functions and the training methods of speech SSL models.

Contrastive Loss and Autoregressive Predictive Training

Generally, the autoregressive predictive training means the feature aggregator is autoregressive. This can be achieved by using a stack of LSTM layers or causal convolution layers (van den Oord et al., 2016). In Wav2vec and VQ-Wav2vec, the feature aggregator is a stack of causal convolution layers. Both used the combination of the contrastive loss and the autoregressive predictive training.

Let the representations extracted by the feature encoders be $\mathbf{Z} = \{\mathbf{z}_t | t = 1, ..., T\}$, the aggregated representations be $\mathbf{C} = \{\mathbf{c}_t | t = 1, ..., T\}$. Since the feature aggregator is autoregressive, so that one aggregated feature step $\mathbf{c}_t$ only corresponds to $\{\mathbf{z}_1, ..., \mathbf{z}_{t-1}\}$. The autoregressive predictive training aims to maximise the mutual information $I(\mathbf{z}_{t+l}, \mathbf{c}_t)$ between a frame of the aggregated representations $\mathbf{c}_t$ with its L future steps of the representations $\{\mathbf{z}_{t'} | t' = t, ..., t+l, ..., t+L\}$, respectively.

Since the feature aggregators aim to learn the contextual information of the representations $\mathbf{Z}$, the aggregated feature step $\mathbf{c}_t$ can be regarded as a representative of the previous feature frames $\{\mathbf{z}_1, ..., \mathbf{z}_{t-1}\}$. Hence, maximising the mutual information $I(\mathbf{z}_{t+l}, \mathbf{c}_t)$ can be regarded as maximising the mutual information $I(\mathbf{z}_{t+l}, \{\mathbf{z}_1, ..., \mathbf{z}_{t-1}\})$ of a feature frame $\mathbf{z}_{t+l}$ with its previous frames $\{\mathbf{z}_1, ..., \mathbf{z}_{t-1}\}$. This follows the idea of SSL models that maximising the mutual information between each part of data with its neighbouring data parts.

Hence, the autoregressive predictive training method can be represented as the following loss function.

$$\mathcal{L}_{arp} = \mathbb{E}_{p(\mathbf{Z})}\{-\sum_{t=1}^{T-L}\sum_{l=1}^{L} logI(\mathbf{z}_{t+l}, \mathbf{c}_t)\}. \tag{3.18}$$

Since both $\mathbf{Z}$ and $\mathbf{C}$ are latent representations, the mutual information $I(\mathbf{z}_{t+l}, \mathbf{c}_t)$ is intractable, an approximating method is needed for estimating the mutual information. Wav2vec based models use a contrastive method, InfoNCE (Gutmann and Hyvärinen, 2010), to estimate the mutual information. Generally, InfoNCE aims to use a density function to calculate a lower bound of the mutual information. Through maximising the lower bound, the mutual information can be maximised.

The following introduces how InfoNCE defines a density function and how it defines a lower bound of the mutual information. The mutual information $I(\mathbf{z}_{t+l}, \mathbf{c}_t)$ can be written as:

$$I(\mathbf{z}_{t+l}, \mathbf{c}_t) = log\frac{p(\mathbf{z}_{t+l}|\mathbf{c}_t)}{p(\mathbf{z}_{t+l})}. \tag{3.19}$$

Instead of directly computing $I(\mathbf{z}_{t+l}, \mathbf{c}_t)$, InfoNCE defines a density function $f(\mathbf{z}_{t+l}, \mathbf{c}_t) \propto \frac{p(\mathbf{z}_{t+l}|\mathbf{c}_t)}{p(\mathbf{z}_{t+l})}$ that is proportional to the mutual information. Then the density function is used to estimate the mutual information.

Specifically, by using the density function $f()$, InfoNCE defines a lower bound $\mathcal{L}_N(\mathbf{z}_{t+l},\mathbf{c}_t)$ of the mutual information $I(\mathbf{z}_{t+l},\mathbf{c}_t)$. The lower bound $\mathcal{L}_N(\mathbf{z}_{t+l},\mathbf{c}_t)$ can be written as:

$$\mathcal{L}_N(\mathbf{z}_{t+l},\mathbf{c}_t) = \mathbb{E}_{p(\mathbf{z})}\{log\frac{f(\mathbf{z}_{t+l},\mathbf{c}_t)}{\sum_j^N f(\mathbf{z}_j,\mathbf{c}_t)}\}, \tag{3.20}$$

where $\mathbf{z}_j$ is randomly sampled from $p(\mathbf{z})$.

The lower bound can be interpreted as follows, given a query $\mathbf{c}_t$ and a positive sample $\mathbf{z}_{t+l}$, the lower bound is a ratio between $f(\mathbf{z}_{t+l},\mathbf{c}_t)$ and $\sum_j^N f(\mathbf{z}_j,\mathbf{c}_t)$. The second term $\sum_j^N f(\mathbf{z}_j,\mathbf{c}_t)$ can be regarded as a normalisation over negative samples $\{\mathbf{z}_j\}_{j=1}^N$ of $f(\mathbf{z}_j,\mathbf{c}_t)$, where the negative samples $\{\mathbf{z}_j\}_{j=1}^N$ are randomly sampled from $p(\mathbf{z})$. Hence, what the lower bound is trying to do is to maximising the distances between the negative samples and the query, meanwhile, minimising the distances between the positive sample and the query, where the density function $f()$ can be regarded as a distance function.

For each future time step $l = 1,...,L$, Wav2vec defines a specific density function $f_l(\mathbf{z}_t,\mathbf{c})$. For the density function $f(\mathbf{z}_t,\mathbf{c})$, it is defined as:

$$f_l(\mathbf{z}_{t+l},\mathbf{c}) = exp(\mathbf{z}_{t+l}^{\top}\mathbf{W}_l\mathbf{c}_t), \tag{3.21}$$

where $\mathbf{W}_l$ is a trainable parameter.

Finally, the autoregressive predictive objective $\mathcal{L}_{arp}$ can be written as:

$$\mathcal{L}_{arp} = \mathbb{E}_{p(\mathbf{z})}\{-\sum_{t=1}^{T-L}\sum_{l=1}^{L} log\frac{exp(\mathbf{z}_{t+l}^T\mathbf{W}_l\mathbf{c}_t)}{\sum_j^N exp(\mathbf{z}_j^T\mathbf{W}_l\mathbf{c}_t)}\}. \tag{3.22}$$

Contrastive Loss and Masked Predictive Training

Wav2vec 2.0 used a BERT model as the feature aggregator, it followed the training method of the BERT model. The training method of the BERT model is the masked predictive training.

The masked predictive training randomly masks several frames of the representations $\mathbf{Z} = \{\mathbf{z}_t|t = 1,...,T\}$. Let the randomly sampled masking positions be $\mathbf{p} = \{p_0,...,p_s\}$, where $s < T$. Let the aggregated representations be $\mathbf{C} = \{\mathbf{c}_t|t = 1,...,T\}$. The training objective is to maximising the mutual information $I(\mathbf{z}_{t'},\mathbf{c}_{t'})$, where $t' = p_0,...,p_s$. The training objective $\mathcal{L}_{mp}$ can be written as follows:

$$\mathcal{L}_{mp} = \mathbb{E}_{p(\mathbf{z})}\{-\sum_{i\in\mathbf{p}} logI(\mathbf{z}_i,\mathbf{c}_i)\}. \tag{3.23}$$

Similar to Wav2vec and VQ-Wav2vec, the Wav2vec 2.0 uses the InfoNCE (Gutmann and Hyvärinen, 2010) to estimate the mutual information. Specifically, given the query $\mathbf{c}_i$, the positive sample $\mathbf{z}_i$ and a set of randomly sampled negative samples $\{\mathbf{z}_j \in p(\mathbf{z}) | j = 1, ..., N\}$, the InfoNCE objective can be written as:

$$\mathcal{L}_{mp} = \mathbb{E}_{p(\mathbf{z})}\{-\sum_{i \in \mathbf{p}} log \frac{exp(\mathbf{z}_i^{\top}\mathbf{W}_i\mathbf{c}_i)}{\sum_{j=1}^{N} exp(\mathbf{z}_j^{\top}\mathbf{W}_i\mathbf{c}_i)}\}. \tag{3.24}$$

L1 Loss and Autoregressive Predictive Training

APC and VQ-APC used autoregressive predictive training. They used a stack of the LSTM layers for the feature aggregator. They take in speech features as inputs. Let the speech features representations be $\mathbf{X} = \{\mathbf{x}_t | t = 1, ..., T\}$ and the aggregated representations be $\mathbf{C} = \{\mathbf{c}_t | t = 1, ..., T\}$. The autoregressive predictive training maximises the mutual information $I(\mathbf{x}_{t+l}, \mathbf{c}_t)$, where $l = 1, ..., L$. $I(\mathbf{x}_{t+l}, \mathbf{c}_t)$ is the mutual information between the aggregated feature $\mathbf{c}_t$ with its future L steps input frames $\{\mathbf{x}_{t'} | t' = t+1, ..., t+l, ..., t+L\}$.

In order to maximising mutual information, APC and VQ-APC use the L2 distance. Hence, the whole training objective of APC and VQ-APC can be written as follows:

$$\mathcal{L}_{apr} = \mathbb{E}_{p(\mathbf{X})}\{\sum_{t=1}^{T=L} ||\mathbf{x}_{t+l} - \mathbf{c}_t||_1\}. \tag{3.25}$$

L1 Loss and Masked Predictive Training

Mockingjay used the masked predictive training as the BERT model, which masked the representations and predicted the masked frames based on the unmasked frames. Mockingjay masks several frames of the representations $\mathbf{X} = \{\mathbf{z}_t | t = 1, ..., T\}$. Let the randomly sampled masking positions be $\mathbf{p} = \{p_0, ..., p_s\}$, where $s < T$. Let the aggregated representations be $\mathbf{C} = \{\mathbf{c}_t | t = 1, ..., T\}$. The training objective is to maximising the mutual information $I(\mathbf{x}_{t'}, \mathbf{c}_{t'})$, where $t' = p_0, ..., p_s$. The training objective $\mathcal{L}_{mpr}$ can be written as follows:

$$\mathcal{L}_{mpr} = -\mathbb{E}_{\mathbf{Z}}\{\sum_{i \in \mathbf{p}} log I(\mathbf{x}_i, \mathbf{c}_i)\}. \tag{3.26}$$

In order to achieve this, Mockingjay reconstruct input data and minimising the L1 distance between input data and reconstructed data,

$$\hat{\mathbf{x}}_i = \mathbf{W}_i\mathbf{c}_i. \tag{3.27}$$

Hence, the training objective of Mockingjay can be written as:

$$\mathcal{L}_{mock} = \mathbb{E}_{p(\mathbf{X})}\{\sum_{i \in \mathbf{p}} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||_1\}, \tag{3.28}$$

where $\mathbf{W}_i$ is a trainable parameter.

3.6 Datasets, Evaluation metrics and Challenges of Unsupervised Phonetic Representation Learning

3.6.1 Datasets

Speech datasets with large amounts of data are required for the unsupervised phonetic representation learning task. For example, the LibriSpeech dataset (Panayotov et al., 2015) has been used widely in many works (Schneider et al., 2019, Baevski et al., 2020, 2019, Hsu et al., 2021, Chorowski et al., 2019). Apart from the LibriSpeech dataset, the ZS challenge series also provided datasets for this task, including the ZS 2015 challenge (Versteegh et al., 2015), the ZS 2017 (Dunbar et al., 2017) and ZS 2019 (Dunbar et al., 2019). The details of the ZS 2019 datasets have already been introduced in Chapter 2, this section introduces only the datasets of the ZS 2015 and ZS 2017 challenges.

LibriSpeech

The LibriSpeech dataset is a speech corpus that consists of 1000 hours of read English data. The dataset contains speech samples obtained from audiobooks. The whole dataset contains 2484 speakers. The LibriSpeech dataset provides text transcriptions for speech recordings. It has been widely used for ASR (Amodei et al., 2016, Gulati et al., 2020). (Chorowski et al., 2019) used the LibriSpeech dataset for the unsupervised phonetic representation learning task. Also the LibriSpeech dataset has been widely used in various speech SSL models (Baevski et al., 2020, Hsu et al., 2021, Baevski et al., 2019).

The LibriSpeech dataset is composed of seven subsets, according to the quality of recordings. Table 3.3 presents the details of subsets of the LibriSpeech dataset.

ZS 2015 and ZS 2017

The ZS 2015 challenge (Versteegh et al., 2015) and the ZS 2017 challenge (Dunbar et al., 2017) are two episodes of the ZS challenge series. The objective of these two challenges was to learn unsupervised phonetic representations. The learned phonetic representations are

Table 3.3 The subsets of the LibriSpeech dataset, where "Hours" denotes the number of hours and "Speakers" denotes the number of speakers.

subset	Hours	Speakers
Train-clean-100	100.6	251
Train-clean-360	363.6	921
Train-other-500	496.7	1166
dev-clean	5.4	40
dev-other	5.3	33
test-clean	5.4	40
test-other	5.1	33

supposed to be invariant to the other information of speech signals such as speaker-dependent information. The ZS 2015 challenge dataset is composed of an English dataset and a Xisonga language dataset. The Xisonga language is a so-called ’low-resource’ language. The English dataset is composed of two subsets, a training subset and a test subset. The English training subset has two speakers, 40 minutes data per speaker. The English test subset contains 12 speakers, 16-30 minutes data per speaker. There is a test subset for the Xisonga language, which has 24 speakers, 2-29 minutes data per speaker.

The ZS 2017 challenge dataset is larger than the ZS 2015 challenge dataset. The ZS 2017 challenge dataset contains five languages: English, French and Mandarin and two other ’low-resource’ languages. The English dataset has 45 hours data with 60 speakers. The French dataset has 24 hours data with 18 speakers. The Mandarin dataset has 2.5 hours data with 8 speakers. The other two ’low-resource’ datasets have 25 hours data with 20 speakers and 10 hours data with 14 speakers, respectively.

3.6.2 Evaluation Metrics

In order to measure the phonetic discriminability of representations, there have been two types of metrics. One type is to evaluate the representations through distance comparison., for example, ABX score (Schatz et al., 2014, Ludusan et al., 2014). It has been widely used in the ZS challenge series. Another type of metric is to conduct a supervised phone classification task (Yang et al., 2021, Chorowski et al., 2019). Apart from the metrics that aim at evaluating the phonetic information and the speaker-dependent information of representations can also be evaluated, for example, a supervised speaker classification task (Yang et al., 2021, Chorowski et al., 2019).

ABX Score

The ABX score is an objective metric to assess the phonetic discriminability of representations. Given a list of testing speech samples, representations need to be extracted by models. A toolkit (Bernard, 2020) can be used to compute the ABX score based on extracted representations.

A test pair set is usually required for this metric. The test pair is composed of a number of triplet pairs of tri-phone speech segments. For example, a triplet pair can be represented as (**A**,**B**,**X**), where A is a tri-phone segment of 'beg', B is a tri-phone segment of 'bag' and X is another tri-phone segment of 'beg'. The only difference between the A,B and X is the middle phoneme. Note that A and B should come from the same speaker, while X should come from another speaker. Given a distance function $d()$ that measures the distance of two tri-phone segments, the ABX score is the percentage number that satisfies $d(\mathbf{A},\mathbf{X}) < d(\mathbf{B},\mathbf{X})$ for all the triplet pairs. The distance function is a cosine similarity distance based on a DTW alignment (Bellman and Kalaba, 1959). The ABX score ABX can be computed as follows.

$$ABX = \frac{\sum_{(\mathbf{A},\mathbf{B},\mathbf{X})} \mathbb{1}[d(\mathbf{A},\mathbf{X}) < d(\mathbf{B},\mathbf{X})]}{N_{ABX}}, \tag{3.29}$$

where N_{ABX} is the number of the triplet pairs, $\mathbb{1}$ denotes the indicator function. The lower ABX score means the better phonetic discriminability.

Bit-rate

The bit-rate metric (Dunbar et al., 2019) measures the entropy per second of representations. Formally, given a sequence of representation vectors $\mathbf{U} = \{\mathbf{s}_1, ..., \mathbf{s}_P\}$ of length P. The bit-rate $B()$ of $\mathbf{U}$ can be calculated as follows.

$$B(\mathbf{U}) = \frac{P \sum_{i=1}^{P} p(s_i) log p(s_i)}{D}, \tag{3.30}$$

where D is the duration (seconds) of the sequence $\mathbf{U}$. The higher bit-rate means the higher averaged entropy of representations.

Classification based Evaluation

Phone classification and speaker classification are commonly used in unsupervised phonetic representation learning works (Chorowski et al., 2019, Yang et al., 2021). These two metrics require a dataset with phoneme transcriptions and speaker identities, for example, the LibriSpeech dataset (Panayotov et al., 2015). On a whole or a subset of a dataset, representations need to be extracted from speech samples by trained representation learning models. Classification tasks are to be conducted on extracted representations.

For a phone classification task, extracted representations need to be split into a training set and a test set. A phone classification model can be a simple model. The model takes in representations and predicts A phoneme class for each frame. After the training of the classification model, a test accuracy can be reported as a result.

For a speaker classification task, extracted representations need to be split into a training set and a test. A speaker classification model can be a simple model. The model takes in representations and predicts a speaker class for the whole sequence of the representations. After the training of the classification model, the classification accuracy on a test set can be reported as the result.

3.6.3 Challenges: A Review of the ZS Challenge Series

The ZS 2015 challenge (Versteegh et al., 2015) and the ZS 2017 challenge (Dunbar et al., 2017) were the first two episodes of the ZS challenge series. The objective of these two challenges was to build speech processing techniques that can be applied to 'low-resource' languages, which usually are languages that do not have enough resources for training large ASR or TTS models. In order to achieve this, learning unsupervised phonetic representation is a solution. These two challenges set up benchmarks, provided datasets and evaluations, so that the performance of different techniques can be compared. The ZS 2019 challenge was the further episode after these two challenges, which has been introduced in Chapter 2.

In these two challenges, AUD tasks were conducted. The evaluation metric was the ABX score. In the ZS 2015 challenge, the best performing model (Chen et al., 2015) was a system based on the Dirichlet Process Gaussian Mixture Model (DPGMM) (Görür and Edward Rasmussen, 2010). The DPGMM model is a GMM model with Dirichlet process. Before this, the GMM based models have been exploited for learning unsupervised phonetic representations in various works (Kamper et al., 2014, Zhang and Glass, 2010, Lee and Glass, 2012). The DPGMM model produces frame-wise posterior grams, so that these posterior grams can be used as phonetic representations.

In the ZS 2017 challenge, a larger dataset was used for this challenge. This dataset contains speech recordings from five languages. AUD tasks were conducted within the dataset of each language. The DPGMM model (Heck et al., 2016) was one of the popular models in this challenge. There have also been several works (Chen et al., 2017, Shibata et al., 2017) using the bottleneck features of a DNN model as phonetic representations. Among the models, the DPGMM model (Heck et al., 2016) was still the best performing model. It is worthy to note that, the results of the VQ-WAE model of the ZS 2017 challenge was presented in (Chorowski et al., 2019). It was found that the VQ-WAE model outperformed the DPGMM model (Heck et al., 2016).

3.7 Summary

This chapter briefly reviews unsupervised phonetic representation learning. Motivations of applying disentanglement learning methods for learning phonetic representations from speech data are firstly introduced. Then, methods of learning disentangled phonetic representations are demonstrated, including normalisation, adaptation, VQ and SVQ. Moreover, two types of model architecture are reviewed. Auto-encoders and its two variants are compared in terms of learning disentangled phonetic representations. Several speech SSL models are also compared and analysed. Finally, this chapter reviews the datasets, evaluation metrics, challenges of unsupervised phonetic representation learning.

Chapter 4

Solving the Information Loss Issue for Auto-Encoder based VC Models

4.1 Introduction

As introduced in Section 2.4, auto-encoder based VC model is a type of text-free models. They (Qian et al., 2019, Hsu et al., 2016, Chou et al., 2018, Chou and Lee, 2019, Cho et al., 2019, Tjandra et al., 2019, 2020, Wu et al., 2020, Tang et al., 2022, Eloff et al., 2019, Liu et al., 2019, Wang et al., 2021, Wu and Lee, 2020) typically apply disentanglement learning methods to remove the speaker information in latent representations.

Many disentanglement learning methods have been found effective for this purpose. In the context of VC, Auto-Encoder based VC models mainly focused on exploring various disentanglement learning methods for one-shot VC. However, generally speaking, they can not reach the same level of performance of text-dependent VC models.

Disentanglement learning methods can also be exploited for learning unsupervised phonetic representations from speech. In a recent work (Chorowski et al., 2019), in the context of learning phonetic representations, the quality of learned representations of auto-encoder based models were analysed. It was found that the VQ-WAE model outperformed other models on preserving phonetic information and removing speaker information.

More recently, the AUD task and the VC task were combined in the ZS 2019 challenge (Dunbar et al., 2019). As introduced in Section 2.6, the VQ-WAE model (Cho et al., 2019) obtained good performance on the AUD task, however, performed poorly on the voice conversion task. The reasons for the poor VC performance of the VQ-WAE model have been discussed in the summary paper of the ZS 2019 (Dunbar et al., 2019). As a disentanglement learning method, the VQ in the VQ-WAE model compresses information. It works effectively on compressing speaker information but also hurts the linguistic information.

This chapter focuses on a multi-task scenario, including unsupervised phonetic representation learning and VC. The objective of this chapter is to solve the linguistic information loss issue of the VQ-WAE model and improve the VC performance of the VQ-WAE model. This work builds two models upon the VQ-WAE model. In order to solve the linguistic information loss issue, this work introduces two alternative disentanglement learning methods, to replace the VQ in the VQ-WAE model. The results are submitted to the ZS 2020 challenge (Dunbar et al., 2020) for evaluation.

4.1.1 Chapter Outline

The remainder of this chapter is organised as follows:

- Section 4.2 introduces the previous VQ-WAE model.
- Section 4.3 introduces the proposed IN-WAE model and SVQ-WAE model.
- Section 4.4 presents and discusses the results of the ZS 2020 challenge.
- Section 4.5 presents the motivation of the further analysis experiments and discusses the results of the further analysis experiments.
- Section 4.6 is the conclusion.

4.2 An Introduction of the VQ-WAE Model

The VQ-WAE model was proposed in (Chorowski et al., 2019). It was proposed for learning phonetic representations from speech data. It was further applied to VC in several works (Zhang, 2020b, Cho et al., 2019) and obtained competitive performance in several challenges (Dunbar et al., 2019, Yi et al., 2020).

The VQ-WAE model is an extension of the VQVAE model (Van Den Oord et al., 2017). The details of the VQVAE model has been introduced in Section 2.4. The VQ-WAE model takes in a sequence of speech features such as mel-frequency cepstral coefficients (MFCCs) (Mermelstein, 1976) and reconstructs the waveforms of the speech signal. The VQ-WAE model is composed of an encoder $Enc()$, a VQ block $VQ()$ and a decoder $Dec()$. The VQ-WAE model used a WaveNet (van den Oord et al., 2016) as the decoder. One difference of the VQ-WAE model with other VQVAE based models (Tjandra et al., 2019, 2020, Liu et al., 2019, Wu et al., 2020, Tang et al., 2022, Wu and Lee, 2020) is that it utilised a WaveNet as

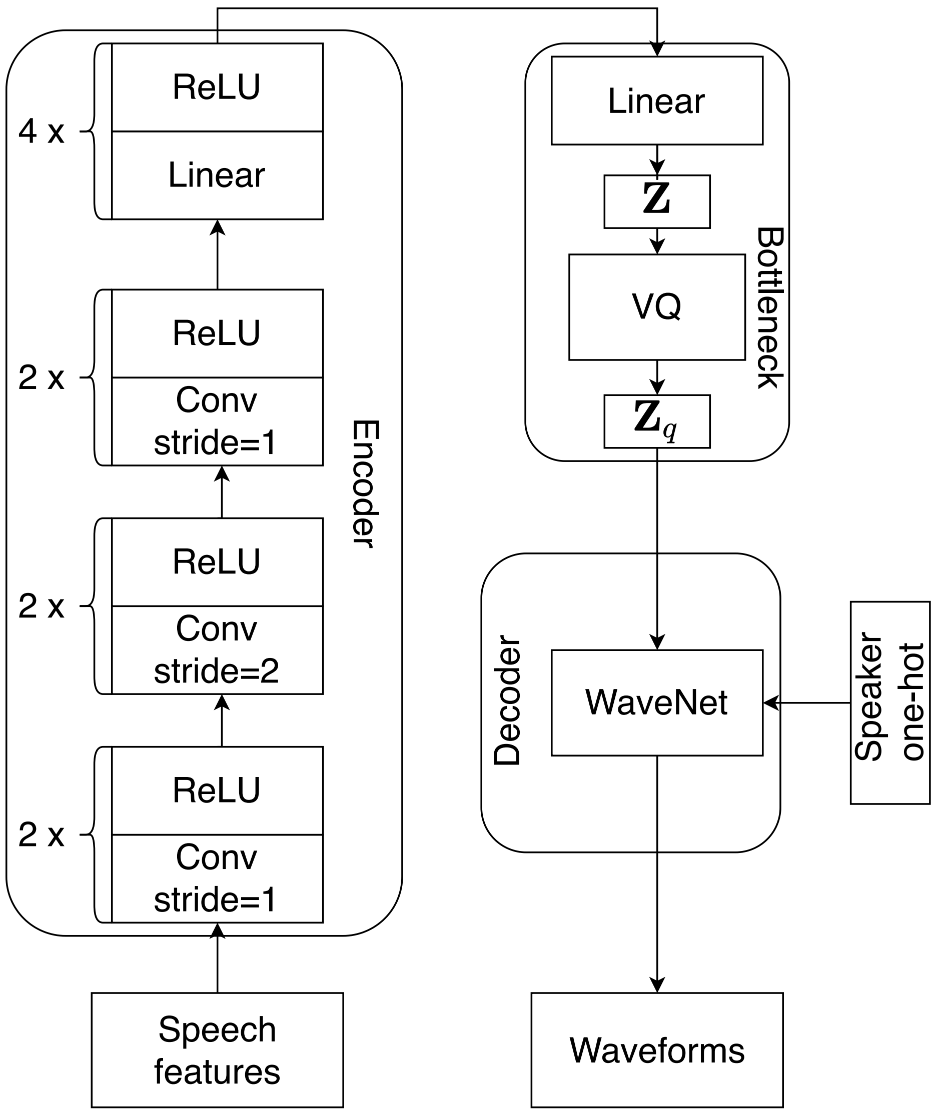

Fig. 4.1 The model architecture of the VQ-WAE model, where "Conv", "stride" and "ReLU" denote 1D-convolution layer, stride size and the ReLU activation layer (Glorot et al., 2011) respectively. "$\mathbf{Z}$" and "$\mathbf{Z}_q$" denote the latent representation and the discrete latent representations of the model, respectively.

the decoder. The WaveNet decoder directly reconstructs waveforms, while the other VQVAE based models reconstruct input speech features. The reason for this has been explained in (Chorowski et al., 2019). It was found that directly reconstructing waveforms improved the phonetic information modeling in the latent representations of the model. Figure 4.1 illustrates the model architecture of the VQ-WAE model.

Let speech features be $\mathbf{X} \sim \mathcal{X}$, the corresponding waveforms be $\mathbf{y} = \{y_1, ..., y_{T'}\} \sim \mathcal{Y}$ and the speaker identity one-hot vector be $\mathbf{s} \sim \mathcal{S}$, where $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{S}$ denote the distribution of speech features, waveforms and speaker identities, respectively.

At training time, the VQ-WAE model takes in the speech features $\mathbf{X}$, the speaker one-hot vector $\mathbf{s}$ and reconstructs the waveforms $\mathbf{y}$. The encoder encodes the speech features into latent representations $\mathbf{Z} = Enc(\mathbf{X})$. The VQ block quantises the latent representations $\mathbf{Z}$ to discrete latent representations $\mathbf{Z}_q = VQ(\mathbf{Z})$. The decoder takes in the discrete latent representations $\mathbf{Z}_q$ and the speaker one-hot vector $\mathbf{s}$ then reconstructs the waveforms $\hat{\mathbf{y}} = Dec(\mathbf{Z}_q, \mathbf{s})$.

At inference time, in the context of learning phonetic representations, the discrete latent representations are the outputs of the model. In the context of VC, a converted speech signal is needed. The decoder takes in quantised latent representations and a one-hot vector of a target speaker then generates waveforms.

4.2.1 Model Architecture of the Encoder

The encoder is used to encode input speech features into latent representations. The encoder contains six convolution blocks and four linear blocks. The convolution block is a cascade of a 1D-convolution layer and a ReLU activation layer (Glorot et al., 2011). The linear block is composed of a cascade of a linear layer and a ReLU activation layer. Figure 4.1 illustrates the model architecture of the encoder. In order to down-sample input speech features, down-sampling convolution layers are used in the encoder. The down-sampling convolution layer sets a stride size of 2, which down-samples the input features of a layer across time by 2 times. Hence, the frame rate of the latent representations can be controlled by the configurations of these down-sampling convolution layers in the encoder. The frame rate of latent representations means how many frames of features are used to represent the information of one second of speech signal. For example, a frame rate of 100 Hz means 100 frames are used to represent one second of speech signal.

The VQ-WAE model controls the frame rate through configuring the number of down-sampling 1D-convolution layers in the encoder. Specifically, in Figure 4.1, there are two convolution layers with a stride of 2. This implies that input speech features will be down-sampled by 4 times across time. If the frame rate of input speech features is 100 Hz, so that the frame rate of discrete latent representations becomes 25 (100 / 4 =25) Hz. To set the frame rate of latent representations to 50 Hz, only one down-sampling convolution layer is needed in the encoder.

4.2.2 Training Objectives of VQ-WAE

The VQ-WAE model is an extension of the VQVAE model, so that the training objective of the VQ-WAE model is similar to the VQVAE model. As introduced in Section 2.4 that the training objective of the VQVAE model is composed of three terms, which has been shown in Equation 2.15. The first term of Equation 2.15 is the reconstruction objective, which minimises the distance between the reconstructed data and the input data. The second and the third terms of Equation 2.15 are relevant to the VQ block.

In the VQ-WAE model, since the model reconstructs the waveforms of a speech signal, the reconstruction term becomes the distance between the reconstructed waveforms and the groundtruth waveforms. This training objective follows the training objective of the WaveNet model (van den Oord et al., 2016), which has been introduced in Section 2.2. So the waveform reconstruction loss $\mathcal{L}_{wav}$ can be written as follows.

$$\mathcal{L}_{wav} = \mathbb{E}_{\mathcal{Y},\mathcal{X},\mathcal{S}}\{-logp(\mathbf{y}|\mathbf{Z}_q,\mathbf{s})\}. \tag{4.1}$$

The second and third terms of Equation 2.15 are relevant to the VQ block, they minimise the L2 distances between latent representations and discrete latent representations. These two terms are reused in the training objective of the VQ-WAE model. The training objective of the VQ-WAE model can be written as follows.

$$\mathcal{L}_{vqwae} = \mathbb{E}_{\mathcal{Y},\mathcal{X},\mathcal{S}}\{-logp(\mathbf{y}|\mathbf{Z}_q,\mathbf{s}) + ||sg(\mathbf{Z}) - \mathbf{Z}_q||_2 + \lambda||\mathbf{Z} - sg(\mathbf{Z}_q)||_2\}, \tag{4.2}$$

where λ is a weight and the $sg()$ function is the stop gradient function. The stop gradient function stops the backward gradient propagation through a tensor in a neural network training, which can be written as:

$$sg(\mathbf{x}) = \begin{cases} \mathbf{x}, forward \\ 0, backward \end{cases} \tag{4.3}$$

4.3 An Introduction of IN-WAE and SVQ-WAE

This section introduces the two proposed models: IN-WAE and SVQ-WAE. The idea of these two models is to alleviate the discretisation of the VQ in the VQ-WAE model, in order to solve the linguistic information loss issue. Firstly, this section introduces the motivation of this idea. Then this section demonstrates the model architecture of IN-WAE and SVQ-WAE, respectively. Both IN-WAE and SVQ-WAE are based on the VQ-WAE model, so this section will mainly focus on the differences that have been made on the VQ-WAE model.

4.3.1 Alleviating the Discretisation of VQ

As introduced in the Section 2.6 that VQ-WAE performed well on the AUD task but bad on the VC task in the recent ZS 2019 challenge. Specifically, it suffered from a poor performance on naturalness and intelligibility.

Not only the VQ-WAE model, it has been observed that most of VQVAE based models suffered a poor performance on the VC task in the ZS 2019 challenge (Dunbar et al., 2019). As concluded in the summary paper (Dunbar et al., 2019) of the ZS 2019 challenge, there tended to be a trade-off between the discretisation of latent representations and the quality of converted speech signals. In other words, in order to remove speaker information, strong constraints are required. For example, in the VQVAE based models, the VQ blocks were configured with high discretisation. In other words, they typically used a small number of codebook vectors in the VQ, such as 256 (Cho et al., 2019) or even less (Liu et al., 2019). The high discretisation of the VQ block might be the cause of the linguistic information loss issue. This leads to speech signals produced by the decoder having poor naturalness and poor intelligibility.

In order to solve this issue, this work introduces two alternative disentanglement learning methods as the replacements of the VQ block. Both are trying to alleviate the discretisation of the VQ block. This work proposes Instance Normalisation WaveNet Auto-Encoder (IN-WAE) model and Sliced Vector Quantised WaveNet Auto-Encoder (SVQ-WAE) model.

IN-WAE is an extension of the VQ-WAE model but it does not use a VQ block. Instead, IN-WAE builds the model upon the auto-encoder model, which has continuous latent representations and is supposed to be able to preserve more phonetic information than the VQ block (Chorowski et al., 2019). The auto-encoder model has been found performing worse than the VQVAE model on removing speaker information (Chorowski et al., 2019). Hence, a disentanglement learning method is needed for removing speaker information for the IN-WAE model. This work proposes to introduce the IN layer and the AdaIN layer into the IN-WAE model. As introduced in Section 3.1 that speaker normalisation can be used for removing the speaker information. The IN layer has already been found effective for learning disentanglement in several auto-encoder based VC models (Chou and Lee, 2019, Chen et al., 2021b).

In order to alleviate the discretisation of the VQ block, a straightforward method would be increasing the number of codebook vectors. However, as discussed in Section 3.3 that there exists a trade-off between the information compression and the computation efficiency of VQ. Using a large number of codebook vectors would help to preserve information but would also increase the computation costs of VQ (Al-Mualla et al., 2002). Also, it has been concluded in (Chorowski et al., 2019) that "increasing the number of codebook vectors does not trivially lead to improved phone classification accuracies" of latent representations. This implies that directly increasing the number of codebook vectors might not enhance the phonetic information preservation of latent representations. Instead, this work proposes to apply SVQ in the SVQ-WAE model. It has been introduced in Section 3.3 that SVQ is an extension method of the VQ and is supposed to be able to preserve more information without hugely increasing the computation costs of the model. SVQ-WAE replaces the VQ block with a SVQ block, keeping the model architecture the same as the VQ-WAE model.

4.3.2 Model Architecture of IN-WAE and SVQ-WAE

The general idea of the IN-WAE model is to use continuous latent representations and apply the IN layer and the AdaIN layer for removing speaker information. Figure 4.2 presents the model architecture of the IN-WAE model. The main difference of IN-WAE is that it replaces the VQ block with an AdaIN layer. So the latent representations ($\mathbf{Z}_a$ in Figure 4.2) are continuous. Another difference is that IN-WAE introduces the IN layer to the encoder, which is shown in Figure 4.2. The IN layers in the encoder are supposed to remove speaker information. The AdaIN layer is used to adapt the latent representations to a target speaker. Besides, a speaker encoder is applied to extract speaker embeddings from speech features for the AdaIN layer. The speaker encoder contains five 1D-convolution layers, each of which is followed by a ReLU activation layer. Following the five 1D-convolution layers, the speaker encoder uses an average pooling layer and a linear layer. The speaker encoder outputs a single vector as a speaker embedding.

The model architecture of the SVQ-WAE model is illustrated in Figure 4.3. It reuses almost the same model architecture of the VQ-WAE model. The only difference is that SVQ-WAE replaces the VQ with the SVQ. This section omits the other details of the SVQ-WAE model, since the details of the SVQ-WAE model is the same as the VQ-WAE model, which has been introduced in Section 4.2.

4.4 ZS 2020 Challenge: Results and Discussions

The ZS 2020 challenge (Dunbar et al., 2020) is the following of the ZS 2019 challenge (Dunbar et al., 2019). The task set up of the ZS 2020 challenge is the same as the ZS 2019 challenge, which is composed of a VC task and an AUD task. This work submitted the results of the IN-WAE model and the SVQ-WAE model to the ZS 2020 challenge for evaluation. This section presents and discusses the results of the IN-WAE and the SVQ-WAE model in the ZS 2020 challenge.

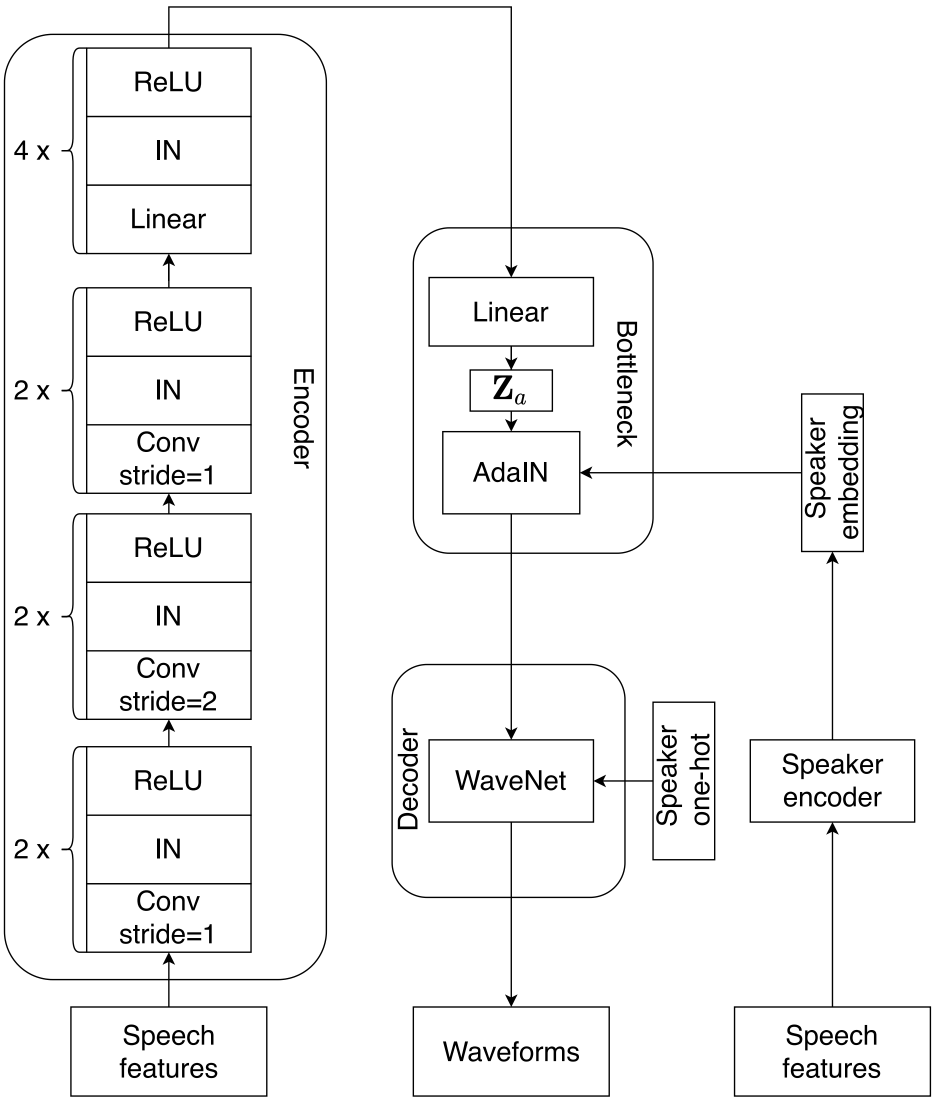

Fig. 4.2 The model architecture of the IN-WAE model, where "Conv", "stride" denotes the 1D-convolution layer and the stride size, "$\mathbf{Z}_a$" denotes the latent representations.

4.4.1 Experiment Setup

The objective of the ZS 2020 challenge is to develop unsupervised techniques for VC and AUD, so that these techniques can be applied to 'low-resource' languages that usually can not be represented by texts. In this challenge, a model needs to be developed for the two tasks. The dataset of the ZS 2020 challenge is the same as the ZS 2019 challenge, which has been introduced in Section 2.6. The dataset has an English dataset and an Austronesian dataset. The challenge participants need to develop their models on the English dataset, including the training and the configuration tuning of models. The tuned model configurations will be reused on the Austronesian dataset. The VC results and the AUD results of the two languages

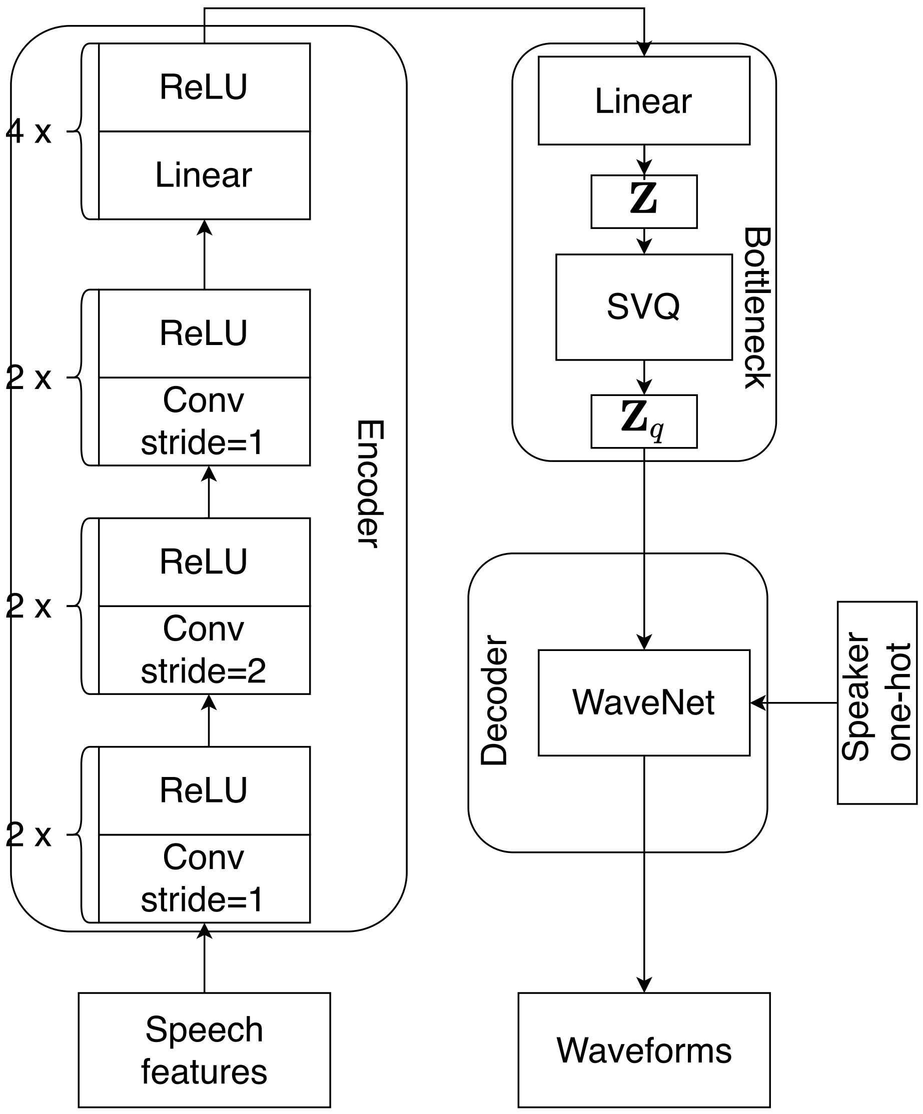

Fig. 4.3 The model architecture of the SVQ-WAE model, where "Conv" and "stride" denotes the 1D-convolution layer and the stride size,"$\mathbf{Z}_q$" denotes the discrete latent representations.

will be presented. The English dataset results will be regarded as validation results and the Austronesian dataset results will be regarded as testing results.

The evaluation metrics used in the ZS 2020 challenge are separated by the sub-tasks. For the AUD task, the ABX score and the bit-rate were used in this challenge, these two metrics have been introduced in Section 3.6. For the VC task, the MOS test was used for evaluating the naturalness and the speaker similarity of converted speech signals. In addition, a human transcribing test was used for evaluating the intelligibility of converted speech signals. These three evaluation metrics are subjective evaluation metrics, which have been introduced in Section 2.6.

4.4.2 Model Configurations and Implementations

There are no open-source official implementations of the VQ-WAE model. So this work implements the VQ-WAE model, the IN-WAE model and the SVQ-WAE model based on an open-source WaveNet implementation (Yamamoto, 2018).

The model configuration of the VQ-WAE model is kept the same as in the VQ-WAE paper (Chorowski et al., 2019). The model configurations of the WaveNet decoder of these three models are the same, following the WaveNet configurations in (Chorowski et al., 2019). Appendix A shows the details of the implementations of the model configurations of the VQ-WAE model, the IN-WAE model and the SVQ-WAE model, respectively.

This work follows the feature extraction process in (Chorowski et al., 2019), 13 dimensional MFCCs as well as the first and second derivatives are extracted as speech features. Waveforms are firstly re-sampled to a 16000 sampling rate. For MFCCs extraction, the hop size is 10 ms, the window size is 25 ms. The Librosa (McFee et al., 2015) toolkit is used for extracting MFCCs. The mean and variance normalisation is conducted on the MFCCs features. At training time, MFCCs are cropped into segments of 32 frames. This is conducted by randomly selecting a start frame of MFCCs of a speech utterance and cropping a segment of 32 frames. Accordingly, the start position of corresponding waveforms can be computed by the MFCCs start frame and the hop size. The corresponding waveform segment can be cropped, which has a length of 5200. The batch size is 10. The optimizer of the training process is the Adam optimizer (Kingma and Ba, 2015). The training process uses a learning rate of 4e-4. The training process takes 400k steps.

4.4.3 English Dataset Results

Table 4.1 The English dataset results. The bold results denote the best results in the challenge, where "BR", "MOS", "SS-MOS", "CER" denote bit-rate, mean opinion score, speaker similarity mean opinion score and character error rate, respectively. $\uparrow$ denotes the higher the better, $\downarrow$ denotes the lower the better.

Model	ABX (%) $\downarrow$	BR	MOS $\uparrow$	SS-MOS $\uparrow$	CER $\downarrow$
VQ-WAE	30.04	163.89	-	-	-
SVQ-WAE	26.06	385.75	2.88	2.35	0.47
IN-WAE	20.19	377.05	3.61	2.57	**0.18**

Table 4.1 presents the English dataset results and Table 4.2 shows the ranking positions in the challenge. In total there were 22 submissions in this challenge. Generally, the IN-WAE model obtained good performance on naturalness, intelligibility and phonetic discriminability.

Table 4.2 The ranking positions among 22 submissions of the IN-WAE model and the SVQ-WAE model on the English dataset.

Model	Metric	Ranking position
IN-WAE	ABX	3
	MOS	3
	SS-MOS	13
	CER	1
SVQ-WAE	ABX	5
	MOS	8
	SS-MOS	14
	CER	9

Specifically, the IN-WAE model got an ABX score of 20.19 % and a MOS result of 3.61, obtaining the third position for both the ABX and MOS. In addition, it obtained the first position for the CER, which was 0.18. Compared with the VQ-WAE model, the IN-WAE model obtained a relative improvement by 32% on the ABX score.

However, the IN-WAE model got a poor performance on speaker similarity, the SS-MOS result was 2.57 and the 13th position. Also it got a higher bit-rate than the VQ-WAE model. The increased bit-rate of the IN-WAE model shows its latent representations contain more information. As a summary of the IN-WAE model on the English dataset, although obtaining a competitive performance on naturalness, intelligibility and phonetic discriminability, the IN-WAE model sacrificed speaker similarity performance.

From these results, it is clear that by using continuous latent representations, the IN-WAE model enhances the ability of preserving phonetic information in latent representations. In addition, the naturalness performance and the intelligibility performance of the IN-WAE model was highly competitive. This may be due to the improved phonetic information preservation in latent representations. However, the poor speaker similarity performance of the IN-WAE model shows the latent representations of the model still contain speaker-dependent information. This implies the performance of removing speaker information of the IN layer was poor.

From Table 4.1 and Table 4.2, the SVQ-WAE model performed well on only one metric, the ABX score. It obtained an ABX score of 26.06%, getting the 5th position. Compared with the VQ-WAE model, the SVQ-WAE got a better ABX score with a relative improvement by 13%. However, the results show that the SVQ-WAE model could not obtain competitive performance on intelligibility, naturalness and speaker similarity. Although obtaining a slight improvement on preserving phonetic information in latent representations, the SVQ-WAE could not improve VC performance.

Table 4.3 The Austronesian dataset results. The bold results denote the best results in the challenge, where "BR", "MOS", "SS-MOS", "CER" denote bit-rate, mean opinion score, speaker similarity mean opinion score and character error rate, respectively. $\uparrow$ denotes the higher the better, $\downarrow$ denotes the lower the better.

Model	ABX (%) $\downarrow$	BR	MOS $\uparrow$	SS-MOS $\uparrow$	CER $\downarrow$
SVQ-WAE	16.47	384.23	2.28	2.50	0.55
IN-WAE	12.50	387.83	**4.06**	2.67	**0.15**

Table 4.4 The ranking positions among the 22 submissions of the IN-WAE model and the SVQ-WAE model on the Austronesian dataset.

Model	Metric	Ranking Position
IN-WAE	ABX	3
	MOS	1
	SS-MOS	10
	CER	1
SVQ-WAE	ABX	6
	MOS	12
	SS-MOS	13
	CER	10

4.4.4 Austronesian Dataset Results

Table 4.3 presents the results on the Austronesian dataset. Generally, the IN-WAE model got competitive performance on naturalness, intelligibility and phonetic discriminability. Specifically, the IN-WAE got a MOS score of 4.06 and a CER of 0.15, both obtaining the first positions. Additionally, it got the third place on the ABX score, which was 12.5%. However, on speaker similarity, it did not perform well. It got a speaker similarity MOS of 2.67, which was the 10th position. These results show a similar picture as on the English dataset. The IN-WAE model gained highly competitive performance on preserving phonetic information in latent representations, naturalness and intelligibility, while sacrificing speaker similarity performance.

On the Austronesian dataset, the SVQ-WAE model only performed well on the ABX score, but performed poorly on all the other metrics. The SVQ-WAE model got an ABX result of 16.47% , which was the 6th position. It got a MOS score of 2.28, a SS result of 2.5 and a CER result of 0.55, ranking at the 12th, 13th and the 10th position respectively.

4.4.5 Summary

In order to improve the VC performance of the VQ-WAE model, two models were proposed: the IN-WAE model and the SVQ-WAE model. The results of these two models were submitted to the ZS 2020 challenge for evaluation. The challenge evaluation results show that the IN-WAE model obtained good performance on naturalness, intelligibility and phonetic discriminability but poor performance on speaker similarity. The SVQ-WAE model obtained good phonetic discriminability performance. On the Austronesian dataset, the IN-WAE model obtained a MOS score of 4.06 and a CER of 0.15, achieving the first positions on naturalness and intelligibility. However, both of the two models suffered from poor speaker similarity performance. Besides, the SVQ-WAE model performed poorly on naturalness and intelligibility. It can be concluded that the IN-WAE model can improve the naturalness and the intelligibility performance. But both models suffered from a poor speaker similarity performance.

4.5 Further Analysis of Disentanglement Learning

4.5.1 Motivation

Both of the two proposed models performed poorly on speaker similarity. This observation leads to a hypothesis that there exists a trade-off between phonetic and speaker-dependent information of latent representations when alleviating the discretisation of the VQ. However, the evaluations of the challenge can not be repeated for further studies, because they utilised subjective evaluations.

In order to study the effects of the disentanglement learning methods proposed in this work, further analysis is needed. This work tries to answer two questions:

- When alleviating the discretisation of latent representations, will latent representations experience the entanglement between phonetic and speaker-dependent information?
- If so, will this entanglement impact the VC performance of the models?

4.5.2 Datasets

Instead of using the ZS 2020 challenge dataset, this section conducts the further analysis experiments on two other datasets: the LibriSpeech (Panayotov et al., 2015) dataset and the VCTK dataset (Yamagishi et al., 2019). The details of the LibriSpeech dataset have been introduced in Section 3.6. The details of the VCTK dataset have been introduced in Section

2.6. The reasons for changing the dataset are due to two issues of the ZS 2020 challenge dataset:

- The ABX score implies the phonetic discriminability of latent representations. It is a metric based on distance comparison, but it is not directly relevant to explicit phoneme labels. This work aims to evaluate the phonetic discriminability through groundtruth phoneme transcriptions. But the ZS 2020 dataset did not provide phoneme transcriptions.

- The challenge dataset contains limited target speakers (2 for the English dataset, 1 for the Austronesian dataset) for the VC task. The results are highly biased because the numbers of target speakers are small.

Different from the ZS 2020 challenge dataset, both the LibriSpeech dataset (Panayotov et al., 2015) and the VCTK dataset (Yamagishi et al., 2019) provide text transcriptions. Also the VCTK dataset contains 109 English speakers.

4.5.3 Experiment Setup

In order to answer the two questions mentioned before, the further analysis experiments conduct two tasks: unsupervised phonetic representation learning and VC. For the unsupervised phonetic representation learning task, the objective is to analyse the quality of latent representations of models. In other words, when alleviating the discretisation of latent representations, whether there exists the entanglement between phonetic information and speaker-dependent information. For the VC task, the objective is to study whether the entanglement would impact the VC performance.

This work selects two models as the baseline models. This work uses the VQ-WAE model as the baseline model. Besides, this work uses a variant model of the VQ-WAE model as the baseline model, namely WaveNet Auto-Encoder (WAE). The WAE model is the VQ-WAE model that removes the VQ block, which means the WAE model has continuous latent representations. The difference of the WAE model with the IN-WAE model is that the WAE model does not use the IN layer or the AdaIN layer, so that the effects of the IN layer and the AdaIN layer can be shown through a comparison between them.

Unsupervised Phonetic Representation Learning

The LibriSpeech train-clean-100h subset is used for the unsupervised phonetic representation learning task. In order to analysis the performance of the disentanglement learning on latent representations, two down-stream classification tasks are used: phone classification and speaker classification. This work uses the S3PRL toolkit (Yang et al., 2021) for implementing the phone and speaker classification tasks.

This experimental setup is similar to the experiment setup in the VQ-WAE paper (Chorowski et al., 2019), which also used phone classification and speaker classification for evaluations.

More specifically, the auto-encoder models are trained on the train-clean-100h subset of the LibriSpeech dataset. Then the models are used as a feature extractor. Latent representations are extracted and fed to a phone classification task and a speaker classification task, respectively. In the classification tasks, latent representations are used as training and testing inputs to classification models. After training the classification models, the testing accuracy results are reported. Hence, a good disentanglement learning method would lead to a high phone classification accuracy result and a low speaker classification accuracy result.

This work uses phoneme transcriptions provided by the S3PRL toolkit and the number of phonemes is 41. The phone classification model is a simple model that contains a single linear layer. The model predicts a phoneme class for each frame. For training the phone classification model, the batch size is 32, the learning rate is 2e-4 and the optimizer is Adam (Kingma and Ba, 2015). The phone classification model is trained for 200k steps, the best test accuracy result is reported.

For the speaker classification task, input latent representations are averaged across time. The averaged vector is fed into a speaker classification model. The model contains a single linear layer. For training the speaker classification model, the batch size is 32, the learning rate is 2e-4 and the optimizer is Adam (Kingma and Ba, 2015). The speaker classification model is trained for 300k steps, the best test accuracy result is reported.

VC

The VCTK dataset is used for the VC task. The auto-encoder based models are trained on the VCTK dataset. After training, the models are used to generate converted speech samples. Similar to the ZS 2020 challenge, this work evaluates converted speech samples from three aspects: naturalness, intelligibility and speaker similarity. In principle, a good disentanglement learning method would lead to good performance for the three aspects.

For the testing set of the experiments, this work selects 10 speakers (5 female and 5 male) from the VCTK dataset. For each speaker, a source speech sample is randomly selected, then this source sample will be converted to all the other 9 speakers. Hence, there will be 90 testing samples generated for evaluations.

For naturalness evaluations, a trained MOSNet model (Lo et al., 2019) is used. This work uses the speechmetrics (Liutkus et al., 2019) toolkit for the implementations of the MOSNet model. For one speech sample, the MOSNet model gives out a predicted MOS result. The averaged MOSNet result can be reported.

For speaker similarity evaluations, a trained speaker recognition model, ECAPA-TDNN (Desplanques et al., 2020), is used to conduct a speaker verification task. The speaker recognition model is trained on a mix of the VoxCeleb 1 dataset (Nagrani et al., 2017) and the VoxCeleb 2 dataset (Nagrani et al., 2020). To conduct the speaker verification task, this work produces a verification pair list on the VCTK dataset. The pair list is composed of positive pairs and negative pairs. The positive pair contains a converted speech sample and a real sample of its target speaker. The negative pair contains a converted speech sample and a real sample from a speaker that is not its target speaker. The speaker verification pair list contains 2600 pairs, including 1300 positive pairs and 1300 negative pairs. The EER of the speaker verification task is reported. This work uses the speaker verification recipe in the SpeechBrain toolkit (Ravanelli et al., 2021) and uses the model weights of the ECAPA-TDNN model provided by the SpeechBrain toolkit.

For intelligibility, a trained ASR model (Ravanelli et al., 2021) is used. The ASR model transcribes converted samples to texts. Then, a WER can be obtained and reported. This work uses a trained Transformer ASR model (Dong et al., 2018) provided by the SpeechBrain (Ravanelli et al., 2021) toolkit.

4.5.4 Experiment Results of IN-WAE

The unsupervised phonetic representation learning and VC experiment results of the IN-WAE model are presented in Table 4.5 and 4.6, respectively. This work compares the results of the IN-WAE model with the VQ-WAE model and the WAE model.

For the IN-WAE model, this work explores two factors: the frame rate of latent representations and the feature dimensionality (h) of latent representations. As in the VQ-WAE paper (Chorowski et al., 2019), the frame rate has been found being relevant to the phonetic information preservation of latent representations. In this work, three frame rates are explored: 25 Hz, 50 Hz and 100 Hz. Moreover, as found in AutoVC (Qian et al., 2019) that the latent feature dimensionality of an auto-encoder model can impact the disentanglement learning performance, this work also studies this factor. Considering the default value of latent feature dimensionality was 64 in the VQ-WAE model (Chorowski et al., 2019), this work explores three values of this factor: 32, 64 and 128.

Table 4.5 The results of unsupervised phonetic representation learning task of the IN-WAE model, the VQ-WAE model and the WAE model, where "h" denotes the feature dimensionality of latent representations, "r" denotes the frame rate of latent representations

Model	Phone (%) ↑			Speaker (%) ↓		
	h=32	h=64	h=128	h=32	h=64	h=128
IN-WAE (r=25)	46.8	45.3	44.3	12.0	10.1	9.3
IN-WAE (r=50)	50.1	53.0	52.5	44.9	72.0	65.3
IN-WAE (r=100)	48.9	51.4	50.7	24.0	46.9	39.5
VQ-WAE (r=25)	35.9	37.5	35.4	23.1	25.3	33.6
VQ-WAE (r=50)	37.0	39.7	34.4	4.5	22.7	26.7
VQ-WAE (r=100)	37.0	33.8	35.1	22.0	16.9	27.2
WAE (r=25)	52.0	47.9	53.2	84.0	72.0	81.2
WAE (r=50)	51.3	53.0	54.0	89.2	95.8	75.0
WAE (r=100)	48.9	50.3	53.5	91.3	94.6	98.7

Phonetic Representation Learning Results of IN-WAE

Table 4.5 shows the phone classification results and the speaker classification results of the IN-WAE model. Comparing the IN-WAE model to the WAE model, generally speaking, the IN-WAE model obtained lower phone accuracies and lower speaker accuracies than the WAE model. When the frame rate was 50 Hz or 100 Hz, the phone classification performance gap between them was small. However, when the frame rate was 25 Hz, the IN-WAE model performed worse than the WAE model. This implies that the IN layer might cause an information loss issue when latent representations were down-sampled to 25 Hz.

When the frame rate was 50 Hz or 100 Hz, it is clear that the IN layer can remove speaker information, while slightly hurting phonetic information. When the frame rate was 25 Hz, the speaker accuracies of the IN-WAE model were lower than the WAE model. This also indicates that when the frame rate was 25 Hz, latent representations of the IN-WAE model suffered from an information loss issue. This might be due to that down-sampling input speech features by 4 times caused latent representations being too short for the IN layer. As introduced in Chapter 1, it has been found in speech recognition that normalisation methods might suffer from an information loss issue when input utterance is too short.

Compared with the VQ-WAE model, it is clear that the IN-WAE model obtained better phone accuracies. From these results, the IN-WAE model showed an obvious advantage on phonetic information preservation over the VQ-WAE model.

As for speaker accuracies, the results of the VQ-WAE model stayed in a range from 16.9% to 27.2%, while the results of the IN-WAE model had a larger variance, varying from 9.3% to 72.0%. Despite of when the frame rate was 25 Hz, it is clear that that speaker classification results of the IN-WAE model were higher than the VQ-WAE model. These results indicate that, in the IN-WAE model, speaker-dependent information and phonetic information were highly entangled.

VC Results of IN-WAE

Table 4.6 VC results of the IN-WAE model, the VQ-WAE model and the WAE model, where "h" denotes the feature dimensionality of latent representations, "r" denotes the frame rate of latrent representations.

Model	WER (%) ↓			EER (%) ↓			MOSNet ↑		
	h=32	h=64	h=128	h=32	h=64	h=128	h=32	h=64	h=128
IN-WAE(r=25)	49.4	54.3	43.4	37.6	38.1	32.9	3.10	3.04	3.18
IN-WAE(r=50)	16.6	13.8	13.4	40.9	43.9	42.4	3.05	2.99	2.99
IN-WAE(r=100)	14.8	14.0	14.0	46.6	46.1	45.2	2.78	3.05	3.01
VQ-WAE(r=25)	73.2	71.6	81.9	15.1	14.1	15.0	3.27	3.32	3.42
VQ-WAE(r=50)	37.1	91.8	83.3	3.4	15.6	12.8	3.53	3.47	3.67
VQ-WAE(r=100)	52.5	85.5	72.8	12.8	13.0	15.5	3.48	3.23	3.35
WAE(r=25)	18.6	11.5	12.9	24.2	43.0	26.1	3.42	3.63	3.43
WAE(r=50)	10.7	8.9	10.1	44.4	48.1	43.4	3.36	3.51	3.67
WAE(r=100)	10.0	8.7	8.918.2	44.4	49.1	3.35	3.49	3.60	

Table 4.6 shows the results of the VC task of the IN-WAE, WAE and VQ-WAE model. Firstly, in order to show the effects of the IN layer and the AdaIN layer, the results of the IN-WAE model and the WAE model are compared. When the frame rate was 25 Hz, the IN-WAE model obtained worse results than the WAE model on all three metrics. This might be due to the IN layer suffering from an information loss issue at this frame rate.

When the frame rate was 50 Hz or 100 Hz, the IN-WAE model obtained slightly better speaker similarity performance than the WAE model. But the intelligibility performance of the IN-WAE model was slightly worse than the WAE model. From these results, it is clear that the speaker removal effects of the IN layer in the IN-WAE model were minor. And the speaker adaptation effects of the AdaIN layer were minor. The IN-WAE model obviously suffered from poor speaker similarity performance.

Comparing the IN-WAE model and the VQ-WAE model, generally, the IN-WAE model obviously outperformed the VQ-WAE model on intelligibility. However, on speaker similarity and naturalness, the IN-WAE model performed worse than the VQ-WAE model.

From the intelligibility results of the VQ-WAE model, it is clear that the converted speech samples of the VQ-WAE model could hardly preserve the phonetic information of a source speech sample. The WER results of the VQ-WAE model stayed at a range of 37.1% - 91.8%.

The IN-WAE model obviously improved the intelligibility performance of the VQ-WAE model. The WER results of the IN-WAE were in a range of 13.4%-54.3%.

However, for speaker similarity, it is clear the IN-WAE model obtained worse speaker similarity performance than the VQ-WAE model. The EER results of the IN-WAE model were in a range of 32.9%-46.6%. The VQ-WAE model obtained EER results that stay in a range from 12.8%-15.1%. The speaker similarity performance gap between these two models was obvious. These results show that the IN layer and the AdaIN used in the IN-WAE hurt speaker similarity performance, due to their poor speaker removal effects.

For naturalness, the IN-WAE model performed worse than the VQ-WAE. The VQ-WAE obtained MOSNet results that were in a range from 3.2-3.5. The MOSNet results of the IN-WAE model stayed in a range of 2.7-3.1.

Overall, for the VC task, the IN-WAE model brought an obvious improvement on intelligibility performance, but also caused a degradation of speaker similarity performance. For naturalness, the IN-WAE model performed poorly on this VC task.

It can be observed that the VC performance of the IN-WAE model in the ZS 2020 challenge was different from in this task. The IN-WAE model obtained good naturalness and intelligibility performance in the challenge evaluations. One possible reason is the differences of the two datasets. The challenge dataset has only 2 target speakers in the English dataset and 1 target speaker in the Austronesian dataset. The averaged data duration of the target speakers of the challenge dataset is 2 hours. These imply the challenge dataset is highly biased. However, in the VCTK dataset, there are more target speakers (109) and there is less data per speaker (25 min). As introduced in Chapter 1 that it has been found in speech recognition that adaptation methods might suffer from an information loss issue when data is insufficient. This implies that the AdaIN layer in the IN-WAE model might suffer from an information loss issue when data is insufficient. Also the duration of input speech features is 32 frames, which might be too short for the IN layer to estimate means and standard deviations.

4.5.5 Experiment Results of SVQ-WAE

In order to study the effects of the SVQ in the SVQ-WAE model, this work compares the results of the SVQ-WAE model with the VQ-WAE model. This work explores two factors for the SVQ-WAE model and VQ-WAE model: the number of slices N and the number of codebook vectors K in the VQ or SVQ. Since the SVQ is an extension of the VQ, when the number of slices N is one, the SVQ degenerates to the VQ. For the factor N, this work explores three options: 1, 2 and 4. For the factor K, this work explores five options: 64, 128, 256, 512, 1024. In order to control the other factors, this work uses a frame rate of 50 Hz and a feature dimensionality of 64. From Table 4.5, the VQ-WAE model obtained the highest phone classification accuracy under these configurations.

Phonetic Representation Learning Results of SVQ-WAE

Table 4.7 The latent representation learning results of the SVQ-WAE, where N denotes the number of Slices, K denotes the number of centroids in the VQ or SVQ block.

K	Phone (%)↑			Speaker (%)↓		
	N=1	N=2	N=4	N=1	N=2	N=4
64	33.5	27.6	38.8	26.2	19.2	32.9
128	39.7	34.6	38.4	34.7	28.5	30.7
256	33.4	41.1	44.8	21.7	39.7	52.4
512	37.0	37.0	43.6	22.7	26.4	31.3
1024	38.0	37.7	40.7	23.7	24.2	30.8

Table 4.7 shows the unsupervised phonetic representation learning results of the SVQ-WAE model and VQ-WAE model. Generally, compared with when N=1, the SVQ-WAE model with $N = 4$ obviously improved the phone classification results. But when $N = 2$, the SVQ-WAE model could only improve the phone accuracies when $K = 256$, 512 and 1024. This implies that when $N = 2$ and K= 64 and 128, the complexity of the SVQ was still not enough for preserving phonetic information. When $N = 4$ it is clear that the SVQ-WAE model could preserve more phonetic information than the VQ-WAE model.

In terms of removing speaker information, increasing N brought higher speaker classification accuracies, which means latent representations containing more speaker-dependent information. It can be observed that when increasing N from 1 to 2, the speaker accuracies increased. Then when increased N from 2 to 4, the speaker accuracies further increased. Especially, when K = 256, increasing N from 1 to 4 caused a boost of the speaker accuracy to 52.4%. From these results, when replacing the VQ with the SVQ, there existed the entanglement of phonetic information and speaker-dependent information in latent representations.

VC Results for SVQ-WAE

Table 4.8 shows the VC results of the SVQ-WAE model and the VQ-WAE model. Generally, when $N = 4$, the SVQ-WAE model improved the intelligibility performance and slightly enhanced the speaker similarity performance of the VQ-WAE model.

For naturalness, when N=4, the results of the SVQ-WAE model and VQ-WAE model stayed at similar levels. Both the two models obtained MOSNet results in a range of 3.3-3.7.

Table 4.8 The VC results of the SVQ-WAE model, where *N* denotes the number of slices, *K* denotes the number of centroids of the VQ or SVQ block.

K	WER (%)↓			EER (%)↓			MOSNet ↑		
	N=1	N=2	N=4	N=1	N=2	N=4	N=1	N=2	N=4
64	91.8	108.1	69.8	15.6	15.9	14.3	3.47	3.42	3.64
128	54.5	74.2	70.0	14.9	16.4	13.4	3.46	3.31	3.38
256	85.3	41.4	22.0	13.4	15.4	20.0	3.54	3.30	3.49
512	65.0	47.6	40.0	14.9	13.5	13.3	3.65	3.65	3.32
1024	60.7	72.2	59.7	16.0	11.9	13.3	3.45	3.70	3.64

This implies that although the phonetic information preservation of latent representation was improved, the SVQ could not enhance the naturalness performance of the model.

As for intelligibility performance, when $N = 1$, the WER results of the VQ-WAE model stayed at a range of 54.5%-91.8%. When increasing *N* from 1 to 2, the SVQ-WAE model did not show any improvements on the WER results, except from when $K = 256$. When increasing *N* from 1 to 4, the SVQ-WAE model obtained better WER results, which were in a range of 22.0%-70.0%.

In terms of speaker similarity, when $N = 2$ and $K = 64$, 128 and 256, the SVQ did not enhance the EER results. When $N = 2$ and K= 512 and 1024, the SVQ obtained better EER results. Furthermore, when $N = 4$, it is clear that the SVQ enhanced the EER results, except when $K = 256$ that the SVQ-WAE model obtained a worse EER result than the VQ-WAE model.

From these results, it can be observed that the phonetic information preservation of the SVQ is relevant to the complexity of the SVQ. When increasing *N* and *K*, the complexity of the SVQ would increase, so that more phonetic information can be preserved in latent representations. Especially when $N = 4$, the SVQ-WAE model tends to outperform the VQ-WAE model on intelligibility performance and speaker similarity performance. Hence, although it has been found that there exists the entanglement in latent representations, the SVQ-WAE model obtained better VQ performance than the VQ-WAE model.

On the challenge dataset, the SVQ-WAE model performed worse than the IN-WAE model. But on the VCTK dataset, it is clear that the SVQ-WAE model outperformed the IN-WAE model. This shows that when applying disentanglement learning methods in auto-encoder based VC models, the VC performance might be dependent on training data.

4.6 Conclusion

In this chapter, in order to solve the linguistic information loss issue of the VQ-WAE model, this work tried to alleviate the discretisation of latent representations of the VQ-WAE model. In order to achieve this, this work introduced two alternative disentanglement learning methods for replacing the VQ in the VQ-WAE model. This work proposed two models based on the VQ-WAE model, IN-WAE and SVQ-WAE.

To evaluate the proposed models, the results were submitted to the ZS 2020 challenge. Among 22 submissions, the IN-WAE model obtained a MOS score of 4.06 on the test dataset, winning the first position on naturalness. Meanwhile, it got a 0.15 CER result, winning the first position on intelligibility. Moreover, the SVQ-WAE model obtained a 6th position for the latent phonetic discriminability. However, both of the two models performed poorly on speaker similarity. The SVQ-WAE model also performed poorly on naturalness and intelligibility.

Based on the poor speaker similarity performance of the two proposed models, this work further hypothesed that alleviating the discretisation of the VQ caused the entanglement of phonetic information and speaker-dependent information in latent representations. In order to study this hypothesis, this work further conducted analysis experiments. The experiments were conducted on the LibriSpeech dataset and the VCTK dataset. The results showed that, in both of the two proposed models, there existed the entanglement of phonetic information and speaker-dependent information. In terms of the VC performance, this entanglement caused the IN-WAE model to perform poorly on the VC task, especially on speaker similarity and naturalness. The SVQ-WAE model improved the VC performance of the VQ-WAE model, despite the entanglement in latent representations.

In conclusion, the proposed IN-WAE model improved intelligibility and naturalness on the ZS 2020 dataset. Both the two proposed models brought the entanglement of phonetic information and speaker-dependent information in latent representations. Because of this entanglement, on the VCTK dataset, the IN-WAE model performed poorly on the VC task. But, the SVQ-WAE model improved the VC performance of the VQ-WAE model.

Chapter 5

Solving the Information Loss issue of GAN based VC Models

5.1 Introduction

GAN based VC model (Kaneko and Kameoka, 2017, Kaneko et al., 2019a, 2020, Kameoka et al., 2018, Kaneko et al., 2019b) is a type of text-free model. As introduced in Section 2.5 that the GAN based VC models (Kaneko et al., 2019a, 2020, 2019b) typically apply normalisation layers or adaptation layers as disentanglement learning methods. Generally speaking, GAN based models can not outperform text-dependent models in VC benchmarks (Yi et al., 2020, Lorenzo-Trueba et al., 2018), but one of its advantages is that it can be easily applied in a target domain where a large training dataset is not available.

StarGAN-VC2 (Kaneko et al., 2019b) is one of the recent proposed GAN based VC models (Kaneko et al., 2020, 2019b). In the recent VCC 2020 challenge, it was one of the best-performing GAN based models (Yi et al., 2020). As introduced in Section 2.5, the StarGAN-VC2 model is an extension of the StarGAN-VC model (Kameoka et al., 2018). It enhanced the speaker similarity performance of the StarGAN-VC model by introducing an adaptation layer, the CIN layer (Dumoulin et al., 2017).

As introduced in Section 3.2 that there have been many normalisation methods and adaptation methods used in speech tasks such as speech recognition (Huang et al., 2005, Gales, 1998, Liu et al., 1993, Viikki and Laurila, 1998, Eide and Gish, 1996). Some adaptation methods, such as FMLLR (Gales, 1998), adapt the input data of ASR models to a target speaker through a linear transform, which is similar to the CIN layer. When data is insufficient, the FMLLR has been found causing an information loss issue of adapted data, which would cause the ASR model suffering from an over-fitting issue (Joy et al., 2018).

This implies that the CIN layer might also have a similar information loss issue when data is insufficient.

One drawback of the StarGAN-VC2 model is that it has only been studied on one dataset, the VCC 2018 dataset (Lorenzo-Trueba et al., 2018). According to the StarGAN-VC2 paper (Kaneko et al., 2019b), a subset of the VCC 2018 dataset was used. The StarGAN-VC2 model was trained and tested on a dataset with only four speakers and 5 minutes data per speaker. Compared to other VC models (Liu et al., 2021, Huang et al., 2020a,c) that were commonly trained with more than 100 speakers and have been tested in various scenarios, the StarGAN-VC2 model still needs more studies and evaluations. It is not clear whether the performance of the StarGAN-VC2 model can be maintained when trained on various datasets. For example, when there are more speakers or there is less data per speaker, it is not clear whether the CIN layer would have an information loss issue.

The objective of this chapter is to evaluate the StarGAN-VC2 model on different training data situations, for example, more speakers and less data per speaker. This work establishes multiple training data situations with varying numbers of speakers and varying numbers of utterances per speaker. This work assesses whether the performance of the StarGAN-VC2 model can be maintained under these training data situations. It was observed in the preliminary experiments that the StarGAN-VC2 model suffered from poor performance and poor robustness. In order to improve the performance and the robustness of the StarGAN-VC2 model, this chapter proposes a new model, namely Weight Adaptive GAN VC (WAGAN-VC). The WAGAN-VC model introduces a new speaker adaptation layer, namely Weight Adaptive Instance Normalisation (WAdaIN), as a replacement of the CIN layer.

5.1.1 Chapter Outline

The remainder of this chapter is organised as follows:

- Section 5.2 introduces a new speaker adaptation layer, WAdaIN.
- Section 5.3 introduces the proposed WAGAN-VC model.
- Section 5.4 introduces the experiment setup.
- Section 5.5 presents and discusses the results of the experiment.
- Section 5.6 is the conclusion.

5.2 A New Model based Speaker Adaptation Method

5.2.1 Feature Space Adaptation and Model Space Adaptation

The WAdaIN layer was recently proposed in the StyleGAN2 model (Karras et al., 2020). As introduced in Section 3.2 that there have been many adaptation methods used in VC, for example the AdaIN layer (Huang and Belongie, 2017) and the CIN layer (Dumoulin et al., 2017). These two layers are similar, because both of them are an extension layer of the IN layer. The IN layer normalises intermediate features across time. These two layers extend the IN layer with a linear transform. This linear transform adapts normalised features to a target speaker. The difference between these two layers is that they use different methods to produce the parameters of this linear transform. The CIN layer produces the parameters from a speaker identity one-hot vector. While the AdaIN layer applies a speaker encoder neural network that extracts a speaker embedding from a sample from a target speaker, then the speaker embedding is used to produce the parameters.

In Section 3.2, speaker adaptation methods for model space (i.e. MLLR (Gales, 1998, Nguyen et al., 1999, Ferras et al., 2007, Gales et al., 1996)) and feature space (i.e. fMLLR (Varadarajan et al., 2008, Gales, 1998, Huang et al., 2005, Li et al., 2002, Huang and Sim, 2015)) have been introduced. It was discussed in (Gales, 1998) that model space speaker adaptation methods have advantages over feature space methods when adapting a speech recognition model to a target speaker. The reason was that the model based adaptation methods are more flexible.

In the context of VC, both the CIN layer and the AdaIN layer are regarded as feature space adaptation methods, because both act on intermediate features. In contrast, the WAdaIN layer acts on the kernels of a convolution layer in neural networks. Hence, the WAdaIN layer is regarded as a model space adaptation method.

5.2.2 WAdaIN

The idea of the WAdaIN layer is that, given kernels of a convolution layer and a target speaker, it firstly adapts the kernels to the target speaker, so that the adapted kernels are conditioning on the target speaker. Then the adapted kernels are applied in a convolution layer, which takes in intermediate features and adapts them to the target speaker. Figure 5.1 illustrates the details of WAdaIN. The process is composed of three steps: (1) kernel scaling, (2) kernel demodulation; (3) applying new kernels in convolution.

Let the kernels of a 1D-convolution layer be $\mathbf{W} \in \mathbb{R}^{I \times J \times K}$, where I denotes the output feature dimensionality, J denotes the input feature dimensionality , K denotes the kernel size.

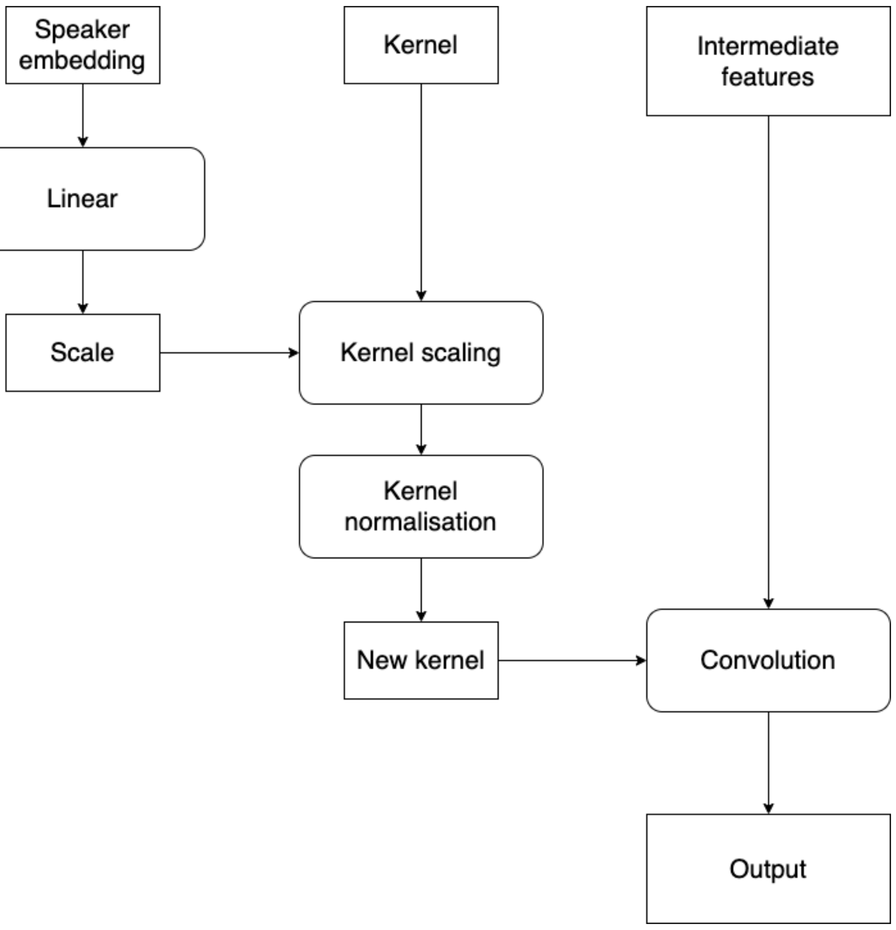

Fig. 5.1 The details of WAdaIN, where "Kernel" denotes the kernels of a convolution layer. "Linear" denotes linear layer, "Convolution" denotes 1D-convolution layer.

Before the training starts, the kernels are initialised randomly. In the first step, the kernels are scaled by a scale factor $\mathbf{s} \in \mathbb{R}^J$, which is produced from the speaker embedding $\mathbf{s}_{emb}$. Let $w_{i,j,k}$ be the element of $\mathbf{W}$, let s_j be the element of $\mathbf{s}$ and let $w'_{i,j,k}$ be the element of scaled kernels $\mathbf{W}'$. Then $w'_{i,j,k}$ can be obtained as follows.

$$w'_{i,j,k} = w_{i,j,k} s_j. \tag{5.1}$$

Similar to AdaIN, the scale factor $\mathbf{s}$ can be produced from a speaker embedding $\mathbf{s}_{emb}$ through a linear layer $Linear()$, which can be written as follows.

$$\mathbf{s} = Linear(\mathbf{s}_{emb}). \tag{5.2}$$

After scaling the convolution kernels, in the second step, the scaled kernel $\mathbf{W}'$ is then normalised. The scaled kernels matrix $\mathbf{W}'$ is divided by its standard deviation. Let $w''_{i,j,k}$ be

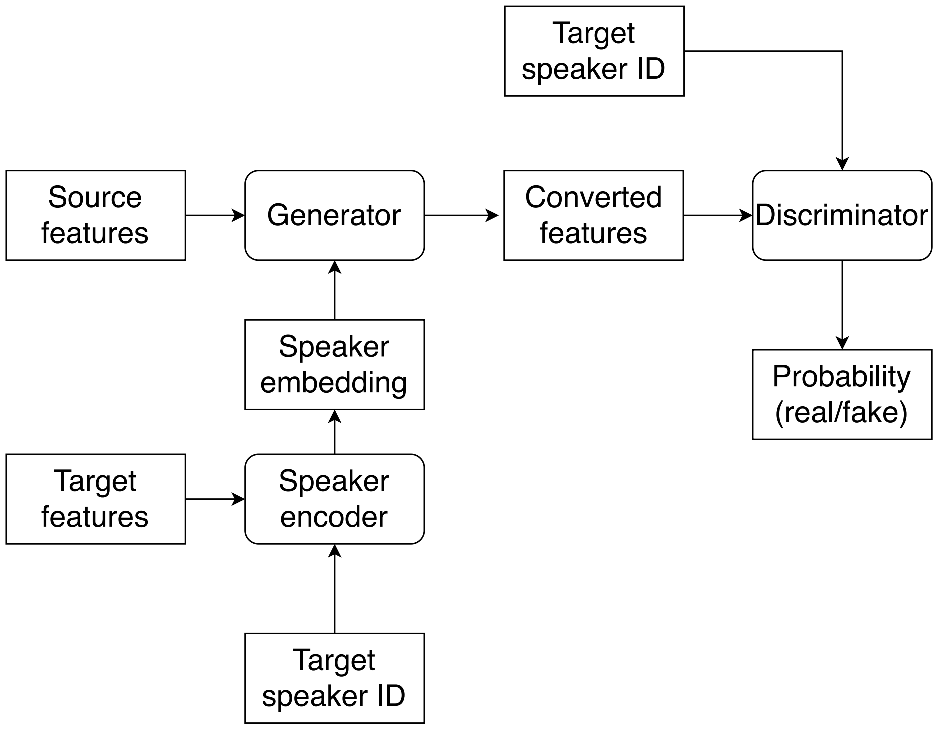

Fig. 5.2 The model architecture of the WAGAN-VC model.

the element of the normalised kernels $\mathbf{W}''$. Then $w''_{i,j,k}$ can be obtained as follows.

$$w''_{i,j,k} = \frac{w'_{i,j,k}}{\sigma}, \tag{5.3}$$

where σ is the standard deviation of $\mathbf{W}'$. The adapted kernels $\mathbf{W}''$ have the same matrix shape as the original kernels $\mathbf{W}$. After the kernel normalisation step, the normalised kernel can be applied in a convolution layer. This convolution layer takes in intermediate features then adapts them to a target speaker.

5.3 WAGAN-VC: A New StarGAN based VC Model

5.3.1 Overview

WAGAN-VC is built upon the StarGAN-VC2 model. The WAGAN-VC model is making mainly three changes on the StarGAN-VC2 model. The first is that the WAGAN-VC model introduces the WAdaIN layer into the generator, as a replacement of the CIN layer. The second change is that WAGAN-VC employs a speaker encoder, which extracts speaker embeddings for WAdaIN. Thirdly, WAGAN-VC updates the discriminator model architecture of the StarGAN-VC2 model.

WAGAN-VC is composed of a generator, a speaker encoder and a discriminator. Figure 5.2 illustrates the model architecture of the WAGAN-VC model. The training phase of the WAGAN-VC model is similar to the GAN model, which has been introduced in Section 2.5. At the training phase, the inputs of the model are a pair of source speech features and target speech features, as well as the speaker identities of a source speaker and a target speaker. The speaker encoder extracts a target speaker embedding from the target features. The generator takes in the source features and the target speaker embedding then generates converted features. The discriminator takes in the converted features and the target speaker one-hot vector and outputs a probability that the converted features are real speech features. At the inference phase, only the generator and the speaker encoder are used.

5.3.2 Training Objectives

Let the generator be $G()$, discriminator be $D()$ and the speaker encoder be $SEnc()$. Also let the source features be $\mathbf{X}^{src} \sim \mathcal{X}$, target features be $\mathbf{X}^{tgt} \sim \mathcal{X}$, source speaker identity one-hot vector be $\mathbf{s}^{src} \sim \mathcal{S}$ and target speaker identity one-hot vector be $\mathbf{s}^{tgt} \sim \mathcal{S}$, where $\mathcal{X}$ and $\mathcal{S}$ are the distribution of speech features and speaker identities, respectively.

Adversarial Objectives

The adversarial objectives of the WAGAN-VC model are similar to the StarGAN-VC model (Kameoka et al., 2018), which has been introduced in Section 2.6. The speaker encoder $SEnc()$ extracts a speaker embedding $\mathbf{s}_{emb}$ from the target features $\mathbf{X}^{tgt}$ and the target speaker one-hot $\mathbf{s}^{tgt}$, which can be written as follows.

$$\mathbf{s}_{emb} = SEnc(\mathbf{X}^{tgt}, \mathbf{s}^{tgt}). \tag{5.4}$$

The generator takes in the source features $\mathbf{X}^{src}$ and the speaker embedding $\mathbf{s}_{emb}$ and generates the converted features $\hat{\mathbf{X}}$, which can be written as follows.

$$\hat{\mathbf{X}} = G(\mathbf{X}^{src}, \mathbf{s}_{emb}). \tag{5.5}$$

The adversarial training objective of the discriminator can be written as follows.

$$\mathcal{L}^{D}_{wa\_adv} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{-D(\mathbf{X}^{src}, \mathbf{s}^{src})\} - \mathbb{E}_{\mathcal{X},\mathcal{S}}\{1 - D(\hat{\mathbf{X}}, \mathbf{s}^{tgt})\}. \tag{5.6}$$

For the generator and the speaker encoder, the adversarial objective can be written as:

$$\mathcal{L}^G_{wa\_adv} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{-D(\hat{\mathbf{X}}, s^{tgt})\}. \tag{5.7}$$

Cycle-Consistency Objective

Apart from the adversarial objectives, the WAGAN-VC model reuses the cycle-consistency objective and the identity-mapping objective, which have been used in the training of the StarGAN-VC model and the StarGAN-VC2 model. For the cycle-consistency objective, the generator takes in the converted features $\hat{\mathbf{X}}$ and the source speaker embedding then generates the back converted speech features $\mathbf{X}' = G(\hat{\mathbf{X}}, \mathbf{s}^{src}_{emb})$, where the source speaker embedding can be extracted by the speaker encoder as $\mathbf{s}^{src}_{emb} = SpeakerEnc(\mathbf{X}^{src}, \mathbf{s}^{src})$. The cycle-consistency objective minimises the L1 distance between $\mathbf{X}$ and $\mathbf{X}'$, which can be written as follows.

$$\mathcal{L}^G_{wa\_cyc} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{||\mathbf{X} - \mathbf{X}'||_1\}. \tag{5.8}$$

Identity-Mapping Objective

For the identity-mapping objective, the generator takes in the source features $\mathbf{X}^{src}$ and the source speaker embedding $\mathbf{s}^{src}_{emb}$ then reconstructs the source features $\mathbf{X}'' = G(\mathbf{X}, \mathbf{s}^{src}_{emb})$. The identity-mapping objective function minimises the distance between $\mathbf{X}''$ and $\mathbf{X}$, which can be written as follows.

$$\mathcal{L}^G_{wa\_id} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{||\mathbf{X}'' - \mathbf{X}||_1\}. \tag{5.9}$$

Speaker Embedding Reconstruction Objective

In addition, WAGAN-VC uses a speaker embedding reconstruction objective, in order to enhance the speaker similarity of the WAGAN-VC model. For the converted features $\hat{\mathbf{X}}$, the speaker encoder extracts a speaker embedding from it. Then the objective minimises the distance between the speaker embeddings of the converted features and the target features. The speaker embedding reconstruction objective can be written as follows.

$$\mathcal{L}^G_{spk} = \mathbb{E}_{\mathcal{X},\mathcal{S}}\{||\mathbf{s}_{emb} - SpeakerEnc(\hat{\mathbf{X}}, \mathbf{s}^{tgt})||_1\}. \tag{5.10}$$

Overall Training Objectives

Overall, the training objectives of StarGAN-VC2 for the generator and discriminator can be written respectively as:

$$\mathcal{L}^{G}_{wa} = \mathcal{L}^{G}_{wa\_adv} + \lambda_{cyc}\mathcal{L}^{G}_{wa\_cyc} + \lambda_{id}\mathcal{L}^{G}_{wa\_id} + \lambda_{spk}\mathcal{L}^{G}_{spk}, \tag{5.11}$$

$$\mathcal{L}^{D}_{wa} = \mathcal{L}^{D}_{wa\_adv}, \tag{5.12}$$

where λ_{cyc}, λ_{id} and λ_{spk} are hyper-parameters.

5.3.3 Model Architecture of WAGAN-VC

Generator

The generator of WAGAN-VC reuses the "2-1-2D" architecture of the StarGAN-VC2 model, which has been introduced in Section 5.2. The "2-1-2D" architecture means that the generator is composed of a cascade of down-sampling 2D-convolution blocks, bottleneck 1D-convolution blocks and up-sampling 2D-convolution blocks. WAGAN-VC keeps mostly the same generator model architectures as the StarGAN-VC2 model. The difference is that the WAGAN-VC replaces the CIN layers in bottleneck blocks of the StarGAN-VC2 model with the WAdaIN layers.

Figure 5.3 illustrates the model architecture of the generator. The generator is composed of an input block, two down-sampling blocks, nine bottleneck bottleneck blocks, two up-sampling blocks and one output layer. Specifically, the input block contains a 2D-convolution layer and a Gate Linear Units (GLU) (Dauphin et al., 2017) activation layer. The down-sampling block contains a 2D-convolution layer with a stride of 2, an IN layer and a GLU activation layer. The bottleneck block contains a WAdaIN layer and a GLU activation layer. The up-sampling block contains a transposed 2D-convolution layer (Radford et al., 2016), an IN layer and a GLU activation layer. The output layer is a 2D-convolution layer.

Discriminator

This work slightly updates the model architecture of the discriminator of the StarGAN-VC2 model. The discriminator of the StarGAN-VC2 model is introduced in Appendix B. Briefly, it is composed of a stack of 2D-convolution layers with the IN layers. The model architecture of the discriminator follows the discriminator architecture of the StarGAN-V2 model (Choi et al., 2020), which was originally proposed for an image style transfer task. Compared to the discriminator of the architecture of the discriminator of the StarGAN-VC2

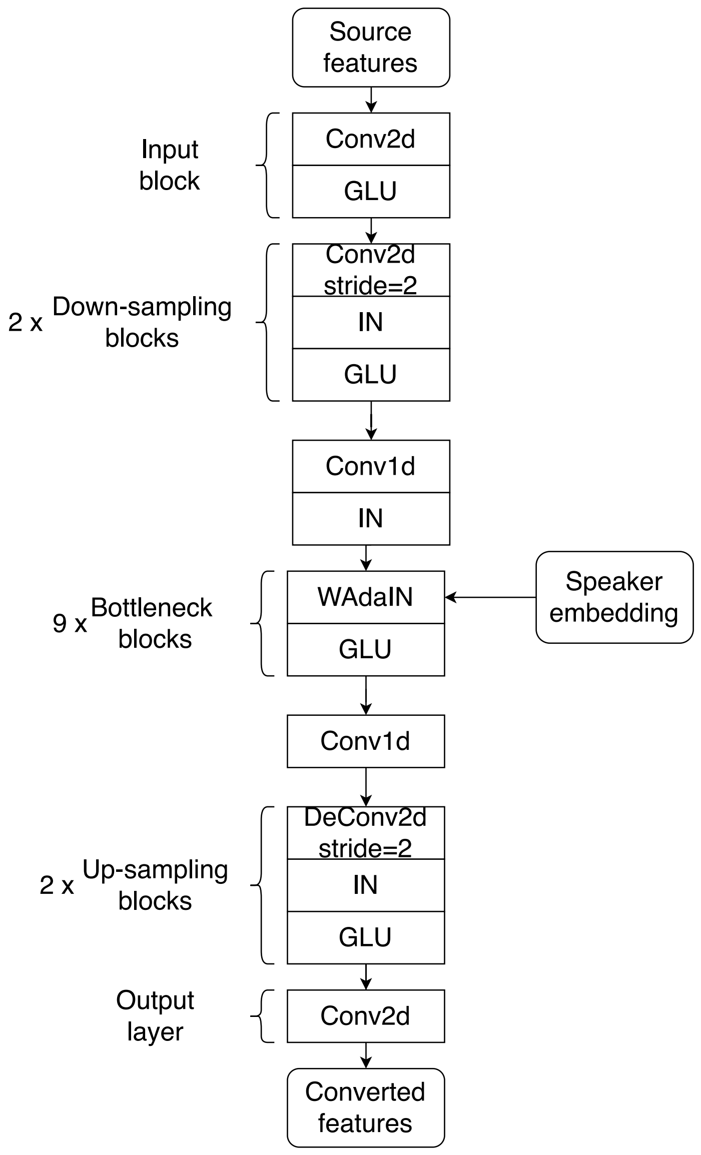

Fig. 5.3 The generator model architecture of the WAGAN-VC model, where "Conv2d", "stride", "DeConv2d" denote 2D-convolution layer, the stride size and transposed 2D-convolution layer (Radford et al., 2016), respectively.

model, the discriminator of the WAGAN-VC model does not use the IN layer. Because the IN layer removes speaker information, applying it in the discriminator might impact the ability of the discriminator to distinguish the speaker identity of the converted features. H The discriminator takes in converted features produced by the generator, a speaker one-hot vector and outputs a probability that the converted features are real speech features. The discriminator contains four down-sampling blocks. The down-sampling block contains a 2D-convolution layer with a stride of 2 and a ReLU activation layer. As shown in Figure 5.4,

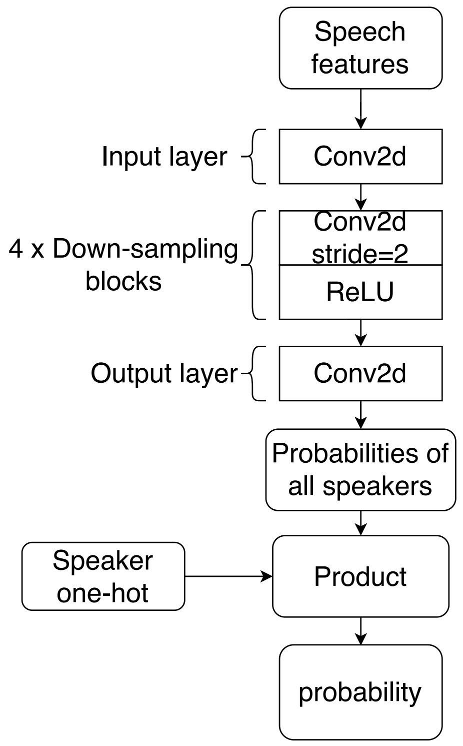

Fig. 5.4 The discriminator model architecture of the WAGAN-VC model, where "Conv2d" and "stride" denote 2D-convolution layer and stride size, respectively.

the output of the down-sampling blocks is fed to the output layer. The output layer outputs the probabilities of all speakers, which is a vector. The output dimensionality of the output layer is set to be the number of speakers in training data. Next, the probability of the target speaker can be obtained by an element-wise product of the target speaker one-hot vector and the all speaker probabilities. The output of the discriminator is the probability of the target speaker.

Speaker Encoder

The model architecture of the speaker encoder also follows the StarGAN-V2 model (Choi et al., 2020). The speaker encoder takes in speech features, a speaker one-hot vector and outputs a speaker embedding. The speaker encoder is composed of five down-sampling blocks, an average pooling layer and a list of linear layers. The number of linear layers in this list is set to the number of speakers in training data.

The down-sampling block is composed of a 1D-convolution layer with a stride of 2, and a ReLU activation layer. The output of the down-sampling blocks is fed to an average pooling

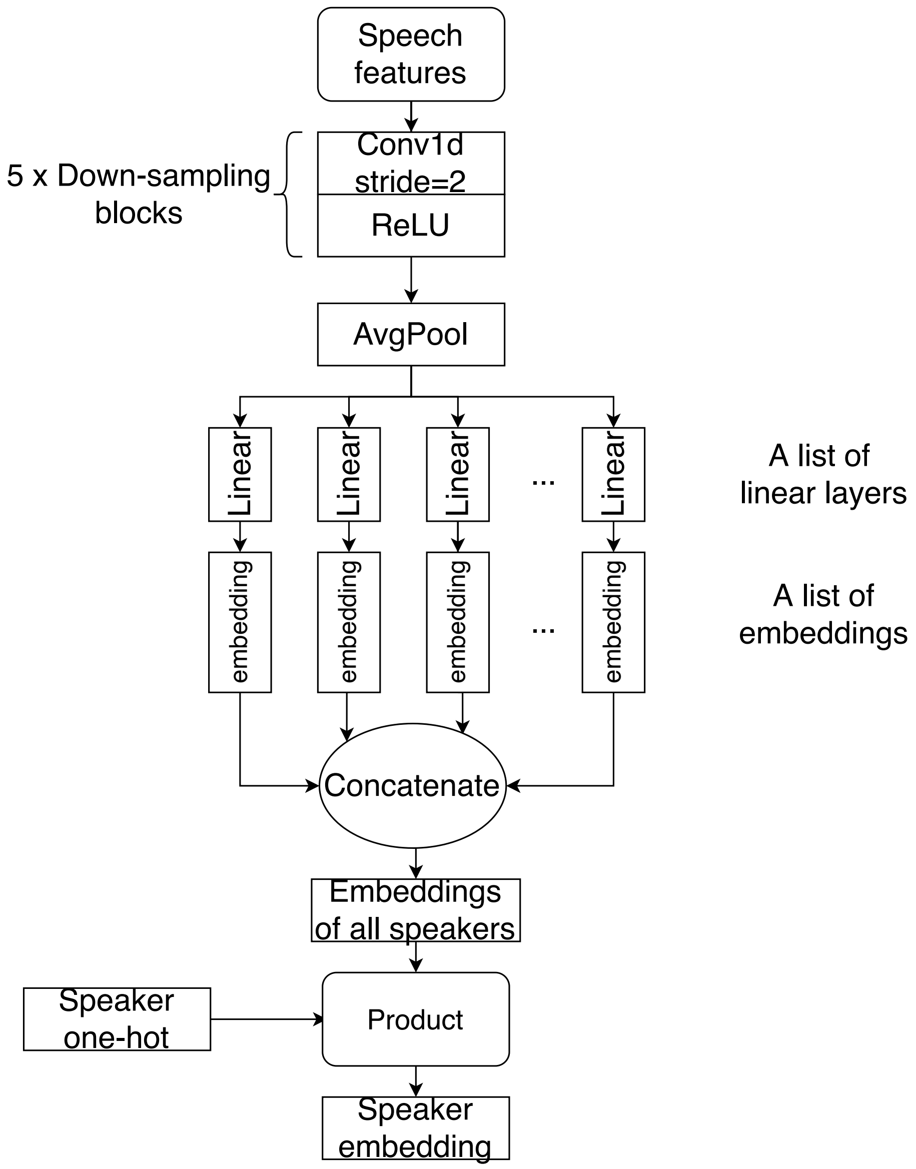

Fig. 5.5 The speaker encoder model architecture of the WAGAN-VC model, where "Conv1d", "stride" and "AvgPool" denote 1D-convolution layer, stride size and average pooling layer.

layer, which conducts an average operation across time. As shown in Figure 5.5, the output of the average pooling layer is a vector.

The speaker encoder contains a list of linear layers, each of which is aligned with a speaker in training data. The output vector of the average pooling layer is the input to each of these linear layers. Then these linear layers output a list of speaker embeddings. These speaker embeddings can be regarded as the speaker embeddings of all speakers in training data. Next, these speaker embeddings are concatenated into a matrix, where each row vector is a speaker embedding. The speaker embedding of the speaker can be obtained by a product of the speaker embedding matrix and a speaker one-hot vector.

5.4 Experiment Setup

This work uses the VCTK dataset (Yamagishi et al., 2019) for the experiments. As introduced in Section 2.6, this dataset contains 109 speakers and the data duration per speaker is 25 minutes. These attributes allow this work to establish various training data situations based on the VCTK dataset. These data situations are used for evaluating the performance and the robustness of the models.

5.4.1 Evaluation Metrics

This work evaluates the VC performance from three aspects: naturalness, intelligibility and naturalness. Three objective evaluation metrics are employed.

For naturalness evaluations, this work uses the same objective metric used in Section 4.5. A trained MOSNet model (Lo et al., 2019) is used, this work uses the speechmetrics (Liutkus et al., 2019) toolkit for the MOSNet implementation. For each speech sample, the MOSNet model gives out a score, then the scores are averaged over the whole test set, the averaged MOSNet result is reported.

For the intelligibility, a trained ASR model (Ravanelli et al., 2021) is used. The ASR model transcribes converted samples to texts. Then a WER can be computed and reported. This work uses a trained Transformer ASR model (Dong et al., 2018) provided by the SpeechBrain (Ravanelli et al., 2021) toolkit. This objective metric is the same as in Section 4.5.

For the speaker similarity, a Xvector (Snyder et al., 2018) model is trained on the VCTK dataset. Then it is used as a speaker classification model. Converted samples are fed into the Xvector model and the Xvector model predicts a speaker identity. Therefore, the accuracy of speaker classification can be reported. The higher accuracy implies a better speaker similarity.

5.4.2 Model Configurations and Implementations

For the StarGAN-VC model, since no official source code implementation is available, this work used an unofficial implementation (Liu, 2018). For the StarGAN-VC2 model, due to there is no official source code implementation, this work implemented the StarGAN-VC2 model according to the StarGAN-VC2 paper (Kaneko et al., 2019b). The model configurations of the StarGAN-VC model, the StarGAN-VC2 model and the WAGAN-VC model are presented in Appendix B.

The models in this work share the same speech features. For speech features extraction, this work down-samples waveforms into 22.05 KHz sampling rate. Then 36-dimensional MCEPs features are extracted using the PyWorld toolkit. (Morise et al., 2016). Following the implementations in the StarGAN-VC paper (Kameoka et al., 2018) and the StarGAN-VC2 paper (Kaneko et al., 2019b), the MCEPs features, logarithmic fundamental frequencies (log-F0s) and aperiodicities (APs) are extracted.

At the inference phase, the models are used to convert the MCEPs of a source speech to a target speaker. Then log-f0s of the source speech signal is adapted to the target speaker through a linear transform. The parameters of this linear transform are the means and the standard deviations of the log-F0s of the source speaker and the target speaker. After getting converted MCEPs, linearly transformed log-F0s and APs of the source speech signal, the WORLD (Morise et al., 2016) vocoder synthesises a converted speech signal.

The training and the inference are implemented using the PyTorch toolkit (Paszke et al., 2019). Mean variance normalisations are conducted for speech features. For the StarGAN-VC model, the training batch size is 32, speech features are cropped into 256-frame segments. This is conducted by randomly selecting a start frame position of speech features and cropping a segment of 256 frames. The learning rate of the generator, the discriminator and the speaker classifier is 2e-4, 1e-4 and 1e-4, respectively. The hyper-parameter λ_{cyc}, λ_{id} and λ_{cls} is 10, 5, 10, respectively. The Adam optimiser is used for training. The training of the StarGAN-VC model takes 200 K steps. The identity mapping objective is dropped after 10 K training steps.

For the StarGAN-VC2 model, the training batch size is 8, speech features are cropped into 128-frame segments. The learning rate of the generator and the discriminator is 2e-4 and 1-e4. The hyper-parameters λ_{cyc}, λ_{id} are 10 and 5, respectively. The Adam optimiser is used for training. The training of the StarGAN-VC2 model takes 200 K steps. The identity mapping objective is dropped after 10 K training steps.

For the WAGAN-VC model, the training batch size is 8, speech features are cropped into 256-frame segments. The learning rate of the generator and the discriminator is 2e-4 and 1-e4. The hyper-parameters λ_{cyc}, λ_{id} and λ_{spk} are 4, 2 and 1, respectively. The Adam optimiser is used for training. The training of the WAGAN-VC model takes 100 K steps. The identity mapping objective is dropped after 10 K training steps.

5.4.3 Experiment Setup

The objective of this chapter is to evaluate the models on various training data situations. Specifically, two factors are explored: the number of speakers and the number of utterances per speaker. Let the number of speakers be M and the number of utterances per speaker be N.

This work set up two VC experiments. The first experiment is to explore M and N with the whole number of training utterances keeping fixed. To achieve this, six data situations are established. In these data situations, M is increasing and N is decreasing. The second experiment keeps M fixed and decreases N. This experiment is to explore the performance and the robustness of the models in low-resource data situations. Next, in order to better understand the effect of each component of the WAGAN-VC, this work conducts ablation studies.

One key motivation of this work is to evaluate whether models can maintain performance in different data situations. In order to show the robustness of models, means and standard deviation of results over all of the six data situations are presented. A good-performing robust model is expected to get a good mean result and a small standard deviation.

Exploring Training Data Situations

Table 5.1 The details of six data situations for exploring M and N.

Index	M	N	Total Utterances
1	10	350	3500
2	20	180	3600
3	40	90	3600
4	60	60	3600
5	90	40	3600
6	109	35	3815

The details of six data situations are presented in Table 5.1. This work firstly sets up a base data situation with $M = 10$ and $N = 350$. Then this work increases M and decreases N gradually. All these six data situations have a similar number of total training utterances, which is around 3600.

For the test data of this experiment, ten speakers are randomly selected from the VCTK dataset. For each test speaker, ten testing samples are randomly selected as source speech samples. Each speaker is converted to all the other nine test speakers. Hence there are 90 ($9 \times 10 = 90$) source-to-target mappings. For each source-to-target mapping, ten source samples will be used. Totally, there are 900 converted samples for evaluation. The 900-samples test list will be reused in the following experiments.

Exploring Low-Resource Data Situations

Table 5.2 The detailed information of the four low-resource data situations.

Index	M	N	Total Utterances
1	109	35	3815
2	109	20	2180
3	109	10	1090
4	109	5	545

The objective of this experiment is to explore whether the models can maintain the performance in low-resource data situations, which could happen in practical applications (Baas and Kamper, 2022, 2021, Benisty et al., 2016). This work sets up four low-resource data situations. The details of low-resource data situations are presented in Table 5.2. In these low-resource data situations, M is fixed to 109 and N is decreased from 35 to 5 gradually. The total numbers of utterances also decreased from 3815 to 545 gradually. In evaluations, this experiment reuses the 900-samples test list. In order to show an overall performance of the models, the means of results over all data situations are presented. In addition, in order to show the robustness of the models, the standard deviations of results are presented.

Ablation Study: The Effects of the WadaIN Layer

In order to study the effects of the WadaIN layer, this work compares the WAdaIN layer and the AdaIN layer. Specifically, this work additionally builds a variant of the WAGAN-VC model, namely WAGAN-VC (AdaIN-1). The WAGAN-VC (AdaIN-1) model replaces the WAdaIN layers with the AdaIN layers and keeps all the other model configurations.

In preliminary experiments, it was observed that the WAGAN-VC (AdaIN-1) model performed much poorly than the WAGAN-VC model, which could be due to the hyper-parameter settings. So this work further builds another variant of the WAGAN-VC model, namely WAGAN-VC (AdaIN-2). The WAGAN-VC (AdaIN-2) model increases the hyper-parameters λ_{cyc} to 10 and λ_{id} to 5. Increasing them enhances the objectives for preserving the phonetic information of source speech features. Note that in the WAGAN-VC model and the WAGAN-VC (AdaIN-1) model, λ_{cyc} is set to 4 and λ_{id} is set to 2. The results of the data situations in Table 5.1 will be presented.

Ablation Study: The Effects of the Speaker Encoder

The WAGAN-VC model uses a speaker encoder while the baseline models StarGAN-VC and StarGAN-VC2 do not. Therefore, this work studies the effects of the speaker encoder.

Following StarGAN-VC and StarGAN-VC2 that used a speaker one-hot vector for a target speaker, this work builds a variant of the WAGAN-VC model, namely WAGAN-VC (1hot-1) model. The WAGAN-VC (1hot-1) model removes the speaker encoder and feeds a speaker one-hot vector to the WAdaIN layers.

In preliminary experiments, the WAGAN-VC (1hot-1) model suffered from an overfitting issue. Hence this work further builds another variant of the WAGAN-VC model, namely WAGAN-VC (1hot-2) with different hyper-parameters. The WAGAN-VC (1hot-2) model increases the hyper-parameters λ_{cyc} to 10 and λ_{id} to 5, which enhances the objectives for preserving the phonetic information of source speech features. The results of the data situations in Table 5.1 will be presented.

Ablation Study: The Effects of the Discriminator

The WAGAN-VC model introduces a new model architecture for the discriminator. The model architecture of this discriminator is different from the discriminator used in the StarGAN-VC2 model. In order to study the effects of the new discriminator model architecture, this work builds up a variant of the WAGAN-VC model, with the discriminator of StarGAN-VC2, namely WAGAN-VC (ProjectD). In the StarGAN-VC2 paper (Kaneko et al., 2019b), the discriminator was named as "Projection Discriminator", so that this chapter reuses this name. For this study, the results of the six data situations in Table 5.1 will be presented.

Ablation Study: The Effects of the Speaker Embedding Reconstruction Objective

The WAGAN-VC model uses a speaker embedding reconstruction objective, in order to enhance speaker similarity performance. In order to study the effects of this objective, this work compares the WAGAN-VC model with and without this objective. For this study, the results of the six data situations in Table 5.1 will be presented.

5.5 Results and Discussions

5.5.1 Exploring Training Data Situations

StarGAN-VC

Table 5.3 presents the results of the models under the six data situations. It is clear that StarGAN-VC suffered from a poor speaker similarity performance, which has also been proved in the StarGAN-VC2 paper (Kaneko et al., 2019b). As M was increasing, the

Table 5.3 The results of exploring *M* and *N*, where "all" denotes the means and standard deviations over data situations, ACC denotes speaker classification accuracy.

Model	Index	*M*	WER (%) ↓	ACC (%) ↑	MOSNet ↑
StarGAN-VC	1	10	16.5	78.8	2.85
	2	20	11.5	56.8	2.90
	3	40	11.8	62.4	2.79
	4	60	11.0	63.0	2.88
	5	90	11.3	70.1	2.85
	6	109	11.6	61.7	2.85
	all	-	12.2±1.9	65.4±7.11	2.85±0.03
StarGAN-VC2	1	10	23.3	89.7	2.86
	2	20	24.7	79.8	2.81
	3	40	26.0	86.0	2.88
	4	60	27.0	88.4	2.83
	5	90	35.3	73.1	2.83
	6	109	35.9	82.5	2.83
	all	-	28.7±5.0	83.2±5.6	2.84±0.02
WAGAN-VC	1	10	15.5	90.1	2.86
	2	20	15.3	89.4	2.86
	3	40	16.0	94.1	2.85
	4	60	14.1	87.4	2.89
	5	90	14.7	89.1	2.91
	6	109	14.2	88.7	2.93
	all	-	14.9±0.6	89.8±2.0	2.88±0.02

StarGAN-VC model experienced a degradation of the speaker classification accuracy from 78.8% to 61.7%. The degradation was probably due to the fact that the StarGAN-VC model used a speaker classifier. When increasing *M* and decreasing *N*, it would be more difficult for the speaker classifier to distinguish speakers.

The StarGAN-VC model performed good on WER results, with a mean WER of 12.2%, which might be because the generator is fully based on 2D-convolution layers. As discussed in the StarGAN-VC2 paper (Kaneko et al., 2019b), the 2D-convolution layers were more suitable for preserving the phonetic information of source speech features.

Overall, when increasing *M* and decreasing *N*, StarGAN-VC suffered from poor speaker similarity performance and robustness.

StarGAN-VC2

Comparing the StarGAN-VC2 model to the StarGAN-VC model, the StarGAN-VC2 improved the speaker similarity performance and robustness. As *M* was increasing and *N* was decreasing, StarGAN-VC2 experienced a degradation of the WER results from 23.3% to 35.9%. Also the standard deviation of its WER results was high, which was 5.0. This might be because of the information loss issue caused by the CIN layers. When the amount of data per speaker was decreasing, it is clear the information loss issue was getting worse.

The speaker accuracy of the StarGAN-VC2 model were better than the StarGAN-VC model, with a mean value of 83.2%. However, the mean value of the MOSNet results of the StarGAN-VC2 model (2.84) was slightly worse than the StarGAN-VC model (2.85).

WAGAN-VC

It is clear that WAGAN-VC obtained good performance on the EER and MOSNet metrics. Except the WER results of the WAGAN-VC model were worse than the StarGAN-VC model. The mean of WER results of WAGAN-VC is 14.9%, slightly worse than StarGAN-VC, which is 12.2%. But the standard deviation of WER results of WAGAN-VC is smaller than StarGAN-VC.

When increasing N and decreasing M, the WER results of the WAGAN-VC model showed a slight improvement from 15.5% to 14.2%. These WER results were worse than the StarGAN-VC model, but the standard deviation of the WAGAN-VC model (0.6) was smaller than the StarGAN-VC model (1.9).

The ACC results of the WAGAN-VC slightly decreased from 90.1% to 88.7%. From the means and the standard deviations, the WAGAN-VC model outperformed the other two models on the performance and the robustness of speaker similarity. The WAGAN-VC model obtained the highest mean of ACC results (89.8%) and the lowest standard deviation of ACC results (2.0).

The WAGAN-VC obtained the highest mean of the MOSNet results (2.88), outperforming the other two models. Overall, it can be observed that the WAGAN-VC model outperformed the StarGAN-VC2 model on VC performance and robustness. In addition, the WAGAN-VC model outperformed the StarGAN-VC on the performance and the robustness of speaker similarity and naturalness. Although the WAGAN-VC model performed worse than the StarGAN-VC model on intelligibility, the WAGAN-VC obtained better performance and robustness on speaker similarity and naturalness.

5.5.2 Exploring Low-Resource Data Situations

StarGAN-VC

From the results of the StarGAN-VC model, when decreasing N, the StarGAN-VC model experienced an obvious degradation of the ACC results, from 61.7% to 41.4%. These

Table 5.4 The results of low-resource data situations, where "all" denotes the means and standard deviations over data situations, ACC denotes speaker classification accuracy.

Model	Index	N	WER (%) $\downarrow$	ACC (%) $\uparrow$	MOSNet $\uparrow$
StarGAN-VC	1	35	11.6	61.7	2.84
	2	20	10.5	43.1	2.89
	3	10	9.6	51.4	2.86
	4	5	9.6	41.4	2.80
	all	-	10.3±0.8	49.4±8.0	2.84±0.03
StarGAN-VC2	1	35	35.9	82.5	2.83
	2	20	41.5	88.8	2.72
	3	10	76.2	76.0	2.72
	4	5	108.0	51.5	2.60
	all	-	65.4±29.0	74.7±14.1	2.71±0.08
WAGAN-VC	1	35	14.2	88.7	2.93
	2	20	14.3	93.1	2.88
	3	10	13.4	88.2	2.88
	4	5	25.5	95.1	2.81
	all	-	16.8±5.0	91.2±2.9	2.87±0.04

results show that the effects of the speaker classifier were weakened when N was decreasing. When training with fewer utterances, the StarGAN-VC model suffered from a poor speaker similarity performance.

The WER results of the StarGAN-VC model showed a stable performance, with a mean of 10.3% and a standard deviation of 0.8%. Considering both the intelligibility and the speaker similarity, when decreasing N, the StarGAN-VC model obtained good intelligibility, however, suffered from a poor speaker similarity performance.

StarGAN-VC2

The StarGAN-VC2 model suffered from an obvious degradation of the WER results. When decreasing N, the WER results increased from 35.9% to 108.0%. Especially, when the N was only 5, a 108.0% WER showed that the model could hardly preserve the phonetic information of source speech features.

Not only the intelligibility performance, decreasing N also impacted the ACC results and the MOSNet results of the StarGAN-VC2 model. The ACC results decreased from 82.5% to 51.5%, the MOSNet results decreased from 2.83 to 2.60. It can be concluded from these results that the performance of the StarGAN-VC2 model was dependent on N. As N was decreasing, the StarGAN-VC2 model suffered from performance degradation, which might be due to the information loss issue of the CIN layers, especially when data was insufficient.

WAGAN-VC

When decreasing N, the WER results of the WAGAN-VC model increased from 14.2% to 25.5%. Although the WAGAN-VC model suffered from an intelligibility degradation, the WAGAN-VC model obtained lower mean and smaller standard deviation of the WER results than the StarGAN-VC2 model. The standard deviation of the WER results of the WAGAN-VC model was 5.0, however, the StarGAN-VC2 model had a larger standard deviation, 29.0. The mean of the WER results of the WAGAN-VC (16.8%) was better than the StarGAN-VC2 model (65.4%). The WAGAN-VC model performed worse on WER results than the StarGAN-VC model.

The WAGAN-VC model outperformed the other two models on the ACC results and the MOSNet. The mean and the standard deviation of the ACC results of the WAGAN-VC model were 91.2% and 2.9. These results were better than the StarGAN-VC2 model (74.7% $\pm$ 14.1) and the StarGAN-VC model (49.4% $\pm$ 8). The mean of the MOSNet results of the WAGAN-VC was 2.87, while the StarGAN-VC2 model only obtained 2.71 and the StarGAN-VC model only obtained 2.84.

In low-resource data situations, the WAGAN-VC model clearly outperformed the StarGAN-VC2 model on the performance and the robustness of VC. Although the WAGAN-VC model performed worse than the StarGAN-VC model on intelligibility, it outperformed the StarGAN-VC model on speaker similarity and naturalness.

5.5.3 Ablation Study: The Effects of the WAdaIN Layer

As introduced before, the WAGAN-VC (AdaIN)-1 model only replaced the WAdaIN layer with the AdaIN layer, without changing the hyper-parameters. From the results in Table 5.5, the AdaIN layer caused an obvious degradation on all three metrics. This is probably because of the information loss issue of the AdaIN layer. Especially when M was 109, the WER was 101.0%, which means the model could hardly preserve the phonetic information of source speech features. When setting the hyper-parameters λ_{cyc} to 10 and λ_{id} to 5, the WAGAN-VC (AdaIN)-2 model obtained a better performance than the WAGAN-VC (AdaIN)-1 model.

Comparing the WAGAN-VC model with the WAGAN-VC (AdaIN)-2 model, although tuning hyper-parameters showed improvements on the WER results, however, the WAGAN-VC model still obtained better WER results and better ACC results.

Overall, the effects of the WAdaIN layers have been shown by comparing the WAdaIN layer with the AdaIN layer. The WAGAN-VC model showed a better intelligibility performance and a slightly better speaker similarity performance. Moreover, the WAdaIN layer showed better robustness of the intelligibility performance and the speaker similarity

Table 5.5 The results of the ablation study of the WAdaIN layer, where "all" denotes means and standard deviations over data situations, "ACC" denotes speaker classification accuracy.

Model	Index	M	WER (%) ↓	ACC (%) ↑	MOSNet ↑
WAGAN-VC (AdaIN)-1	1	10	56.4	94.2	2.75
	2	20	89.9	88.3	2.14
	3	40	94.2	83.5	2.08
	4	60	79.9	87.5	2.34
	5	90	94.3	83.4	2.48
	6	109	101.0	78.2	2.20
	all	-	85.9±14.6	85.8±4.9	2.33±0.22
WAGAN-VC (AdaIN)-2	1	10	30.8	91.8	2.89
	2	20	28.6	91.4	2.90
	3	40	39.1	84.5	2.88
	4	60	34.8	88.0	2.90
	5	90	26.5	80.2	2.90
	6	109	33.3	85.6	2.92
	all	-	32.1±4.1	86.9±4.0	2.89±0.01
WAGAN-VC	1	10	15.5	90.1	2.86
	2	20	15.3	89.4	2.86
	3	40	16.0	94.1	2.85
	4	60	14.1	87.4	2.89
	5	90	14.7	89.1	2.91
	6	109	14.2	88.7	2.93
	all	-	14.9±0.6	89.8±2.0	2.88±0.02

performance. These results show that the WadaIN layer performed better than the AdaIN layer. The information loss issue of the AdaIN layer was alleviated by the WadaIN layer.

5.5.4 Ablation Study: The Effects of the Speaker Encoder

From the results of the WAGAN-VC (1hot)-1 model in Table 5.6, when using speaker one-hot vectors to replace the speaker embeddings extracted from the speaker encoder, the WAGAN-VC (1hot)-1 model improved the ACC results. However, the WAGAN-VC (1hot)-1 model got worse WER results and worse MOSNet results. Especially when M was 109, the WER result degraded to 92.8% and the ACC result increased to 96.7%. This might indicate that the WAGAN-VC (1hot)-1 model suffered from an over-fitting issue. The reason might be the discriminator was over-fitting, so that it caused distortions in converted speech samples.

After changing the hyper-parameters λ_{cyc} to 10 and λ_{id} to 5, the WAGAN-VC (1hot)-2 model obtained better intelligibility than the WAGAN-VC (1hot)-1 model. As M was increasing, the WAGAN-VC (1hot)-2 model obtained better WER results but worse ACC results.

Table 5.6 The results of the ablation study of the effects of the speaker encoder module of the WAGAN-VC model, where "all" denotes means and standard deviations over data situations, "ACC" denotes speaker classification accuracy.

Model	Index	M	WER (%) ↓	ACC (%)↑	MOSNet ↑
WAGAN-VC (1hot)-1	1	10	20.0	92.4	2.83
	2	20	22.2	91.8	2.85
	3	40	24.2	92.1	2.84
	4	60	24.3	94.3	2.83
	5	90	21.3	90.3	2.85
	6	109	92.8	96.7	2.84
	all	-	34.1±26.2	93.3±1.6	2.84±0.008
WAGAN-VC (1hot)-2	1	10	15.5	90.6	2.83
	2	20	19.7	88.7	2.82
	3	40	18.3	92.7	2.88
	4	60	17.8	89.1	2.81
	5	90	11.2	75.2	2.85
	6	109	10.6	72.8	2.86
	all	-	15.5±3.4	84.8±7.8	2.84±0.02
WAGAN-VC	1	10	15.5	90.1	2.86
	2	20	15.3	89.4	2.86
	3	40	16.0	94.1	2.85
	4	60	14.1	87.4	2.89
	5	90	14.7	89.1	2.91
	6	109	14.2	88.7	2.93
	all	-	14.9±0.6	89.8±2.0	2.88±0.02

Comparing the averaged results of the WAGAN-VC (1hot)-2 model with the WAGAN-VC model, using the speaker one-hot vector obtained a similar intelligibility performance, a slightly worse speaker similarity performance. But the WAGAN-VC model obtained smaller standard deviations of the WER results and smaller standard deviations of the ACC results. Hence, compared with using the speaker one-hot vector, the speaker encoder improved the robustness of the performance.

5.5.5 Ablation Study: The Effects of the Discriminator

From the results in Table 5.7, the WAGAN-VC (ProjectD) model suffered from a poor robustness issue. As M was increasing, the WER results improved from 17.3% to 10.3%, however, the ACC results also decreased from 97.2% to 71.2%. The standard deviations of the WER results and the ACC results got increased. The WAGAN-VC (ProjectD) model obtained worse MOSNet results than the WAGAN-VC model. These results show that the

Table 5.7 The results of the ablation study of the effects of the discriminator, where "all" denotes means and standard deviations over data situations, "ACC" denotes speaker classification accuracy.

Model	Index	M	WER (%) $\downarrow$	ACC (%)$\uparrow$	MOSNet$\uparrow$
WAGAN-VC (ProjectD)	1	10	17.3	97.2	2.82
	2	20	15.4	87.4	2.81
	3	40	12.0	87.4	2.82
	4	60	11.3	81.4	2.83
	5	90	11.9	78.5	2.86
	6	109	10.3	71.2	2.84
	all	-	13.0±2.4	83.2±7.9	2.82±0.01
WAGAN-VC	1	10	15.5	90.1	2.86
	2	20	15.3	89.4	2.86
	3	40	16.0	94.1	2.85
	4	60	14.1	87.4	2.89
	5	90	14.7	89.1	2.91
	6	109	14.2	88.7	2.93
	all	-	14.9±0.6	89.8±2.0	2.88±0.02

discriminator of the WAGAN-VC model brought better speaker similarity and naturalness performance than the discriminator of the StarGAN-VC2 model. But it slightly hurt the intelligibility performance.

5.5.6 Ablation Study: The Effects of the Speaker Embedding Reconstruction Objective

Table 5.8 shows the results of the WAGAN-VC model and the WAGAN-VC (w.o. $\mathcal{L}_{gen}^{emb}$). The effect of the speaker embedding reconstruction objective was minor. This objective decreased the standard deviation of the WER results from 0.9 to 0.6 and increased the standard deviation of the ACC results from 1.3 to 2.0. Hence, from these results, the speaker embedding reconstruction objective improved the robustness of the intelligibility and slightly hurt the robustness of the speaker similarity.

5.6 Conclusion

This chapter evaluated the performance and the robustness of the models on different data situations. In order to explore different data situations, this chapter focused on controlling two factors: the number of speakers and the number of utterances per speaker. In order to

Table 5.8 The results of the ablation study of the effects of the speaker embedding reconstruction objective. , where "WER", "ACC" and "MOSNet" denote the Word Error Rate, speaker classification accuracy and the MOSNet score, respectively. The "all" denotes means and standard deviations over the six data situations. "WAGAN-VC (w.o. $\mathcal{L}_{gen}^{emb}$)" denotes the variant model of the WAGAN-VC model that does not use the speaker embedding reconstruction objective.

Model	Index	M	WER (%)	ACC (%)	MOSNet
WAGAN-VC (w.o. $\mathcal{L}_{gen}^{emb}$)	1	10	16.2	92.1	2.88
	2	20	16.1	89.3	2.87
	3	40	14.7	89.6	2.85
	4	60	14.1	89.2	2.87
	5	90	13.9	87.8	2.90
	6	109	14.9	90.6	2.92
	all	-	15.0±0.9	89.7±1.3	2.88±0.02
WAGAN-VC	1	10	15.5	90.1	2.86
	2	20	15.3	89.4	2.86
	3	40	16.0	94.1	2.85
	4	60	14.1	87.4	2.89
	5	90	14.7	89.1	2.91
	6	109	14.2	88.7	2.93
	all	-	14.9±0.6	89.8±2.0	2.88±0.02

improve the model performance and robustness, t his chapter further proposed a new model, WAGAN-VC. The WAGAN-VC model introduced a model space speaker adaptation method, WAdaIN.

The experiments were conducted on the VCTK dataset. The results show that, when changing the training data situations, the WAGAN-VC obtained better speaker similarity and naturalness performance and better robustness than the StarGAN-VC model and the StarGAN-VC2 model. Furthermore, in the experiment of exploring low-resource data situations, the WAGAN-VC model obtained better speaker similarity and naturalness performance and better robustness than the two baseline models. Four ablation studies were conducted for analysing the effects of the WAdaIN layer, the discriminator, the speaker encoder and the speaker embedding reconstruction objective, respectively. The results show that the WAdaIN layer performed better than the AdaIN layer on speaker similarity and intelligibility. It alleviated the information loss issue of the AdaIN layer. The new discriminator architecture in the WAGAN-VC model obtained better speaker similarity and naturalness performance than the discriminator used in the StarGAN-VC2 model. The use of the speaker encoder improved the robustness of the WAGAN-VC model.

Chapter 6

Self-Supervised Learning Model for VC

6.1 Introduction

Encoder-decoder based VC models (Liu et al., 2018, 2020b, 2021, Huang et al., 2020a,c, Zhang et al., 2020) have gained successes in the recent VC challenges. In the VCC 2018 challenge (Lorenzo-Trueba et al., 2018), an encoder-decoder based model (Liu et al., 2018) became the best performing model. In the recent VCC 2020 challenge (Yi et al., 2020), the encoder-decoder based VC models (Liu et al., 2020b, Huang et al., 2020a,c, Zhang et al., 2020, Zhang, 2020b) have been widely used and they have obtained highly competitive performance. Among 33 participants, 24 participants built their models upon the encoder-decoder framework (Yi et al., 2020). The best performing model, Taco-AR (Liu et al., 2020b), achieved a level of naturalness performance slightly worse than real human voices and reached the same level of speaker similarity performance of real human voices in the subjective evaluations of the intra-lingual VC task (Yi et al., 2020).

The encoder-decoder framework has been introduced in Section 2.4. In this framework, one or more encoders are employed to extract representations from a speech signal. A commonly used implementation of linguistic encoder is to employ an ASR model. However, the training of ASR models usually requires large datasets with text transcriptions (Deng and Li, 2013), which could be a limitation in practical applications.

In order to release the dependence on ASR, one solution is to use an unsupervised phonetic representation learning model as the linguistic encoder. SSL models (Schneider et al., 2019, Baevski et al., 2020, 2019, Chung and Glass, 2020, Chung et al., 2020, Liu et al., 2020a, Hsu et al., 2021) aim to learn representations from speech data, so that these representations can be useful for down-stream speech tasks. Recently, S3PRL-VC (Huang et al., 2022b) applied SSL models in VC. The objective of S3PRL-VC was to replace the ASR dependent linguistic encoder in a VC system with a SSL model. They also aimed to compare the VC performance of different SSL models. The VC performance of SSL based VC models were evaluated on the VCC 2020 challenge dataset (Yi et al., 2020). It was concluded that among SSL models, the VQ-Wav2vec model (Baevski et al., 2019) obtained better performance than the other SSL models. They also found that SSL based VC models can not reach the same level of performance of the state-of-the-art models such as Taco-AR. From (Huang et al., 2022b), it is also noticeable that SSL based VC models outperform auto-encoder based models and GAN based models.

As discussed in Section 2.4, there existed a drawback in the experiment setup of the S3PRL-VC work. In their experiments, the decoder of VC models was trained on the VCC 2020 dataset, which only has 2 hours, as shown in Section 2.6. Compared to the 130-hours training data of Taco-AR, the VCC 2020 dataset is too small. Hence, this performance comparison was not fair. It is still not clear whether text-free SSL based VC models can reach the same level of performance as the state-of-the-art text-dependent VC models (Liu et al., 2020b, 2021, 2018).

The objective of this chapter is to apply a SSL model for VC and compare its performance with the state-of-the-art text-dependent VC models. Specifically, the VQ-Wav2vec model is selected as the representative of SSL models. The ASR-BNE model (Liu et al., 2021) is selected as the state-of-the-art linguistic encoder. Motivations of choosing these two models will be explained later. The encoder-decoder framework is utilised and implemented for experiments. In the framework, two implementations of the linguistic encoder are to be compared. Besides, two implementations of the decoder are to be compared. In experiments, three VC tasks are established, including a many-to-many VC task, an intra-lingual one-shot VC task and a cross-lingual one-shot VC task. Furthermore, a speech re-synthesis task will be used for analysis.

6.1.1 Chapter Outline

The remainder of this chapter is organised as follows:

- Section 6.2 introduces the implementations of the encoder-decoder framework for this work.
- Section 6.3 introduces the experiment setup.
- Section 6.4 presents and discusses the results of the experiments and the analysis.
- Section 6.5 is the conclusion.

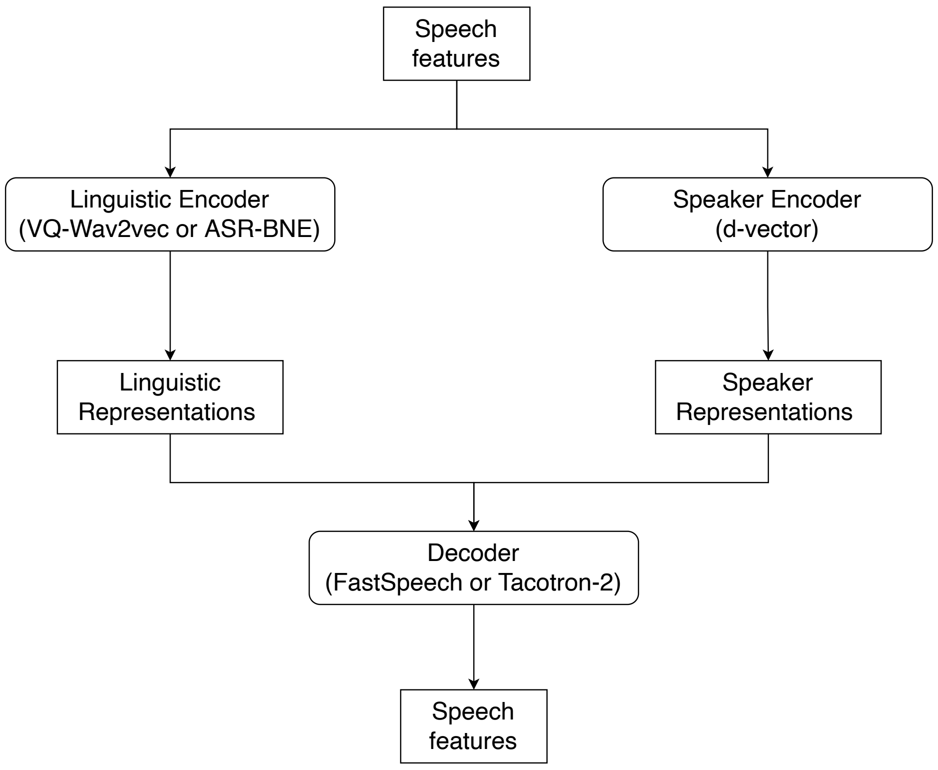
Speech features
Linguistic Encoder (VQ-Wav2vec or ASR-BNE)
Speaker Encoder (d-vector)
Linguistic Representations
Speaker Representations
Decoder (FastSpeech or Tacotron-2)
Speech features

Fig. 6.1 An illustration of the encoder-decoder framework, which is based on the framework of S3PRL-VC (Huang et al., 2022b).

6.2 Implementations of the Encoder-Decoder Framework

This section introduces the implementations of the encoder-decoder framework, which will be used for the experiments. This framework is based on the framework of S3PRL-VC (Huang et al., 2022b). Because the objective of S3PRL-VC was similar to this work. In S3PRL-VC, the objective was to compare the VC performance of SSL models. The objective of this work is to further compare the VC performance of SSL models with the state-of-the-art supervised linguistic encoders. Hence, this work reuses the framework of S3PRL-VC and further extends it with more implementations on the decoder.

The encoder-decoder framework is illustrated in Figure 6.1. The framework is composed of a linguistic encoder, a speaker encoder and a decoder. The implementations for each are presented in Figure 6.1. For the linguistic encoder, two implementations are selected: the VQ-Wav2vec model (Baevski et al., 2019) and the ASR-BNE model (Liu et al., 2021). For the speaker encoder, following the S3PRL-VC framework (Huang et al., 2022b), d-vector (Wan et al., 2018) is chosen. For the decoder, two implementations are selected: the Tacotron-2 model (Shen et al., 2018) and the FastSpeech model (Ren et al., 2019). The following mainly introduces the motivations for selecting these implementations.

6.2.1 Linguistic Encoder Implementations

Motivations for Selecting the VQ-Wav2vec Model

The motivations of selecting the VQ-Wav2vec model as the representative of SSL models are from two observations from the previous works (Yang et al., 2021, Huang et al., 2022b). One is from the previous SUPERB benchmark (Yang et al., 2021), which evaluated the quality of the representations of SSL models. The other one is from S3PRL-VC.

When applied as a linguistic encoder in VC, a SSL model is supposed to be able to learn disentanglement between the speaker-invariant information and the speaker-dependent information of speech data. In other words, SSL models are expected to remove speaker information. As introduced in Section 3.5, the SUPERB benchmark (Yang et al., 2021) evaluated the quality of representations through various down-stream tasks, including a phone classification task and a speaker classification task. The results have been presented in Section 3.5 in Table 2.2. As discussed in Section 3.5, among various SSL models, the speaker accuracy of the VQ-Wav2vec was the lowest among SSL models.

As discussed in Section 2.4, it was concluded in S3PRL-VC (Huang et al., 2022b) that the VQ-Wav2vec performed better than the other SSL models in VC. The reason was also discussed in (Huang et al., 2022b), the VQ-Wav2vec performed better on learning the disentanglement between the speaker-dependent information and the speaker-invariant information.

In summary, this work selects the VQ-Wav2vec model because it performed good on learning disentanglement (Yang et al., 2021) and also it performed good on VC (Huang et al., 2022b).

Motivations for Selecting the ASR-BNE

As introduced in Section 2.4 that the three types of commonly used supervised linguistic encoders are ASR models (Zhang et al., 2020, Huang et al., 2020a), PPGs extractor models (Liu et al., 2020b, 2018) and ASR-BNE models (Liu et al., 2021). The differences between them have been discussed in Section 2.4. Both ASR models and PPGs extractor models might cause a mis-pronunciation issue of converted speech signals. The ASR-BNE model (Liu et al., 2021) was proposed for solving this issue. They conducted a comparison between the ASR-BNE model and a PPGs extractor model from (Liu et al., 2018). They conducted MOS tests for naturalness and speaker similarity evaluations. From the results in (Liu et al., 2021), in the naturalness MOS test, the PPGs extractor model (Liu et al., 2018) could only obtain 2.48, while the ASR-BNE reached 3.75. The speaker similarity MOS of the PPGs extractor model was only 2.78, while the ASR-BNE reached a speaker similarity MOS of

3.82. From these results, the ASR-BNE model can be considered as a representative of the supervised linguistic encoders.

6.2.2 Speaker Encoder

As introduced in Section 6.2 that the two commonly used speaker encoder models are the x-vector model (Snyder et al., 2018) and the d-vector model (Wan et al., 2018). However, there has not been a specific comparison for speaker encoder models for VC. So, this section follows (Liu et al., 2021, Huang et al., 2022b, Liu et al., 2020b) to use the d-vector model as the speaker encoder model.

6.2.3 Decoder

This work uses two implementations for the decoder, the FastSpeech model (Ren et al., 2019) and the Tacotron-2 model (Shen et al., 2018). The FastSpeech model is a non-autoregressive model and the Tacotron-2 model is an autoregressive model. The objective of selecting these two models is to explore whether the non-autoregressive decoder can reach the same level of the autoregressive decoder model.

Both of these two models are TTS models. As discussed in Section 2.4, in recent years, the non-autoregressive TTS models (Ren et al., 2019, 2020) have become a popular research direction. The FastSpeech model (Ren et al., 2019) and the FastSpeech2 model (Ren et al., 2020) are two examples of the non-autoregressive TTS models. The objective of these non-autoregressive TTS models is to speed up the decoding speed of the models. However, from the results of their papers (Ren et al., 2019, 2020), the performance of the non-autoregressive TTS models were still behind the autoregressive models (Shen et al., 2018, Li et al., 2019).

In the context of VC, the most commonly used decoder models are autoregressive models (Liu et al., 2020c, Shen et al., 2018). As introduced in Section 2.4 and in Table 2.1, Tacotron-2 (Shen et al., 2018) is one of the commonly used decoder models. It has obtained good performance in many works (Liu et al., 2021, 2020b, Huang et al., 2022b).

The explorations of applying non-autoregressive decoder models to VC models are still rare. From the summary paper of the VCC 2020 challenge (Yi et al., 2020), a small number of participants (Barbany and Cernak, 2020, Huang et al., 2020c) built their models based on non-autoregressive decoder models and most of them performed poorly. FastSpeech-VC (Zhao et al., 2021) is an example of applying the FastSpeech model (Ren et al., 2019) to VC. They employed a PPGs extractor model as the linguistic encoder and the FastSpeech model as the decoder. However, there is still little comparison analysis between the FastSpeech model and the Tacotron-2 model as the decoder for VC.

6.3 Experiment Setup

The main objective of the experiments of this work is to compare the VC performance of the VQ-Wav2vec model and the ASR-BNE model. Through the experiments, this work also aims to compare the VC performance of the FastSpeech model and the Tacotron-2 model.

This work conducts the experiments on the LibriTTS dataset (Zen et al., 2019) and the VCTK dataset (Yamagishi et al., 2019). As introduced in Section 2.6 that the LibriTTS dataset are split into seven subsets, according to data quality. In order to maintain the quality of training data, this work only uses the train-clean-100 subset and the train-clean-360 subset as the training data. In total, these two subsets contains 1151 speakers and around 240 hours data.

In the experiments, this work sets up three VC tasks: many-to-many VC, intra-lingual one-shot VC and cross-lingual one-shot VC. As introduced in Section 2.1, there are many types of VC tasks. They can be distinguished from whether testing speakers appear in training data and from whether testing speakers speak a different language from a source speaker.

The first task is to conduct VC for test speakers within training data The second task is to test with speakers that do not appear in training data and they speak the same language as in training data. The third task is to test with speakers that do not appear in training data and they speak other languages.

Before conducting the three VC tasks, this work begins with an analysis of the disentanglement learning performance of the representations from the VQ-Wav2vec model and the ASR-BNE model. This work evaluates representations through a phone classification task and two speaker classification tasks. The objective of this analysis is to show how well the representations can preserve phonetic information and remove speaker information. The results might be helpful for understanding the results of the VC tasks. After the three VC tasks, this work further conducts an analysis through a speech re-synthesis task. The VC models are to re-syntheses an input speech signal. Hence, the quality of the re-synthesised speech signals can be helpful to understand the effects of the two linguistic encoders and the two decoders.

6.3.1 Experiment Setup

Analysing the Disentanglement Learning performance

As introduced in Section 3.6 that one type of evaluation metrics for evaluating the phonetic representations is through down-stream classification tasks. This work follows this way and applies the S3PRL toolkit (Yang et al., 2021) for implementing the classification tasks.

This work sets up three classification tasks on the LibriSpeech dataset (Panayotov et al., 2015). The three classification tasks include a phone classification task, a frame-level speaker classification task and an utterance-level speaker classification task. The VQ-Wav2vec model and the ASR-BNE model are used as a feature extractor model, respectively. Representations of speech samples in the LibriSpeech dataset are extracted.

For the phone classification task, a simple linear layer classification model is used. Representations are the inputs to the classification model. The classification model is trained with phoneme transcriptions provided by the S3PRL toolkit. The classification model takes in a sequence of representations and predicts a phoneme label for each frame. Then a frame-level accuracy can be reported as a result.

For the frame-level speaker classification task, a simple linear layer classification model is used. Representations are the inputs to the model. The classification model is trained with speaker identities provided by the LibriSpeech dataset. The classification model takes in a sequence of representations and predicts a speaker label for each frame. Then a frame-level accuracy can be reported as a result.

For the utterance-level speaker classification task, a simple linear layer classification model is used. Before feeding to the classification model, representations are averaged across time. The averaged representations are fed to the classification model. The classification model predicts a speaker identity for the whole utterance. Then a utterance level accuracy can be reported as a result.

Many-to-Many VC Task

The objective of this task is to evaluate test speakers within training data. The VC models are trained and then tested on the LibriTTS dataset and the VCTK dataset, respectively. For testing, 20 speakers are randomly chosen from the LibriTTS dataset and the VCTK dataset, respectively.

Then this work sets up a test sample pair list for each dataset. The test sample pair list is composed of pairs of two samples, a sample from a source speaker and a sample from a target speaker. To produce test sample pair lists, each test speaker is paired with the other 19 test speakers. Hence, there are 380 (380=19$\times$20) source-to-target mappings. For each source-to-target mapping, 2 test samples are randomly selected from the source speaker, 2 samples are randomly selected from the target speaker. Hence, each mapping contains 4 (4=2$\times$2) test sample pairs. In total, there are 1520 ($1520 = 380 \times 4$) test sample pairs, so that 1520 converted samples will be evaluated.

For this VC task, this work evaluates converted speech samples from three aspects: intelligibility, speaker similarity and naturalness. This work uses three objective and two subjective evaluation metrics. The three objective evaluation metrics are: the intelligibility evaluation with an ASR model, the speaker similarity evaluation with a speaker recognition model and the naturalness evaluation with a MOSNet model. The two subjective evaluation metrics are: the MOS tests for naturalness and speaker similarity respectively.

Intra-lingual One-shot VC Task

The objective of this task is to evaluate models with test speakers outside of training data. The VC models are trained on the LibriTTS dataset and the VCTK dataset, respectively. The models are tested on the VCC 2020 dataset. As introduced in Section 2.6 that the VCC 2020 dataset contains four English source speakers and four English target speakers. This task follows the intra-lingual task setup in the VCC 2020 challenge (Yi et al., 2020) and uses these eight English speakers as test speakers. Hence, there are totally 16 ($16=4\times4$) source-to-target mappings. To produce the test sample pair list, for one source-to-target mapping, this work uses the whole 25 test samples from a source speaker and randomly selects 1 test sample from a target speaker. Hence, each source-to-target mapping has 25 ($25=25\times1$) test sample pairs. In total, there are 400 ($400=16\times25$) converted samples to be evaluated.

Similar to the many-to-many VC task, this task uses the same three objective metrics and the same two subjective metrics.

Cross-lingual One-shot VC Task

The objective of this task is to evaluate the models with target speakers outside of training data and speak other languages. The VC models are trained on the LibriTTS dataset and the VCTK dataset, respectively. Then the models are tested on the VCC2020 dataset. This task follows the cross-lingual task setup in the VCC 2020 challenge (Yi et al., 2020). The four English source speakers are used as source speakers. The six target speakers from the other three languages are used as testing target speakers. The six target speakers include two speakers from Mandarin, two speakers from German and two speakers from French.

Hence, there are four source speakers and six target speakers, leading to totally 24 ($24=4\times6$) source-to-target mappings. To produce the test sample pair list, for one source-to-target mapping, this work uses the whole 25 test samples from a source speaker and randomly selects 1 test sample from a target speaker. Hence, for each source-to-target mapping, there are 25 ($25=25\times1$) test sample pairs. In total, there are 600 ($600=24\times25$) converted samples to be evaluated. The evaluation metrics of this task is the same as the three objective metrics and the two subjective metrics used in the many-to-many VC task.

Analysis: Speech Re-synthesis Task

The objective of this speech re-synthesis task is to analyse, given an input speech sample, whether the VC models can perfectly reconstruct the information of an input speech sample. For example, if the model can perfectly reconstruct a speech sample with high intelligibility, it might show the linguistic encoder is performing good on preserving phonetic information. Another motivation is to evaluate whether the VC model can perfectly reconstruct the prosodic information of an input speech sample. If the model can perfectly reconstruct prosodic information, it might show that the decoder is able to control the prosodic information, without an explicit prosodic encoder applied in the framework.

In this task, the models are trained and tested on the LibriTTS dataset and the VCTK dataset, respectively. The 20 test speakers used in the many-to-many task will be reused in this task. For each test speaker, 10 samples are randomly selected as the input speech samples. In total, 200 ($200=10\times20$) re-synthesised samples will be evaluated.

To evaluate the re-synthesised samples, this task uses six objective evaluation metrics. The six objective metrics includes the intelligibility evaluation with an ASR model, the speaker similarity evaluation with a speaker recognition model, the naturalness evaluation with a MOSNet model, the MCD, the F0-CORR and the F0-RMSE. The objective of applying the MCD is to evaluate the overall performance of the model. The F0-RMSE and the F0-CORR evaluate the distances between the F0s of a reconstructed speech sample and the input speech sample.

6.3.2 Evaluation Metrics

Objective Evaluation

As introduced above, this work uses six types of objective evaluation metrics for various aspects of a speech sample.

For naturalness evaluations, this work uses the objective metric introduced in Section 2.6. A trained MOSNet model (Lo et al., 2019) is used to predict a MOS score from a speech sample. This work uses the speechmetrics (Liutkus et al., 2019) toolkit for the MOSNet implementation. This work uses the inference recipe from the speechmetrics toolkit. For each speech sample, the MOSNet model gives out a score, then the scores are averaged over the whole test set, the averaged score is reported.

For intelligibility evaluations, a trained ASR model (Dong et al., 2018) is used. For this evaluation, a ground-truth transcription is needed for a converted speech sample. The ASR model transcribes converted samples to texts. Then, a WER can be computed and reported.

This work uses a trained ASR model (Dong et al., 2018) provided by the SpeechBrain (Ravanelli et al., 2021) toolkit.

For the speaker similarity, a trained speaker recognition model, ECAPA-TDNN (Desplanques et al., 2020), is used to conduct a speaker verification task. The ECAPA-TDNN model is trained on a mix of the VoxCeleb 1 and VoxCeleb 2 datasets (Nagrani et al., 2017, 2020). This work produces a speaker verification pair list, including a number of positive sample pairs and negative sample pairs. The positive pair is composed of a converted speech and a real speech from the target speaker and the negative pair is composed of a converted speech and a real speech from another random selected speaker. The EER of the speaker verification task is reported. This work uses the speaker verification recipe in the SpeechBrain toolkit (Ravanelli et al., 2021). The details of the MCD, the F0-RMSE and the F0-CORR have been introduced in Section 2.6.

Subjective Evaluation

This work conducts subjective evaluations for evaluating the naturalness and the speaker similarity of converted speech samples. The MOS tests (Chu and Peng, 2001) are to be used as the subjective metric for both the naturalness evaluation and the speaker similarity evaluation.

As introduced in Section 2.6 that MOS evaluations have been widely used in many VC works (Yi et al., 2020, Liu et al., 2021, Lorenzo-Trueba et al., 2018, Huang et al., 2022b). In this work, MOS evaluations are conducted through on-line human listening tests. The Amazon MTurk platform (Amazon, 2005) is used for evaluations.

In the human listening tests, the whole test is composed of a number of sessions. Each session contains one or two speech samples, a question and five answers. A human evaluator needs to firstly listen to the samples then answer the question by selecting one of the provided answers.

In the naturalness evaluation, one session contains one speech sample. The question is "How natural is this recording?". The five answers are "5: Excellent-Completely natural speech", "4: Good- Mostly natural speech", "3: Fair- Equally natural and unnatural speech", '2: Poor- Mostly unnatural speech' and "1: Bad- Completely unnatural speech".

In the speaker similarity evaluation, one session contains two samples, one test sample and one reference sample. The test sample is the sample to be evaluated and the reference sample is a real sample from the target speaker of the test sample. The question is "How similar to the reference sample is this recording?". The five answers are "5: Excellent-Completely similar speech", "4: Good- Mostly similar speech", "3: Fair- Equally similar and dissimilar speech", ’2: Poor- Mostly dissimilar speech’ and "1: Bad- Completely dissimilar speech".

In order to reduce variances of results, this work detects outlier human evaluators. This work puts real speech samples into the test speech sample pool. Because the real speech samples are supposed to obtain high MOS results. This work detects outliers by checking the scores of real samples given by each human evaluator. If the mean score of real samples of one evaluator is less than "3" then this evaluator is detected as an outlier. His/her scores will be eliminated in the results collection.

In the many-to-many VC task, this work randomly selects 100 samples from the converted samples of each model. In total, there are 400 testing samples. In addition, 400 real speech samples are used in the evaluations, the real samples are randomly selected from the datasets. Each sample is evaluated by two evaluators, so that there are 800 scores obtained for the testing samples in the evaluation. On the LibriTTS dataset, there are 216 human evaluators involved in the evaluation. In these 216 evaluators, 15 of them are detected as outliers. After removing their scores, there exist 770 valid scores of the testing samples. On the VCTK dataset, there are 131 human evaluators involved. In these 131 human evaluators, 17 of them are detected as outliers. After removing their scores, there exist 694 valid scores of the testing samples.

In the intra-lingual one-shot VC task, this work randomly selects 160 converted samples of each model. In total there are 640 testing samples. In addition, 640 real speech samples are used in the evaluations, the real samples are randomly selected from the VCC 2020 dataset. Each sample is evaluated by two evaluators, so that there are 1280 scores obtained for the testing samples in the evaluation. On the LibriTTS dataset, there are 210 human evaluators involved in the evaluation. In these 210 evaluators, 18 of them are detected as outliers. After removing their scores, there exist 1238 valid scores of the testing samples. On the VCTK dataset, there are 166 evaluators involved. In these 166 evaluators, 11 of them are detected as outliers. After removing their scores, there exist 1235 valid scores of the testing samples.

In the cross-lingual one-shot VC task, this work randomly selects 240 converted samples of each model. In total there are 960 testing samples. In addition, 960 real speech samples are used in the evaluations, the real samples are randomly selected from the VCC 2020 dataset. Each sample is evaluated by two evaluators, so that there are 1920 scores obtained for the testing samples in the evaluation. On the LibriTTS dataset, there are 237 human evaluators involved in the evaluation. In these 237 evaluators, 32 of them are detected as outliers. After removing their scores, there exist 1704 valid scores of the testing samples. On the VCTK dataset, there are 232 evaluators involved. In these 232 evaluators, 30 of them are detected as outliers. After removing their scores, there exist 1734 valid scores of the testing samples.

6.3.3 Implementation Details

This work uses the official source code implementations of the ASR-BNE model (Liu et al., 2021). This work uses the model weights of the ASR-BNE model provided by the official implementations. This work also uses the d-vector implementation in (Liu et al., 2021). This work uses the model weights of the d-vector model provided by (Liu et al., 2021). The d-vector was trained on a mixed dataset, which was a mix of the VoxCeleb 1 (Nagrani et al., 2017) and the VoxCeleb 2 (Nagrani et al., 2020) and the LibriSpeech dataset (Panayotov et al., 2015). The mixed dataset contains more than 8000 speakers.

This work uses the official source code implementations of the VQ-Wav2vec model (Baevski et al., 2019). This work uses the model weights of the VQ-Wav2vec model provided by the official implementations. This work uses the implementations of the Tacotron-2 model (Liu et al., 2020b) from the source code implementations of S3PRL-VC (Huang et al., 2022b). This work uses the implementations of the FastSpeech model (Ren et al., 2019) in the ESPNet toolkit (Hayashi et al., 2020).

This work uses the HifiGAN vocoder (Kong et al., 2020) implementations provided by (Liu et al., 2021). As discussed in Section 2.2 that the HifiGAN vocoder can produce good quality speech and decode at a high speed.

This work extracts 80 dimensional log-mel-spectrograms as speech features, with a window size of 40 ms and a stride size of 10 ms. The training of the models are implemented using the PyTorch toolkit (Paszke et al., 2019). In the training of the models, the Adam optimiser (Kingma and Ba, 2015) is used with a learning rate of 1e-3. The batch size of the training is 20. The speech utterances are padded to 1000 frames, which stands for 10 seconds. The training of the models takes 200 epochs.

6.4 Results and Discussions

In this section, in order to present results with better readability, VC systems with different linguistic encoder and decoder are denoted as in Table 6.1. Please note that VQ-Wav2vec and ASR-BNE are denoted by VQW2V and BNE, respectively. FastSpeech and Tacotron-2 are denoted by FS and T2, respectively. The following results presentations and discussions will be based on combination indices and these abbreviations. From this table, it is easy to compare VQW2V and BNE through comparing the results of Sys-1 and Sy-3, as well as comparing Sys-2 and Sys-4. Comparing FS and T2 can be conducted through comparing results of Sys-1 and Sys-2, also Sys-3 and Sys-4.

Table 6.1 Combinations with abbreviations, listed with indices. "VQW2V" and "BNE" denote VQ-Wav2vec and ASR-BNe, respectively. "FS" and "T2" denote FastSpeech and Tacotron-2, respectively.

System	Linguistic encoder	Decoder
Sys-1	VQW2V	T2
Sys-2	VQW2V	Fs
Sys-3	BNE	T2
Sys-4	BNE	FS

Table 6.2 The results of the quality analysis of the representations, the accuracies of the classification tasks are presented, where "Phone" denotes phone classification accuracy, "Speaker Frame" denotes frame-level speaker classification accuracy and "Speaker Utterance" denotes utterance level speaker classification accuracy.

Features	Phone (%) $\uparrow$	Speaker Frame (%) $\downarrow$	Speaker Utterance (%) $\downarrow$
VQW2V	63.30	24.76	84.00
BNE	85.70	0.02	39.00

6.4.1 Analysing the Disentanglement Learning performance

Table 6.2 presents the accuracies of the phone classification and and the two speaker classification tasks. Generally speaking, BNE obtained a better phone classification accuracy and lower speaker classification accuracies. BNE obtained a 85.70% phone classification accuracy and VQW2V obtained a 63.30% accuracy. This might because that BNE is a supervised model, while VQW2V is an unsupervised model. These results show that BNE performed better on preserving phonetic information.

BNE obtained a 0.02% accuracy for the frame-level speaker classification task and a 39.00% accuracy for the utterance-level speaker classification task. VQW2V performed worse than BNE on removing speaker information. It obtained a 24.76% accuracy for the frame-level speaker classification task and a 84.00% accuracy for the utterance level speaker classification task. These results show that BNE performed better than VQW2V on disentanglement learning.

6.4.2 Many-to-Many VC Task

For the many-to-many VC task, the results on the VCTK dataset and the LibriTTS are presented in Table 6.3.

Table 6.3 Many-to-many VC results, where "System" denotes the systems in Table 6.1, "WER" denotes word error rate (%), "EER" denotes equal error rate (%), "MOSN" denotes MOSNet, "SS-MOS" denotes speaker similarity MOS. The means and the 95% confidence intervals of the MOS and the SS-MOS results are presented. "↑" denotes the higher the better. "↓" denotes the lower the better. Note that models trained on VCTK and LibriTTS are tested on different testing data, so their results are not comparable.

Dataset	System	WER ↓	EER ↓	MOSN ↑	MOS ↑	SS-MOS ↑
VCTK	Sys-1	11.6	2.2	3.95	3.84 ± 0.10	3.81 ± 0.13
	Sys-2	10.3	4.7	3.91	3.78 ± 0.11	3.80 ± 0.11
	Sys-3	8.8	2.2	3.81	3.77 ± 0.11	3.81 ± 0.12
	Sys-4	9.1	3.4	3.65	3.54 ± 0.07	3.47 ± 0.09
LibriTTS	Sys-1	3.7	11.4	3.34	3.81 ± 0.09	3.82 ± 0.10
	Sys-2	3.7	14.1	3.27	3.75 ± 0.10	3.82 ± 0.11
	Sys-3	3.6	10.9	3.16	3.70 ± 0.17	3.74 ± 0.11
	Sys-4	7.7	15.8	3.23	3.62 ± 0.12	3.69 ± 0.12

Intelligibility

Sys-3 obtained the best WER results on VCTK and LibriTTS, which are 8.8% and 3.6%, respectively. It means that on the many-to-many task, in terms of intelligibility, Sys-3 is the best-performing system. On VCTK, Sys-3 obtained a WER of 8.8%, while Sys-1 obtained a worse WER of 11.6%. A similar trend happened on Sys-2 and Sys-4, where Sys-4 (9.1%) outperformed Sys-2 (10.3%) on WER results. Comparing Sys-3 and Sys-1, as well as Sys-2 and Sys-4, on VCTK, it is clear that BNE outperformed VQW2V on intelligibility.

On LibriTTS, Sys-1 obtained a slightly worse WER (3.7%) than Sys-3 (3.6%), while Sys-4 obtained a worse WER (7.7%) than Sys-2 (3.7%). These results show that on LibriTTS when the decoder is TS, BNE slightly shows an advantage on intelligibility. However, when the decoder is FS, BNE is worse than VQW2V.

Overall, in terms of intelligibility performance, in most cases of two datasets, BNE showed an advantage over VQW2V. VQW2V only outperformed BNE in one case.

Speaker Similarity

In terms of speaker similarity, Sys-1 and Sys-3 achieved similar performance on VCTK. These two systems got the same EER result (2.2%). as well as the same SS-MOS results (3.81). Sys-4 obtained a better EER (3.4%) than Sys-2 (4.7%), however, it obtained a worse SS-MOS (3.47) than Sys-2 (3.80). From the divergent EER results and SS-MOS results of Sys-2 and Sys-4, it can be found that there exists a mismatch between the objective EER metric and the subjective SS-MOS metric.

On LibriTTS, Sys-3 got an EER of 10.9%, which is slightly better than Sys-1 (11.4%). SS-MOS results on LibriTTS show a slightly different trend, Sys-1 (3.82) obtained better results than Sys-3 (3.74). Sys-2 obtained a better EER (14.1%) than Sys-4 (15.1%), it also got a better SS-MOS (3.82) than Sys-4 (3.69). Overall, it is not easy to draw a conclusion from these results, due to divergent results of VQW2V and BNE in different cases. This divergence is probably due to the mismatch of EER and SS-MOS, as well as using different decoders.

Naturalness

On VCTK, in terms of naturalness, MOSN results and MOS results showed that Sys-1 outperformed Sys-3. Sys-1 obtained a MOSN of 3.95 and a MOS of 3.84, while Sys-3 obtained a worse MOSN result of 3.81 and a worse MOS result of 3.77. Similarly, Sys-2 outperformed Sys-4. Sys-2 obtained a MOSN of 3.91 a MOS of 3.78, while Sys-4 obtained a MOSN of 3.65 and a MOS of 3.54.

MOSN results and MOS results on LibriTTS also show that Sys-1 outperformed Sys-3 and Sys-2 outperformed Sys-4. Sys-1 obtained a MOSN of 3.34 and a MOS of 3.81, while Sys-3 obtained a MOSN of 3.16 and a MOS of 3.70. Similarly, Sys-2 obtained a MOSN of 3.27 and a MOS of 3.75, while Sys-4 obtained a MOSN of 3.23 and a MOS of 3.62.

Overall, in this task on both datasets, it is clear to draw a conclusion that VQW2V outperformed BNE on naturalness performance.

Comparing FS and T2

Comparing results of Sys-1 and Sys-2, as well as Sys-3 and Sys-4, on both datasets, it can be found that T2 outperformed FS in most cases. On VCTK, Sys-1 obtained an EER of 2.2% and a SS-MOS of 3.81, while Sys-3 obtained a worse EER of 4.7 and a slightly worse SS-MOS of 3.80. Similarly, Sys-3 obtained a better EER and a better SS-MOS than Sys-4. On LibriTTS, it is clear that Sys-1 achieved better speaker similarity performance than Sys-2, as well as that Sys-3 outperformed Sys-4. From these results, it can be concluded that, in this task, T2 outperformed FS on speaker similarity.

In terms of naturalness, it also can be found that Sys-1 outperformed Sys-2 on MOSN results and MOS results. Also, Sys-3 outperformed Sys-4 on MOSN results and MOS results in most cases. Hence, it can be concluded that, in this task, T2 mostly outperformed FS on naturalness performance.

Table 6.4 Intra-lingual one-shot VC results, where "Dataset" denotes training dataset, "System" denotes the systems in Table 6.1, "WER" denotes word error rate (%), "EER" denotes equal error rate (%), "MOSN" denotes MOSNet, "SS-MOS" denotes speaker similarity MOS. The means and the 95% confidence intervals of the MOS and the SS-MOS results are presented. "$\uparrow$" denotes the higher the better. "$\downarrow$" denotes the lower the better.

Dataset	Index	WER $\downarrow$	EER $\downarrow$	MOSN $\uparrow$	MOS $\uparrow$	SS-MOS $\uparrow$
VCTK	Sys-1	8.9	9.0	4.18	3.84 ± 0.08	3.67 ± 0.06
	Sys-2	8.0	7.7	4.23	3.85 ± 0.08	3.69 ± 0.06
	Sys-3	9.2	8.5	4.11	3.88 ± 0.07	3.86 ± 0.08
	Sys-4	8.9	7.5	4.16	3.86 ± 0.08	3.84 ± 0.08
LibriTTS	Sys-1	9.5	4.8	3.80	3.86 ± 0.07	3.80 ± 0.07
	Sys-2	8.5	7.1	3.84	3.92 ± 0.07	3.52 ± 0.07
	Sys-3	9.7	4.1	3.73	3.82 ± 0.08	3.83 ± 0.09
	Sys-4	9.2	6.8	3.64	3.67 ± 0.09	3.71 ± 0.10

6.4.3 Intra-Lingual One-shot VC Task

Table 6.4 presents results of the intra-lingual one-shot VC task.

Intelligibility

In terms of WER results on both datasets, it is clear that Sys-1 outperformed Sys-3 and Sys-2 outperformed Sys-4. On VCTK, Sys-1 obtained a WER of 8.9%, while Sys-3 got a worse WER of 9.2%. Sys-2 (8.0%) gained a better WER than Sys-4 (8.9(%)). A similar trend can be found on LibriTTS, Sys-1 obtained 9.5%, which is better than Sys-3 (9.7%). Sys-2 (8.5%) gained a better WER than Sys-4 (9.2%). These results indicate that, in this task, VQW2V outperformed BNE on intelligibility performance.

Speaker Similarity

As for speaker similarity, Sys-3 outperformed Sys-1 on EER results and SS-MOS results. Also, Sys-4 obtained better EER and SS-MOS results than Sys-2. On VCTK, Sys-3 obtained an EER of 8.5% and a SS-MOS of 3.86. Sys-1 got a worse EER of 9.0% and a worse SS-MOS of 3.67. Sys-4 obtained an EER of 7.5% and a SS-MOS of 3.84, while Sys-2 got a worse EER of 7.7% and a SS-MOS of 3.69.

A similar story can be found on LibriTTS. Sys-3 obtained an EER of 4.1% and a SS-MOS of 3.83, while Sys-1 obtained a worse EER of 4.8% and a worse SS-MOS of 3.80. Sys-4 got an EER of 6.8 and a SS-MOS of 3.71. Sys-2 got a worse EER of 7.1% and a worse

Table 6.5 Cross-lingual one-shot VC results, where "Dataset" denotes training dataset, "System" denotes the systems in Table 6.1, "WER" denotes word error rate (%), "EER" denotes equal error rate (%), "MOSN" denotes MOSNet, "SS-MOS" denotes speaker similarity MOS. The means and the 95% confidence intervals of the MOS and the SS-MOS results are presented. "↑" denotes the higher the better. "↓" denotes the lower the better.

Dataset	System	WER ↓	EER ↓	MOSN ↑	MOS ↑	SS-MOS ↑
VCTK	Sys-1	9.9	20.1	4.29	3.81 ± 0.07	3.61 ± 0.06
	Sys-2	8.6	27.7	4.32	3.71 ± 0.07	3.60 ± 0.06
	Sys-3	9.1	22.1	4.32	3.77 ± 0.07	3.78 ± 0.08
	Sys-4	8.8	24.8	4.23	3.69 ± 0.07	3.74 ± 0.08
LibriTTS	Sys-1	10.1	14.3	3.70	3.83 ± 0.06	3.79 ± 0.07
	Sys-2	8.2	20.7	3.81	3.83 ± 0.06	3.52 ± 0.06
	Sys-3	8.9	11.9	3.50	3.83 ± 0.06	3.83 ± 0.07
	Sys-4	8.9	18.1	3.57	3.70 ± 0.07	3.76 ± 0.08

SS-MOS of 3.52. These results show that, in this task, BNE outperformed VQW2V on speaker similarity performance.

Naturalness

In terms of MOS results on VCTK, all systems obtained similar results. Sys-3 obtained the best MOS, which is 3.88. It is slightly better than Sys-4 (3.86), Sys-2 (3.84) and Sys-1 (3.84). On LibriTTS, it is clear that Sys-1 obtained higher MOS than Sys-3 and Sys-2 got better MOS than Sys-4. From these results, in this task on VCTK, VQW2V reached comparable naturalness performance of BNE. On LibriTTS, in this task, VQW2V outperformed BNE on naturalness performance.

Comparing FS and T2

From WER results on both datasets, it is obvious that Sys-1 outperformed Sys-2 and Sys-3 outperformed Sys-4. In this task, it can be concluded that T2 outperformed FS on intelligibility performance.

6.4.4 Cross-lingual One-shot VC Task

Table 6.5 presents results of the cross-lingual one-shot VC task. Since target speakers in this task are from non-English languages, generally, compared to the above intra-lingual VC task, VC models suffered from performance degradation on speaker similarity.

Intelligibility

In terms of WER results, on both datasets, Sys-3 outperformed Sys-1 and Sys-2 outperformed 4. On VCTK, Sys-1 obtained a WER of 9.9%, while Sys-3 got a better WER of 9.1%. Sys-2 gained a WER of 8.6%, which is slightly better than Sys-4 (8.8%). On LibriTTS, Sys-1 obtained a WER of 10.1%, while Sys-3 got a better WER of 8.9%. Sys-2 obtained a better WER (8.2%) than Sys-4 (8.9%).

Overall, in this task, the intelligibility performance is dependent on the linguistic encoder and decoder. When the decoder is T2, BNE outperformed VQW2V, however, VQW2V outperformed BNE when the decoder is FS.

Speaker Similarity

On VCTK, Sys-1 obtained a better EER (20.1%) than Sys-3 (22.1%). Sys-2 obtained a worse EER (27.7%) than Sys-4 (24.8%). On LibriTTS, Sys-1 got a worse EER (14.3%) than Sys-3 (11.9%) and Sys-2 obtained a worse EER (20.7%) than Sys-4 (18.1%). On the other hand, SS-MOS showed a clear trend, Sys-3 outperformed Sys-1 and Sys-4 outperformed Sys-2. It is clear that there exists a mismatch between EER metric and SS-MOS metric. Hence, from subjective results, it can be concluded that, in this task, BNE outperformed VQW2V on speaker similarity performance.

Naturalness

As for naturalness, in most cases, Sys-1 obtained better MOS results than Sys-3. Sys-2 obtained better MOS results than Sys-4. On VCTK, Sys-1 obtained a MOS of 3.81, while Sys-3 got a worse MOS of 3.71. Similarly, Sys-2 obtained a MOS of 3.71, while Sys-4 obtained a slightly worse MOS of 3.69. On LibriTTS, Sys-1 and Sys-3 gained the same MOS, 3.83 and Sys-2 obtained a better MOS (3.83) than Sys-4 (3.70). Overall, in most cases, VQW2V outperformed BNE on naturalness.

Comparing T2 and FS

Comparing T2 and FS, it is obvious that, in most cases, Sys-1 obtained better WER results than Sys-2 and Sys-3 obtained better WER results than Sys-4. It can be concluded that, in this task, T2 mostly outperformed FS on intelligibility performance. Another obvious trend is that, on EER and SS-MOS results, Sys-1 outperformed Sys-2 and Sys-3 outperformed Sys-4. These results indicate that, in this task, T2 outperformed FS on speaker similarity performance. On the other hand, in terms of naturalness performance, Sys-1 mostly obtained better MOS than Sys-3 and Sys-1 mostly obtained better MOS than Sys-4. Hence, in this task, T2 mostly outperformed FS on naturalness performance. Overall, T2 showed an obvious advantage over FS in this task.

6.4.5 Analysis: Speech Re-synthesis Task

Table 6.6 VCTK speech re-synthesis results, where "Dataset" denotes training dataset, "System" denotes the systems in Table 6.1, "MCD" denotes mel-cepstral distortion, "F0R" denotes F0-RMSE, "F0C" denotes F0-CORR, "WER" denotes word error rate (%), "EER" denotes equal error rate (%), "MOSN" denotes MOSNet. "↑" denotes the higher the better. "↓" denotes the lower the better.

Dataset	System	MCD ↓	F0R ↓	F0C ↑	WER ↓	EER ↓	MOSN ↑
VCTK	Sys-1	6.58	39.0	0.36	9.9	1.4	3.96
	Sys-2	6.52	44.2	0.31	9.7	1.7	3.86
	Sys-3	7.09	41.83	0.31	9.3	1.8	3.96
	Sys-4	6.96	54.5	0.19	9.5	2.4	3.64
LibriTTS	Sys-1	7.79	68.7	0.42	4.0	7.8	3.32
	Sys-2	7.70	70.3	0.43	4.0	9.0	3.21
	Sys-3	8.12	76.1	0.33	3.6	8.5	3.12
	Sys-4	8.18	80.3	0.27	6.9	11.3	3.17

From results of the speech re-synthesis task, which are presented in Table 6.6, it is obvious that Sys-1 outperformed Sys-3 on four metrics, including MCD, F0R, F0C and EER. The same trend happened on Sys-2 and Sys-4, where Sys-2 obtained outperformed Sys-4 on these metrics. Comparing VQW2V and BNE, VQW2V contains more prosodic information, more speaker information than BNE.

In contrast, on WER results, Sys-3 achieved better performance than Sys-1, Sys-4 achieved better performance than Sys-2. These results show that BNE contains more linguistic information than VQW2V. This is probably because VQW2V is an unsupervised model, while BNE is a supervised model trained with text transcriptions.

As for naturalness, Sys-1 and Sys-3 obtained the same MOSN results on VCTK, 3.96. On LibriTTS, Sys-1 (3.32) outperformed Sys-3 (3.12). On LibriTTS, the Sys-2 obtained better MOSN results than Sys-4. Except for one case when the decoder is T2 and the dataset is VCTK, VQW2V obtained better MOSN results than BNE.

Comparing the Sys-1 and Sys-2, it is obvious that Sys-1 obtained better F0R and better F0C results than Sys-2. Also, Sys-3 obtained better F0R and better F0C results than Sys-4. These mean that T2 as a decoder can better reconstruct prosodic information of an input speech than FS. The same story happens on EER results, where Sys-1 outperformed Sys-2 and Sys-3 outperformed Sys-4. Hence, it can be concluded that T2 can better reconstruct speaker information of an input speech than FS.

6.5 Conclusion

This chapter aimed to compare the VC performance of the unsupervised VQ-Wav2vec model and the supervised ASR-BNE model as the linguistic encoder of an encoder-decoder VC framework.

Firstly, the disentanglement learning performance of the two linguistic encoders were evaluated through a phone classification task and two speaker classification tasks. The results show that ASR-BNE performed better on preserving phonetic information and removing speaker information.

Three VC tasks were set up in the experiments, including a many-to-many VC task, an intra-lingual one-shot VC task and a cross-lingual one-shot VC task. On the many-to-many VC task, VQ-Wav2vec can not outperform ASR-BNE on intelligibility and speaker similarity. VQ-Wav2vec obtained better naturalness performance than ASR-BNE. Furthermore, in the intral-lingual one-shot VC task, VQ-Wav2vec obtained better intelligibility but worse speaker similarity than ASR-BNE. Moreover, in the cross-lingual one-shot VC task, VQ-Wav2vec got better naturalness performance but worse speaker similarity than ASR-BNE.

Chapter 7

Conclusion and Future Works

This thesis mainly studies three types of text-free VC models. The first two parts focused on solving the information loss issue caused by disentanglement learning methods. The third work analysed the performance of SSL based VC models. The research questions can be summarised as follows.

- How can the linguistic information loss issue of the VQ be solved in the VQ-WAE model?
- Will the CIN layer in the StarGAN-VC2 model cause a linguistic information issue? If so, how can one improve the performance and the robustness of the StarGAN-VC2 model?
- Can SSL based VC models reach the same level of performance as the text-dependent VC models, when training on the same dataset?

7.1 Contributions

Solving the Information Loss Issue of the Auto-Encoder based VC Model

Based on observations of the performance of the VQ-WAE model in the ZS 2019 challenge, this has focused on improving the VC performance of the VQ-WAE model. To achieve this, this work proposes two alternative disentanglement methods to solve the information loss issue of the VQ of the VQ-WAE model. The first method is using IN as the disentanglement learning method and further applying AdaIN. The second method is to replace VQ with an extension method of VQ, SVQ. This work proposes two models, namely IN-WAE and SVQ-WAE. The results of the two proposed models were submitted to the ZS 20 challenge for evaluation. The two proposed disentanglement learning methods can alleviate the linguistic information loss issue, so that they can improve the naturalness and the intelligibility of VC. The IN-WAE model performed well on naturalness and intelligibility in the VC task of the ZS 20 challenge. However, the two disentanglement learning methods caused a degradation of speaker similarity performance. In the further analysis, it has been found that these two methods caused phonetic information and speaker-invariant information getting entangled. In addition, it has been found that IN suffered from an information loss issue when input utterance is too short. Furthermore, in the further analysis, the VC performance of the IN-WAE model was impacted by the entanglement, but the SVQ-WAE model performed well on VC. Part of this work has been published in the INTERSPEECH 2020 conference (Chen and Hain, 2020).

Solving the Information Loss Issue of the GAN based VC Model

This work has focused on studying the performance and the robustness of the StarGAN-VC2 model on various training data situations. These training data situations are set up with varying numbers of speakers and varying number of utterances per speaker. It has been found that the StarGAN-VC2 model experienced a performance degradation when decreasing the amount of data per speaker. In order to improve the performance and the robustness of the StarGAN-VC2, a new model is proposed, namely WAGAN-V). This new model introduces a new speaker adaptation layer, namely WAdaIN. It has been found that the WAGAN-VC model outperformed the StarGAN-VC2 model on the VC performance and the robustness. In addition, it has been found that the WAdaIN layer can alleviate the linguistic information loss issue. This work leads to a publication (Chen et al., 2021a) in the ICASSP 2021 conference.

Study the Performance of SSL based Models

Based on an observation of the S3PRL-VC, this work has focused on whether SSL based VC models can reach the same level of performance of text-dependent VC models. To analyse the VC performance of the SSL based VC models and the text-dependent VC models, a framework has been set up for experiments. This framework allows a fairly comparison between VC systems implemented with an ASR model and a VC system implemented with a SSL model. This framework also allows researchers to compare the performance of a VC system implemented with an autoregressive TTS model and the performance of a VC system implemented with a non-autoregressive TTS model. The VQ-Wav2vec model (Baevski et al., 2019) is selected as a representative of SSL models. In the experiments, three VC tasks are set up to cover three VC tasks. It has been found that the SSL based VC models can obtain better naturalness performance, however, poorer speaker similarity. The SSL based

VC models performed better on intelligibility when tested on unseen target speakers. In the speech re-synthesis analysis, it has been found that the VQ-Wav2vec model can preserve more prosodic information than the ASR-BNE models. Apart from this, the performance gap between the autoregressive TTS model and the non-autoregressive TTS model has been clearly shown. The autoregressive TTS model obtained better performance because it can better model the prosodic information of a speech signal.

7.2 Future Works

This section presents potential future work directions.

Study Prosodic Information for VC

It would be worthy to study how to enhance the prosodic information modelling in non-autoregressive TTS models in VC systems. Also, it is worthy to study whether a prosodic encoder is necessary in an encoder-decoder based VC model. How to evaluate the prosodic information of a converted speech sample can also be studied.

Cross-Lingual VC

It has been found that VC models typically experienced a degradation when tested with cross-lingual unseen target speakers. Hence, cross-lingual VC is still a challenging task. How to improve the generalisation performance of VC models in cross-lingual VC tasks can be studied.

Improving Speaker Similarity of SSL based VC Models

It has been found that when preserving more phonetic information in representations, disentanglement learning methods typically caused the entanglement of phonetic information and speaker-dependent information. This caused poor speaker similarity performance of text-free VC models. It is still a question how to improve the speaker similarity performance of text-free VC models when applying disentanglement learning methods.

References

M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara. Voice conversion through vector quantization. *Journal of the Acoustical Society of Japan (E)*, 11(2):71–76, 1990.

A. Acero and R. M. Stern. Robust speech recognition by normalization of the acoustic space., 1991.

N. Ahmed. How i came up with the discrete cosine transform. *Digital Signal Processing*, 1 (1):4–5, 1991.

R. Aihara, T. Takiguchi, and Y. Ariki. Phoneme-discriminative features for dysarthric speech conversion. In *Interspeech*, pages 3374–3378, 2017.

M. Al-Mualla, C. N. Canagarajah, and D. R. Bull. *Video coding for mobile communications: efficiency, complexity and resilience*. Elsevier, 2002.

Amazon. Amazon mechanical turk, 2005. URL https://www.mturk.com/.

D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proc. ICML 2016*, pages 173–182. PMLR, 2016.

L. M. Arslan. Speaker transformation algorithm using segmental codebooks (stasc). *Speech Communication*, 28(3):211–226, 1999.

L. M. Arslan and D. Talkin. Voice conversion by codebook mapping of line spectral frequencies and excitation spectrum. In *Proc. Eurospeech 1997*. Citeseer, 1997.

M. Baas and H. Kamper. Stargan-zsvc: Towards zero-shot voice conversion in low-resource contexts. In *Southern African Conference for Artificial Intelligence Research*, pages 69–84. Springer, 2021.

M. Baas and H. Kamper. Voice conversion can improve asr in very low-resource settings. 2022.

A. Baevski, S. Schneider, and M. Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. In *Proc. ICLR 2019*, 2019.

A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Proc. NIPS 2020*, 33:12449–12460, 2020.

V. Balasubramanian, I. Kobyzev, H. Bahuleyan, I. Shapiro, and O. Vechtomova. Polarized-vae: Proximity based disentangled representation learning for text generation. In *Proc. ACL 2021*, pages 416–423, 2021.

O. Barbany and M. Cernak. FastVC: Fast Voice Conversion with non-parallel data. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 145–149, 2020. doi: 10.21437/VCCBC.2020-21.

R. Bellman and R. Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959.

J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *Proc. NIPS 2006*, 19, 2006.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

H. Benisty, D. Malah, and K. Crammer. Grid-based approximation for voice conversion in low resource environments. *EURASIP Journal on Audio, Speech, and Music Processing*, 2016(1):1–14, 2016.

M. Bernard. Zerospeech challenge 2020 python package, 2020. URL https://github.com/zerospeech/zerospeech2020.

H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

C. H. Chan, K. Qian, Y. Zhang, and M. Hasegawa-Johnson. Speechsplit2. 0: Unsupervised speech disentanglement for voice conversion without tuning autoencoder bottlenecks. In *Proc. ICASSP 2022*, pages 6332–6336. IEEE, 2022.

H. Chen, C.-C. Leung, L. Xie, B. Ma, and H. Li. Parallel inference of dirichlet process gaussian mixture models for unsupervised acoustic modeling: A feasibility study. In *Proc. Interspeech 2015*, 2015.

H. Chen, C.-C. Leung, L. Xie, B. Ma, and H. Li. Multilingual bottle-neck feature learning from untranscribed speech. In *Proc. ASRU 2017*, pages 727–733. IEEE, 2017.

M. Chen and T. Hain. Unsupervised acoustic unit representation learning for voice conversion using wavenet auto-encoders. *Proc. Interspeech 2020*, pages 4866–4870, 2020.

M. Chen, Y. Shi, and T. Hain. Towards low-resource stargan voice conversion using weight adaptive instance normalization. In *Proc. ICASSP 2021*, pages 5949–5953, 2021a. doi: 10.1109/ICASSP39728.2021.9415042.

S. Chen, Y. Wu, C. Wang, S. Liu, Z. Chen, P. Wang, G. Liu, J. Li, J. Wu, X. Yu, et al. Why does self-supervised learning for speech recognition benefit speaker recognition? *Proc. Interspeech 2022*, 2022.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *Proc. ICML 2020*, pages 1597–1607. PMLR, 2020.

X. Chen, W. Chu, J. Guo, and N. Xu. Singing voice conversion with non-parallel data. In *Proc. MIPR 2019*, pages 292–296. IEEE, 2019.

Y.-H. Chen, D.-Y. Wu, T.-H. Wu, and H.-y. Lee. Again-vc: A one-shot voice conversion using activation guidance and adaptive instance normalization. In *Proc. ICASSP 2021*, pages 5954–5958. IEEE, 2021b.

P. Cheng, M. R. Min, D. Shen, C. Malon, Y. Zhang, Y. Li, and L. Carin. Improving disentangled text representation learning with information-theoretic guidance. In *Proc. ACL 2020*, pages 7530–7541, Online, July 2020. Association for Computational Linguistics.

T. Chiba and M. Kajiyama. *The vowel: Its nature and structure*, volume 652. Phonetic society of Japan Tokyo, 1958.

D. G. Childers, K. Wu, D. Hicks, and B. Yegnanarayana. Voice conversion. *Speech Communication*, 8(2):147–158, 1989.

S. Cho, Y. Hong, Y. Shin, and Y. Cho. Vqvae with speaker adversarial training, 2019. URL https://github.com/Suhee05/Zerospeech2019.

Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proc. CVPR 2018*, pages 8789–8797, 2018.

Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proc. CVPR 2020*, pages 8188–8197, 2020.

J. Chorowski, R. J. Weiss, S. Bengio, and A. Van Den Oord. Unsupervised speech representation learning using wavenet autoencoders. *IEEE/ACM transactions on audio, speech, and language processing*, 27(12):2041–2053, 2019.

J.-c. Chou and H.-Y. Lee. One-shot voice conversion by separating speaker and content representations with instance normalization. *Proc. Interspeech 2019*, pages 664–668, 2019.

J.-c. Chou, C.-c. Yeh, H.-y. Lee, and L.-s. Lee. Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations. *Proc. Interspeech 2018*, pages 501–505, 2018.

M. Chu and H. Peng. An objective measure for estimating mos of synthesized speech. In *Proc. Eurospeech 2021*, 2001.

Y.-A. Chung and J. Glass. Generative pre-training for speech with autoregressive predictive coding. In *Proc. ICASSP 2020*, pages 3497–3501. IEEE, 2020.

Y.-A. Chung, H. Tang, and J. Glass. Vector-quantized autoregressive predictive coding. *Proc. Interspeech 2020*, pages 3760–3764, 2020.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12 (ARTICLE):2493–2537, 2011.

G. Dahl, M. Ranzato, A.-r. Mohamed, and G. E. Hinton. Phone recognition with the mean-covariance restricted boltzmann machine. *Proc. NIPS 2010*, 23, 2010.

Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *Proc. ICML 2017*, pages 933–941. PMLR, 2017.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the emalgorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.

L. Deng and X. Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, 2013.

L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *Proc. Interspeech 2010*, 2010.

S. Desai, E. V. Raghavendra, B. Yegnanarayana, A. W. Black, and K. Prahallad. Voice conversion using artificial neural networks. In *Proc. ICASSP 2009*, pages 3893–3896. IEEE, 2009.

B. Desplanques, J. Thienpondt, and K. Demuynck. Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. *Proc. Interspeech 2020*, pages 3830–3834, 2020.

L. Dong, S. Xu, and B. Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *Proc. ICASSP 2018*, pages 5884–5888. IEEE, 2018.

R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *Proc. ICLR 2017*. OpenReview.net, 2017. URL https://openreview.net/forum?id=BJO-BuT1g.

E. Dunbar, X. N. Cao, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera, and E. Dupoux. The zero resource speech challenge 2017. In *Proc. ASRU 2017*, pages 323–330. IEEE, 2017.

E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black, L. Besacier, S. Sakti, and E. Dupoux. The Zero Resource Speech Challenge 2019: TTS Without T. In *Proc. Interspeech 2019*, pages 1088–1092, 2019. doi: 10.21437/Interspeech.2019-2904.

E. Dunbar, J. Karadayi, M. Bernard, X.-N. Cao, R. Algayres, L. Ondel, L. Besacier, S. Sakti, and E. Dupoux. The zero resource speech challenge 2020: Discovering discrete subword and word units. *Proc. Interspeech 2020*, pages 4831–4835, 2020.

M. Eichner, M. Wolff, and R. Hoffmann. Voice characteristics conversion for tts using reverse vtln. In *Proc. ICASSP 2004*, volume 1, pages I–17. IEEE, 2004.

E. Eide and H. Gish. A parametric approach to vocal tract length normalization. In *Proc. ICASSP 1996*, volume 1, pages 346–348. IEEE, 1996.

I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. J. Weiss, and Y. Wu. Parallel tacotron: Non-autoregressive and controllable tts. In *Proc. ICASSP 2021*, pages 5709–5713. IEEE, 2021.

R. Eloff, A. Nortje, B. van Niekerk, A. Govender, L. Nortje, A. Pretorius, E. van Biljon, E. van der Westhuizen, L. van Staden, and H. Kamper. Unsupervised Acoustic Unit Discovery for Speech Synthesis Using Discrete Latent-Variable Neural Networks. In *Proc. Interspeech 2019*, pages 1103–1107, 2019. doi: 10.21437/Interspeech.2019-1518.

D. Erro and A. Moreno. Frame alignment method for cross-lingual voice conversion. In *Proc. Interspeech 2007*, 2007.

D. Erro, E. Navas, and I. Hernáez. Iterative mmse estimation of vocal tract length normalization factors for voice transformation. In *Proc. Interspeech 2012*, 2012.

M. Ferras, C. C. Leung, C. Barras, and J.-L. Gauvain. Constrained mllr for speaker recognition. In *Proc. ICASSP 2007*, volume 4, pages IV–53. IEEE, 2007.

S. Furui. Vector-quantization-based speech recognition and speaker recognition techniques. In *Conference Record of the Twenty-Fifth Asilomar Conference on Signals, Systems & Computers*, pages 954–955. IEEE Computer Society, 1991.

A. Gabryś, G. Huybrechts, M. S. Ribeiro, C.-M. Chien, J. Roth, G. Comini, R. Barra-Chicote, B. Perz, and J. Lorenzo-Trueba. Voice filter: Few-shot text-to-speech speaker adaptation using voice conversion as a post-processing module. In *Proc. ICASSP 2022*, pages 7902–7906. IEEE, 2022.

M. J. Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

M. J. Gales, D. Pye, and P. C. Woodland. Variance compensation within the mllr framework for robust speech recognition and speaker adaptation. In *Proc. ICSLP 1996*, volume 3, pages 1832–1835. IEEE, 1996.

J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon techfnical report n*, 93:27403, 1993.

M. Gerosa, D. Giuliani, S. Narayanan, and A. Potamianos. A review of asr technologies for children's speech. In *Proc. Workshop on Child, Computer and Interaction*, pages 1–8, 2009.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proc. AISTAT 2011*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Proc. NIPS 2014*, 27, 2014.

D. Görür and C. Edward Rasmussen. Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25(4):653–664, 2010.

A. Graves, N. Jaitly, and A.-r. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

R. Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *Proc. Interspeech 2020*, pages 5036–5040, 2020.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. AISTAT 2010*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

J. Harvill, D. Issa, M. Hasegawa-Johnson, and C. Yoo. Synthesis of new words for improved dysarthric speech recognition on an expanded vocabulary. In *Proc. ICASSP 2021*, pages 6428–6432. IEEE, 2021.

T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. Tan. Espnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. In *Proc. ICASSP 2020*, pages 7654–7658. IEEE, 2020.

T. Hayashi, W.-C. Huang, K. Kobayashi, and T. Toda. Non-autoregressive sequence-to-sequence voice conversion. In *Proc. ICASSP 2021*, pages 7068–7072. IEEE, 2021.

M. Heck, S. Sakti, and S. Nakamura. Unsupervised linear discriminant analysis for supporting dpgmm clustering in the zero resource scenario. *Procedia Computer Science*, 81:73–79, 2016.

G. E. Hinton and R. Zemel. Autoencoders, minimum description length and helmholtz free energy. *Proc. NIPS 1993*, 6, 1993.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang. Voice conversion from non-parallel corpora using variational auto-encoder. In *Proc. APSIPA 2016*, pages 1–6. IEEE, 2016.

W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

H. Huang and K. C. Sim. An investigation of augmenting speaker representations to improve speaker normalisation for dnn-based speech recognition. In *Proc. ICASSP 2015*, pages 4610–4613. IEEE, 2015.

J. Huang, E. Marcheret, and K. Visweswariah. Rapid feature space speaker adaptation for multi-stream hmm-based audio-visual speech recognition. In *Proc. ICME 2005*, pages 338–341. IEEE, 2005.

W.-C. Huang, T. Hayashi, S. Watanabe, and T. Toda. The Sequence-to-Sequence Baseline for the Voice Conversion Challenge 2020: Cascading ASR and TTS. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 160–164, 2020a. doi: 10.21437/VCC_BC.2020-24. URL http://dx.doi.org/10.21437/VCC_BC.2020-24.

W.-C. Huang, T. Hayashi, Y.-C. Wu, H. Kameoka, and T. Toda. Voice transformer network: Sequence-to-sequence voice conversion using transformer with text-to-speech pretraining. *Proc. Interspeech 2020*, pages 4676–4680, 2020b.

W.-C. Huang, P. L. Tobing, Y.-C. Wu, K. Kobayashi, and T. Toda. The NU Voice Conversion System for the Voice Conversion Challenge 2020: On the Effectiveness of Sequence-to-sequence Models and Autoregressive Neural Vocoders. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 165–169, 2020c. doi: 10.21437/VCCBC.2020-25.

W.-C. Huang, Y.-C. Wu, and T. Hayashi. Any-to-one sequence-to-sequence voice conversion using self-supervised discrete speech representations. In *Proc. ICASSP 2021*, pages 5944–5948. IEEE, 2021.

W.-C. Huang, B. M. Halpern, L. P. Violeta, O. Scharenborg, and T. Toda. Towards identity preserving normal to dysarthric voice conversion. In *Proc. ICASSP 2022*, pages 6672–6676. IEEE, 2022a.

W.-C. Huang, S.-W. Yang, T. Hayashi, H.-Y. Lee, S. Watanabe, and T. Toda. S3prl-vc: Open-source voice conversion framework with self-supervised speech representations. In *Proc. ICASSP 2022*, 2022b.

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

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML 2015*, pages 448–456. PMLR, 2015.

T. Ishihara and D. Saito. Attention-based speaker embeddings for one-shot voice conversion. In *Proc. Interspeech 2020*, pages 806–810, 2020.

K. Ito.o. The lj speech datasetg, 2017. URL https://keithito.com/LJ-Speech-Dataset.

H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

N. M. Joy, S. R. Kothinti, and S. Umesh. Fmllr speaker normalization with i-vector: In pseudo-fmllr and distillation framework. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(4):797–805, 2018.

A. Kain and M. W. Macon. Spectral voice conversion for text-to-speech synthesis. In *Proc. ICASSP 1998*, volume 1, pages 285–288. IEEE, 1998.

L. Kaiser, S. Bengio, A. Roy, A. Vaswani, N. Parmar, J. Uszkoreit, and N. Shazeer. Fast decoding in sequence models using discrete latent variables. In *Proc. ICML 2018*, pages 2390–2399. PMLR, 2018.

N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu. Efficient neural audio synthesis. In *Proc. ICML 2018*, pages 2410–2419. PMLR, 2018.

H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo. Stargan-vc: Non-parallel many-to-many voice conversion using star generative adversarial networks. In *Proc. SLT 2018*, pages 266–273. IEEE, 2018.

T. Kamm, G. Andreou, and J. Cohen. Vocal tract normalization in speech recognition: Compensating for systematic speaker variability. In *Proc. Annual Speech Research Symposium 1995*, pages 161–167. CLSP, Johns Hopkins University Baltimore, MD, 1995.

H. Kamper, A. Jansen, S. King, and S. Goldwater. Unsupervised lexical clustering of speech segments using fixed-dimensional acoustic embeddings. In *Proc. SLT 2014*, pages 100–105. IEEE, 2014.

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

T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. In *Proc. ICASSP 2019*, pages 6820–6824. IEEE, 2019a.

T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo. Stargan-vc2: Rethinking conditional methods for stargan-based voice conversion. *Proc. Interspeech 2019*, pages 679–683, 2019b.

T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo. Cyclegan-vc3: Examining and improving cyclegan-vcs for mel-spectrogram conversion. In *Proc. Interspeech 2020*, 2020.

S. Karita, S. Watanabe, T. Iwata, M. Delcroix, A. Ogawa, and T. Nakatani. Semi-supervised end-to-end speech recognition using text-to-speech and autoencoders. In *Proc. ICASSP 2019*, pages 6166–6170. IEEE, 2019.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR 2020*, 2020.

H. Kawahara, I. Masuda-Katsuse, and A. De Cheveigne. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds. *Speech communication*, 27 (3-4):187–207, 1999.

H. Kawahara, M. Morise, T. Takahashi, R. Nisimura, T. Irino, and H. Banno. Tandem-straight: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, f0, and aperiodicity estimation. In *Proc. ICASSP 2018*, pages 3933–3936. IEEE, 2008.

J. D. M.-W. C. Kenton and L. K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

D. Kim and D. Yook. Fast speech adaptation in linear spectral domain for additive and convolutional noise. In *Proc. ICSLP 2004*, 2004.

H. Kim and A. Mnih. Disentangling by factorising. In *Proc. ICML 2018*, pages 2649–2658. PMLR, 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR 2019*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *Proc. ICLR 2014*, 2014. URL http://arxiv.org/abs/1312.6114.

K. Kobayashi and T. Toda. sprocket: Open-source voice conversion software. In *Proc. Odyssey 2018*, pages 203–210, 2018.

K. Kobayashi, W.-C. Huang, Y.-C. Wu, P. L. Tobing, T. Hayashi, and T. Toda. crank: An open-source software for nonparallel voice conversion based on vector-quantized variational autoencoder. In *Proc. ICASSP 2021*, pages 5934–5938. IEEE, 2021.

J. Kominek and A. W. Black. The cmu arctic speech databases. In *Proc. SSW 2004*, 2004.

J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Proc. NIPS 2020*, 33:17022–17033, 2020.

R. Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Proc. PacRim 1993*, volume 1, pages 125–128. IEEE, 1993.

K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Proc. NIPS 2019*, 32, 2019.

Y. Le Cun and F. Fogelman-Soulié. Modèles connexionnistes de l'apprentissage. *Intellectica*, 2(1):114–143, 1987.

C.-y. Lee and J. Glass. A nonparametric bayesian approach to acoustic model discovery. In *Proc. ACL 2012*, pages 40–49, 2012.

K.-F. Lee. On large-vocabulary speaker-independent continuous speech recognition. *Speech communication*, 7(4):375–379, 1988.

C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer speech & language*, 9 (2):171–185, 1995.

V. I. Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

H. Li, H. Wang, Z. Yang, and M. Odagaki. Variation autoencoder based network representation learning for classification. In *Proc. ACL Student Research Workshop 2017*, pages 56–61, 2017.

N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu. Neural speech synthesis with transformer network. In *Proc. AAAI 2019*, volume 33, pages 6706–6713, 2019.

Y. Li, H. Erdogan, Y. Gao, and E. Marcheret. Incremental on-line feature space mllr adaptation for telephony speech recognition. In *Proc. ICSLP 2002*, 2002.

Y. A. Li, A. Zare, and N. Mesgarani. Starganv2-vc: A diverse, unsupervised, non-parallel framework for natural-sounding voice conversion. In *Proc. Interspeech 2021*, 2021.

J. Lin, Y. Y. Lin, C. Chien, and H. Lee. S2VC: A framework for any-to-any voice conversion with self-supervised pretrained representations. In H. Hermansky, H. Cernocký, L. Burget, L. Lamel, O. Scharenborg, and P. Motlícek, editors, *Proc. Interspeech 2021*, pages 836–840. ISCA, 2021a. doi: 10.21437/Interspeech.2021-1356. URL https://doi.org/10.21437/Interspeech.2021-1356.

Y. Y. Lin, C.-M. Chien, J.-H. Lin, H.-y. Lee, and L.-s. Lee. Fragmentvc: Any-to-any voice conversion by end-to-end extracting and fusing fine-grained voice fragments with attention. In *Proc. ICASSP 2021*, pages 5939–5943. IEEE, 2021b.

A. T. Liu, P. chun Hsu, and H.-Y. Lee. Unsupervised End-to-End Learning of Discrete Linguistic Units for Voice Conversion. In *Proc. Interspeech 2019*, pages 1108–1112, 2019. doi: 10.21437/Interspeech.2019-2048.

A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee. Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders. In *Proc. ICASSP 2020*, pages 6419–6423. IEEE, 2020a.

F.-H. Liu, R. M. Stern, X. Huang, and A. Acero. Efficient cepstral normalization for robust speech recognition. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1993.

L.-J. Liu, Z.-H. Ling, Y. Jiang, M. Zhou, and L.-R. Dai. Wavenet vocoder with limited training data for voice conversion. In *Proc. Interspeech 2018*, pages 1983–1987, 2018.

L.-J. Liu, Y.-N. Chen, J.-X. Zhang, Y. Jiang, Y.-J. Hu, Z.-H. Ling, and L.-R. Dai. Non-parallel voice conversion with autoregressive conversion model and duration adjustment. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 126–130, 2020b.

S. Liu. Stargan-voice-conversion, 2018. URL https://github.com/liusongxiang/StarGAN-Voice-Conversion.

S. Liu, D. Wang, Y. Cao, L. Sun, X. Wu, S. Kang, Z. Wu, X. Liu, D. Su, D. Yu, et al. End-to-end accent conversion without using native utterances. In *Proc. ICASSP 2020*, pages 6289–6293. IEEE, 2020c.

S. Liu, Y. Cao, D. Wang, X. Wu, X. Liu, and H. Meng. Any-to-many voice conversion with location-relative sequence-to-sequence modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1717–1728, 2021.

A. Liutkus, C.-C. Lo, P. Manuel, and J. Turian. speechmetrics, 2019. URL https://github.com/aliutkus/speechmetrics.

C.-C. Lo, 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. *Proc. Interspeech 2019*, pages 1541–1545, 2019.

J. Lorenzo-Trueba, J. Yamagishi, T. Toda, D. Saito, F. Villavicencio, T. Kinnunen, and Z. Ling. The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods. In *Proc. Odyssey 2018*, pages 195–202, 2018.

H. Lu, Z. Wu, D. Dai, R. Li, S. Kang, J. Jia, and H. Meng. One-shot voice conversion with global speaker embeddings. In *Proc. Interspeech 2019*, pages 669–673, 2019.

B. Ludusan, M. Versteegh, A. Jansen, G. Gravier, X.-N. Cao, M. Johnson, and E. Dupoux. Bridging the gap between speech technology and natural language processing: an evaluation toolbox for term discovery systems. In *Proc. LREC 2014)*, pages 560–567, 2014.

A. L. Maas, P. Qi, Z. Xie, A. Y. Hannun, C. T. Lengerich, D. Jurafsky, and A. Y. Ng. Building dnn acoustic models for large vocabulary speech recognition. *Computer Speech & Language*, 41:195–213, 2017.

A. F. Machado and M. G. d. Queiroz. Voice conversion: A critical survey. In *Proceedings*, 2010.

J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

J. Makhoul, S. Roucos, and H. Gish. Vector quantization in speech coding. *Proc. IEEE*, 73 (11):1551–1588, 1985.

A. D. McCarthy, L. Puzon, and J. Pino. Skinaugment: Auto-encoding speaker conversions for automatic speech translation. In *Proc. ICASSP 2020*, pages 7924–7928. IEEE, 2020.

B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. In *Proc. python in science conference*, volume 8, pages 18–25. Citeseer, 2015.

P. Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, 116:374–388, 1976.

C. Miao, S. Liang, M. Chen, J. Ma, S. Wang, and J. Xiao. Flow-tts: A non-autoregressive network for text to speech based on flow. In *Proc. ICASSP 2020*, pages 7209–7213. IEEE, 2020.

H. Ming, D.-Y. Huang, L. Xie, J. Wu, M. Dong, and H. Li. Deep bidirectional lstm modeling of timbre and prosody for emotional voice conversion. In *Proc Interspeech 2016*, pages 2453–2457, 2016.

S. H. Mohammadi and A. Kain. Semi-supervised training of a voice conversion mapping function using a joint-autoencoder. In *Proc. Interspeech 2015*, 2015.

S. H. Mohammadi and A. Kain. A voice conversion mapping function based on a stacked joint-autoencoder. In *Proc. Interspeech 2016*, pages 1647–1651, 2016.

S. H. Mohammadi and A. Kain. An overview of voice conversion systems. *Speech Communication*, 88:65–82, 2017.

S. H. Mohammadi and T. Kim. Investigation of using disentangled and interpretable representations for one-shot cross-lingual voice conversion. *Proc. Interspeech 2018*, pages 2833–2837, 2018.

M. Morise, F. Yokomori, and K. Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE TRANSACTIONS on Information and Systems*, 99(7):1877–1884, 2016.

A. Mouchtaris, Y. Agiomyrgiannakis, and Y. Stylianou. Conditional vector quantization for voice conversion. In *Proc. ICASSP 2007*, volume 4, pages IV–505. IEEE, 2007.

E. Moulines and F. Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 9(5-6):453–467, 1990.

F. M. Mukhneri, I. Wijayanto, and S. Hadiyoso. Voice conversion for dubbing using linear predictive coding and hidden markov model. *Journal of Southwest Jiaotong University*, 55(4), 2020.

A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: A large-scale speaker identification dataset. *Proc. Interspeech 2017*, pages 2616–2620, 2017.

A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman. Voxceleb: Large-scale speaker verification in the wild. *Computer Speech & Language*, 60:101027, 2020.

R. Naqvi, R. Riaz, and F. Siddiqui. Optimized rtl design and implementation of lzw algorithm for high bandwidth applications. *Electrical Review*, 4:279–285, 2011.

P. Nguyen, C. Wellekens, and J.-C. Junqua. Maximum likelihood eigenspace and mllr for speech recognition in noisy environments. In *Proc. Eurospeech 1999*, 1999.

Q. B. Nguyen, J. Gehring, M. Müller, S. Stüker, and A. Waibel. Multilingual shifting deep bottleneck features for low-resource asr. In *Proc. ICASSP 2014*, pages 5607–5611. IEEE, 2014.

A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *Proc. ICML 2018*, pages 3918–3926. PMLR, 2018.

J.-S. Pan, F. R. McInnes, and M. A. Jack. Fast clustering algorithms for vector quantization. *Pattern Recognition*, 29(3):511–518, 1996.

V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proc. ICASSP 2015*, pages 5206–5210. IEEE, 2015.

A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proc. EMNLP 2016*, pages 2249–2255, 2016.

S.-w. Park, D.-y. Kim, and M.-c. Joe. Cotatron: Transcription-guided speech encoder for any-to-many voice conversion without parallel data. *Proc. Interspeech 2020*, pages 4696–4700, 2020.

T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proc. CVPR 2019*, pages 2337–2346, 2019.

S. H. K. Parthasarathi, B. Hoffmeister, S. Matsoukas, A. Mandal, N. Strom, and S. Garimella. fmllr based feature-space speaker adaptation of dnn acoustic models. In *Proc. Interspeech 2015*, 2015.

A. Paszke, 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. *Proc. NIPS 2019*, 32, 2019.

L. Pepino, P. Riera, and L. Ferrer. Emotion recognition from speech using wav2vec 2.0 embeddings. *Proc. Interspeech 2021*, pages 3400–3404, 2021.

W. Ping, K. Peng, K. Zhao, and Z. Song. Waveflow: A compact flow-based model for raw audio. In *Proc. ICML 2020*, pages 7706–7716. PMLR, 2020.

M. Pitz and H. Ney. Vocal tract normalization equals linear transformation in cepstral space. *IEEE Transactions on Speech and Audio Processing*, 13(5):930–944, 2005.

A. Polyak, Y. Adi, J. Copet, E. Kharitonov, K. Lakhotia, W. Hsu, A. Mohamed, and E. Dupoux. Speech resynthesis from discrete disentangled self-supervised representations. In H. Hermansky, H. Cernocký, L. Burget, L. Lamel, O. Scharenborg, and P. Motlícek, editors, *Proc. Interspeech 2021*, pages 3615–3619. ISCA, 2021.

N. V. Prasad and S. Umesh. Improved cepstral mean and variance normalization using bayesian framework. In *Proc. ASRU 2013*, pages 156–161. IEEE, 2013.

R. Prenger, R. Valle, and B. Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *Proc. ICASSP 2019*, pages 3617–3621. IEEE, 2019.

A. Přibilová and J. Přibil. Non-linear frequency scale mapping for voice conversion in text-to-speech system with cepstral description. *Speech Communication*, 48(12):1691–1703, 2006.

K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson. Autovc: Zero-shot voice style transfer with only autoencoder loss. In *Proc. ICML 2019*, pages 5210–5219. PMLR, 2019.

Y. Qian, J. Xu, and F. K. Soong. A frame mapping based hmm approach to cross-lingual voice transformation. In *Proc. ICASSP 2011*, pages 5120–5123. IEEE, 2011.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Y. Bengio and Y. LeCun, editors, *Proc. ICLR 2016*, 2016. URL http://arxiv.org/abs/1511.06434.

M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio. SpeechBrain: A general-purpose speech toolkit, 2021. URL https://github.com/speechbrain/speechbrain.

R. Rehr and T. Gerkmann. Cepstral noise subtraction for robust automatic speech recognition. In *Proc. ICASSP 2015*, pages 375–378. IEEE, 2015.

Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu. Fastspeech: Fast, robust and controllable text to speech. *Proc. NIPS 2019*, 32, 2019.

Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. In *Proc. ICLR 2020*, 2020.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

M. S. Ribeiro, J. Roth, G. Comini, G. Huybrechts, A. Gabryś, and J. Lorenzo-Trueba. Cross-speaker style transfer for text-to-speech using data augmentation. In *Proc. ICASSP 2022*, pages 6797–6801. IEEE, 2022.

D. Saito, K. Yamamoto, N. Minematsu, and K. Hirose. One-to-many voice conversion based on tensor representation of speaker space. In *Proc. Interspeech 2011*, 2011.

Y. Saito, K. Akuzawa, and K. Tachibana. Joint adversarial training of speech recognition and synthesis models for many-to-one voice conversion using phonetic posteriorgrams. *IEICE Transactions on Information and Systems*, 103(9):1978–1987, 2020.

T. Schatz, V. Peddinti, X.-N. Cao, F. Bach, H. Hermansky, and E. Dupoux. Evaluating speech features with the minimal-pair abx task (ii): Resistance to noise. In *Proc. Interspeech 2019*, 2014.

S. Schneider, A. Baevski, R. Collobert, and M. Auli. wav2vec: Unsupervised pre-training for speech recognition. *Proc. Interspeech 2019*, pages 3465–3469, 2019.

S. R. Schweinberger, H. Kawahara, A. P. Simpson, V. G. Skuk, and R. Zäske. Speaker perception. *Wiley Interdisciplinary Reviews: Cognitive Science*, 5(1):15–25, 2014.

A. Senior and I. Lopez-Moreno. Improving dnn speaker independence with i-vector inputs. In *Proc. ICASSP 2014*, pages 225–229. IEEE, 2014.

S. Shahnawazuddin, N. Adiga, K. Kumar, A. Poddar, and W. Ahmad. Voice conversion based data augmentation to improve children's speech recognition in limited data scenario. In *Proc. Interspeech 2020*, pages 4382–4386, 2020.

J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *Proc. ICASSP 2018*, pages 4779–4783. IEEE, 2018.

H. Shibata, T. Kato, T. Shinozaki, and S. Watanabet. Composite embedding systems for zerospeech2017 track1. In *Proc. ASRU 2017*, pages 747–753. IEEE, 2017.

K. Shikano, S. Nakamura, and M. Abe. Speaker adaptation and voice conversion by codebook mapping. In *Proc. ISCAS 1991*, pages 594–597. IEEE, 1991.

D. K. Singh, P. P. Amin, H. B. Sailor, and H. A. Patil. Data augmentation using cyclegan for end-to-end children asr. In *Proc. EUSIPCO 2021*, pages 511–515. IEEE, 2021.

B. Sisman, J. Yamagishi, S. King, and H. Li. An overview of voice conversion and its challenges: From statistical modeling to deep learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:132–157, 2020.

D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *Proc. ICASSP 2018*, pages 5329–5333. IEEE, 2018.

B. A. Sonkamble and D. Doye. Speech recognition using vector quantization through modified k-meanslbg algorithm. *Computer Engineering and Intelligent Systems*, 3(7), 2012.

B. M. L. Srivastava, N. Vauquier, M. Sahidullah, A. Bellet, M. Tommasi, and E. Vincent. Evaluating voice conversion-based privacy protection against informed attackers. In *Proc. ICASSP 2020*, pages 2802–2806. IEEE, 2020.

K. N. Stevens, S. Kasowski, and C. G. M. Fant. An electrical analog of the vocal tract. *The Journal of the Acoustical Society of America*, 25(4):734–742, 1953.

Y. Stylianou. Applying the harmonic plus noise model in concatenative speech synthesis. *IEEE Transactions on speech and audio processing*, 9(1):21–29, 2001.

Y. Stylianou, O. Cappé, and E. Moulines. Continuous probabilistic transform for voice conversion. *IEEE Transactions on speech and audio processing*, 6(2):131–142, 1998.

L. Sun, S. Kang, K. Li, and H. Meng. Voice conversion using deep bidirectional long short-term memory based recurrent neural networks. In *Proc. ICASSP 2015*, pages 4869–4873. IEEE, 2015.

L. Sun, K. Li, H. Wang, S. Kang, and H. Meng. Phonetic posteriorgrams for many-to-one voice conversion without parallel data training. In *Proc. ICME 2016*, pages 1–6, 2016a. doi: 10.1109/ICME.2016.7552917.

L. Sun, K. Li, H. Wang, S. Kang, and H. Meng. Phonetic posteriorgrams for many-to-one voice conversion without parallel data training. In *Proc. ICME 2016*, pages 1–6. IEEE, 2016b.

D. Sundermann and H. Ney. Vtln-based voice conversion. In *Proc. ISSPIT 2003*, pages 556–559. IEEE, 2003.

D. Sundermann, A. Bonafonte, H. Ney, and H. Hoge. Time domain vocal tract length normalization. In *Proc. ISSPIT 2004*, pages 191–194. IEEE, 2004.

S. Tan and K. C. Sim. Learning utterance-level normalisation using variational autoencoders for robust automatic speech recognition. In *Proc. SLT 2016*, pages 43–49. IEEE, 2016.

K. Tanaka, H. Kameoka, T. Kaneko, and N. Hojo. Atts2s-vc: Sequence-to-sequence voice conversion with attention and context preservation mechanisms. In *Proc. ICASSP 2019*, pages 6805–6809. IEEE, 2019.

H. Tang, X. Zhang, J. Wang, N. Cheng, and J. Xiao. Avqvc: One-shot voice conversion by vector quantization with applying contrastive learning. In *Proc. ICASSP 2022*, pages 4613–4617. IEEE, 2022.

R. Terashima, R. Yamamoto, E. Song, Y. Shirahata, H.-W. Yoon, J.-M. Kim, and K. Tachibana. Cross-Speaker Emotion Transfer for Low-Resource Text-to-Speech Using Non-Parallel Voice Conversion with Pitch-Shift Data Augmentation. In *Proc. Interspeech 2022*, pages 3018–3022, 2022. doi: 10.21437/Interspeech.2022-11278.

S. Tibrewala and H. Hermansky. Multi-band and adaptation approaches to robust speech recognition. In *Proc. Eurospeech 1997*, 1997.

A. Tjandra, B. Sisman, M. Zhang, S. Sakti, H. Li, and S. Nakamura. VQVAE Unsupervised Unit Discovery and Multi-Scale Code2Spec Inverter for Zerospeech Challenge 2019. In *Proc. Interspeech 2019*, pages 1118–1122, 2019. doi: 10.21437/Interspeech.2019-3232.

A. Tjandra, S. Sakti, and S. Nakamura. Transformer VQ-VAE for Unsupervised Unit Discovery and Speech Synthesis: ZeroSpeech 2020 Challenge. In *Proc. Interspeech 2020*, pages 4851–4855, 2020. doi: 10.21437/Interspeech.2020-3033.

T. Toda, H. Saruwatari, and K. Shikano. Voice conversion algorithm based on gaussian mixture model with dynamic frequency warping of straight spectrum. In *Proc. ICASSP 2021*, volume 2, pages 841–844. IEEE, 2001.

T. Toda, Y. Ohtani, and K. Shikano. One-to-many and many-to-one voice conversion based on eigenvoices. In *Proc. ICASSP 2007*, volume 4, pages IV–1249. IEEE, 2007.

T. Toda, L.-H. Chen, D. Saito, F. Villavicencio, M. Wester, Z. Wu, and J. Yamagishi. The voice conversion challenge 2016. In *Proc. Interspeech 2016*, pages 1632–1636, 2016.

O. Turk and L. M. Arslan. Robust processing techniques for voice conversion. *Computer Speech & Language*, 20(4):441–467, 2006.

O. Turk, O. Buyuk, A. Haznedaroglu, and L. M. Arslan. Application of voice conversion for cross-language rap singing transformation. In *Proc. ICASSP 2009*, pages 3597–3600. IEEE, 2009.

L. F. Uebel and P. C. Woodland. An investigation into vocal tract length normalisation. In *Proc. Eurospeech 1999*, 1999.

B. Vachhani, C. Bhat, B. Das, and S. K. Kopparapu. Deep autoencoder based speech features for improved dysarthric speech recognition. In *Proc. Interspeech 2017*, pages 1854–1858, 2017.

H. Valbret, E. Moulines, and J.-P. Tubach. Voice transformation using psola technique. *Speech communication*, 11(2-3):175–187, 1992.

C. Valentini-Botinhao, Z. Wu, and S. King. Towards minimum perceptual error training for dnn-based speech synthesis. In *Proc. Interspeech 2015*, 2015.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, page 125, 2016.

A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Proc. NIPS 2017*, 30, 2017.

B. van Niekerk, M.-A. Carbonneau, J. Zaïdi, M. Baas, H. Seuté, and H. Kamper. A comparison of discrete and soft speech units for improved voice conversion. In *Proc. ICASSP 2022*, pages 6562–6566. IEEE, 2022.

B. Varadarajan, D. Povey, and S. M. Chu. Quick fmllr for speaker adaptation in speech recognition. In *Proc. ICASSP 2008*, pages 4297–4300. IEEE, 2008.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Proc. NIPS 2017*, 30, 2017.

M. Versteegh, R. Thiolliere, T. Schatz, X. N. Cao, X. Anguera, A. Jansen, and E. Dupoux. The zero resource speech challenge 2015. In *Proc. Interspeech 2015*, 2015.

O. Viikki and K. Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1-3):133–147, 1998.

F. Villavicencio and J. Bonada. Applying voice conversion to concatenative singing-voice synthesis. In *Proc. Interspeech 2010*, 2010.

L. Wan, Q. Wang, A. Papir, and I. L. Moreno. Generalized end-to-end loss for speaker verification. In *Proc. ICASSP 2018*, pages 4879–4883. IEEE, 2018.

D. Wang, J. Yu, X. Wu, S. Liu, L. Sun, X. Liu, and H. Meng. End-to-end voice conversion via cross-modal knowledge distillation for dysarthric speech reconstruction. In *Proc. ICASSP 2020*, pages 7744–7748. IEEE, 2020a.

D. Wang, L. Deng, Y. T. Yeung, X. Chen, X. Liu, and H. Meng. VQMIVC: Vector Quantization and Mutual Information-Based Unsupervised Speech Representation Disentanglement for One-Shot Voice Conversion. In *Proc. Interspeech 2021*, pages 1344–1348, 2021. doi: 10.21437/Interspeech.2021-283.

R. Wang, Y. Ding, L. Li, and C. Fan. One-shot voice conversion using star-gan. In *Proc. ICASSP 2020*, pages 7729–7733. IEEE, 2020b.

S. Wang and D. Borth. Noisevc: Towards high quality zero-shot voice conversion. *arXiv preprint arXiv:2104.06074*, 2021.

X. Wang, L. Li, and D. Wang. Vae-based domain adaptation for speaker verification. In *Proc. APSIPA 2019)*, pages 535–539. IEEE, 2019.

S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.

S. Wegmann, D. McAllaster, J. Orloff, and B. Peskin. Speaker normalization on conversational telephone speech. In *Proc. ICASSP 1996*, volume 1, pages 339–341. IEEE, 1996.

D.-Y. Wu and H.-y. Lee. One-shot voice conversion by vector quantization. In *Proc. ICASSP 2020*, pages 7734–7738. IEEE, 2020.

D.-Y. Wu, Y.-H. Chen, and H.-y. Lee. Vqvc+: One-shot voice conversion by vector quantization and u-net architecture. *Proc. Interspeech 2020*, pages 4691–4695, 2020.

Y. Wu and K. He. Group normalization. In *Proc. ECCV 2018*, pages 3–19, 2018.

F.-L. Xie, Y. Qian, Y. Fan, F. K. Soong, and H. Li. Sequence error (se) minimization training of neural network for voice conversion. In *Proc. Interspeech 2014*, 2014.

J. Yamagishi, C. Veaux, K. MacDonald, et al. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92). 2019.

R. Yamamoto. Wavenet vocoder, 2018. URL https://github.com/r9y9/wavenet_vocoder.

R. Yamamoto, E. Song, and J.-M. Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *Proc. ICASSP 2020*, pages 6199–6203. IEEE, 2020.

S. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K. tik Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H. yi Lee. SUPERB: Speech Processing Universal PERformance Benchmark. In *Proc. Interspeech 2021*, pages 1194–1198, 2021. doi: 10.21437/Interspeech.2021-1775.

Z. Yi, W.-C. Huang, X. Tian, J. Yamagishi, R. K. Das, T. Kinnunen, Z.-H. Ling, and T. Toda. Voice conversion challenge 2020—intra-lingual semi-parallel and cross-lingual voice conversion—. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 80–98, 2020.

I.-C. Yoo, K. Lee, S. Leem, H. Oh, B. Ko, and D. Yook. Speaker anonymization for personal information protection using voice conversion techniques. *IEEE Access*, 8:198637–198645, 2020.

H. Zen, Y. Nankaku, and K. Tokuda. Probabilistic feature mapping based on trajectory hmms. In *Proc. Interspeech 2008*, 2008.

H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu. Libritts: A corpus derived from librispeech for text-to-speech. *Proc. Interspeech 2019*, pages 1526–1530, 2019.

D. Zeng and Y. Yu. Voice conversion using structrued gaussian mixture model. In *Proc. ICOSP 2010*, pages 541–544. IEEE, 2010.

H. Zhang. The neteasegames system for voice conversion challenge 2020 with vector-quantization variational autoencoder and wavenet. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 175–179, 2020a.

H. Zhang. The NeteaseGames System for Voice Conversion Challenge 2020 with Vector-quantization Variational Autoencoder and WaveNet. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 175–179, 2020b. doi: 10.21437/VCCBC.2020-27.

J.-X. Zhang, Z.-H. Ling, L.-J. Liu, Y. Jiang, and L.-R. Dai. Sequence-to-sequence acoustic modeling for voice conversion. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(3):631–644, 2019a.

J.-X. Zhang, L.-J. Liu, Y.-N. Chen, Y.-J. Hu, Y. Jiang, Z.-H. Ling, and L.-R. Dai. Voice Conversion by Cascading Automatic Speech Recognition and Text-to-Speech Synthesis with Prosody Transfer. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 121–125, 2020. doi: 10.21437/VCCBC.2020-16.

M. Zhang, X. Wang, F. Fang, H. Li, and J. Yamagishi. Joint training framework for text-to-speech and voice conversion using multi-source tacotron and wavenet. *Proc. Interspeech 2019*, pages 1298–1302, 2019b.

Y. Zhang and J. R. Glass. Towards multi-speaker unsupervised speech pattern discovery. In *Proc ICASSP 2010*, pages 4366–4369. IEEE, 2010.

S. Zhao, H. Wang, T. H. Nguyen, and B. Ma. Towards natural and controllable cross-lingual voice conversion based on neural tts model and phonetic posteriorgram. In *Proc. ICASSP 2021*, pages 5969–5973. IEEE, 2021.

Y. Zhao, M. Kuruvilla-Dugdale, and M. Song. Voice conversion for persons with amyotrophic lateral sclerosis. *IEEE journal of biomedical and health informatics*, 24(10):2942–2949, 2019.

W.-Z. Zheng, J.-Y. Han, C.-K. Lee, Y.-Y. Lin, S.-H. Chang, and Y.-H. Lai. Phonetic posteriorgram-based voice conversion system to improve speech intelligibility of dysarthric patients. *Computer Methods and Programs in Biomedicine*, 215:106602, 2022.

Y. Zhou, X. Tian, H. Xu, R. K. Das, and H. Li. Cross-lingual voice conversion with bilingual phonetic posteriorgram and average modeling. In *Proc. ICASSP 2019*, pages 6790–6794. IEEE, 2019.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV 2017*, pages 2223–2232, 2017.

F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He. Supervised representation learning: Transfer learning with deep autoencoders. In *Proc. IJCAI 2015*, 2015.

Appendix A

The Model Configurations of the VQ-WAE, IN-WAE and SVQ-WAE Models

The VQ-WAE model, the IN-WAE model and the SVQ-WAE model share the same WaveNet decoder, but have different model configurations of the encoder and the bottleneck. This appendix starts with the model configurations of the WaveNet decoder. Then this appendix introduces the model configurations of the encoder and the bottleneck in the three models, respectively.

A.1 The WaveNet Decoder

Following the implementation details of the VQ-WAE model in the paper (Chorowski et al., 2019), the WaveNet decoder contains 2 stacks of dilated causal convolution layers (van den Oord et al., 2016). Each stack contains 10 layers, the dilation sizes increase from 1 to 10. For one dilated convolution layer, the kernel size is 3, stride size is 1. The input and output dimensionality of the convolution layer is 368. The up-sampling module up-samples the latent representations to the sampling-rate of the waveforms. The up-sampling module contains 4 up-sampling blocks. Following the implementations of (Yamamoto, 2018), each up-sampling block contains an interpolation algorithm and a 2d convolution layer. The up-sampling scales of the 4 up-sampling blocks are [2,4,4,5]. The dimensionality of the speaker embedding is 32. The up-sampled latent representations and the expanded speaker embedding are added to the dilated causal convolution layers separately. The skip-connection outputs of the dilated causal convolution layers are fed to two linear layers. The dimensionality of the two linear

Table A.1 The details of the encoder and the bottleneck of the VQ-WAE model, where "Conv1d", "Linear", "K", "S", "In dim", "Out dim", "Residual" denote the 1d convolution layer, linear layer, kernel size, stride size, input dimensionality, output dimensionality, whether or not use residual connection, respectively.

Module	Layer	K	S	In dim	Out dim	Residual
Encoder	Conv1d	3	1	39	768	No
	Conv1d	3	1	768	868	Yes
	Conv1d	4	2	768	768	No
	Conv1d	4	2	768	768	No
	Conv1d	3	2	768	768	Yes
	Conv1d	3	2	768	768	Yes
	Linear	-	-	768	768	Yes
	Linear	-	-	768	768	Yes
	Linear	-	-	768	768	Yes
	Linear	-	-	768	768	Yes
Bottleneck	Linear	-	-	768	64	No
	VQ	-	-	64	64	No
	Conv1d	3	1	64	128	No

layers are 256. A ReLU activation layer is used between the two linear layers. The output of the two linear layers are fed to a softmax layer.

A.2 The Encoder and Bottleneck Model configurations

Table A.1, A.2 and A.3 show the model configurations of the encoder and the bottleneck of the VQ-WAE model, the IN-WAE model and the SVQ-WAE model, respectively. For the SVQ-WAE model, the number of slices in the SVQ block is set to 4.

Table A.2 The details of the encoder module and the bottleneck module of the VQ-WAE model, where "Conv1d", "Linear", "K", "S", "In dim", "Out dim", "Residual", "IN" denote the 1d convolution layer, linear layer, kernel size, stride size, input dimensionality, output dimensionality, whether or not use residual connection, whether or not use the IN layer, respectively.

Module	Layer	K	S	In dim	Out dim	Residual	IN
Encoder	Conv1d	3	1	39	768	No	No
	Conv1d	3	1	768	868	Yes	Yes
	Conv1d	4	2	768	768	No	No
	Conv1d	4	2	768	768	No	Yes
	Conv1d	3	2	768	768	Yes	No
	Conv1d	3	2	768	768	Yes	Yes
	Linear	-	-	768	768	Yes	No
	Linear	-	-	768	768	Yes	Yes
	Linear	-	-	768	768	Yes	No
	Linear	-	-	768	768	Yes	Yes
Bottleneck	Linear	-	-	768	64	No	-
	Conv1d	3	1	64	64	No	-
	AdaIN	-	-	64	64	No	-

Table A.3 The details of the encoder module and the bottleneck module of the SVQ-WAE model, where "Conv1d", "Linear", "K", "S", "In dim", "Out dim", "Residual" denote the 1d convolution layer, linear layer, kernel size, stride size, input dimensionality, output dimensionality, whether or not use residual connection, respectively.

Module	Layer	K	S	In dim	Out dim	Residual
Encoder	Conv1d	3	1	39	768	No
	Conv1d	3	1	768	868	Yes
	Conv1d	4	2	768	768	No
	Conv1d	4	2	768	768	No
	Conv1d	3	2	768	768	Yes
	Conv1d	3	2	768	768	Yes
	Linear	-	-	768	768	Yes
	Linear	-	-	768	768	Yes
	Linear	-	-	768	768	Yes
	Linear	-	-	768	768	Yes
Bottleneck	Linear	-	-	768	64	No
	SVQ	-	-	64	64	No

Appendix B

The Model Configurations of the StarGAN-VC Models

This appendix introduces the model configurations of the StarGAN-VC model (Kameoka et al., 2018) the StarGAN-VC2 model (Kaneko et al., 2019b) and the WAGAN-VC model implemented in Chapter 5. Because there are no official open-source implementations for the StarGAN-VC model and the StarGAN-VC2 model. The model configurations of these two models are from their paper. Some details are slightly different from the original papers.

Figure B.1, B.2 and B.3 illustrate the configurations of the generator, the discriminator and the speaker encoder of the StarGAN-VC model, respectively. Figure B.4, B.5 illustrate the configurations of the generator and the discriminator of the StarGAN-VC2 model, respectively. Figure B.6, B.7 and B.8 illustrate the configurations of the generator, the discriminator and the speaker encoder of the WAGAN-VC model, respectively.

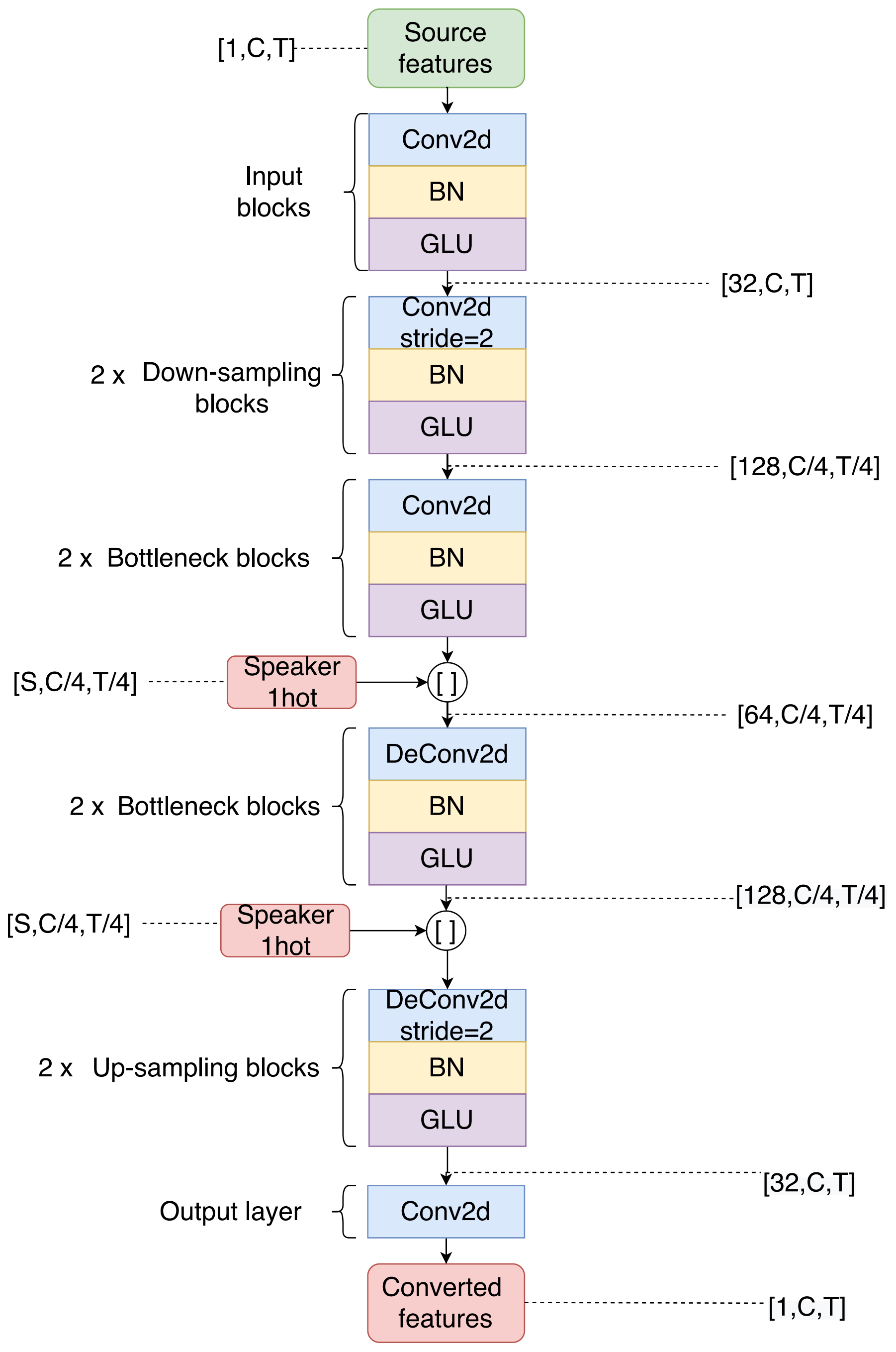

Fig. B.1 The model configurations of the generator of the StarGAN-VC model, where "Conv2d", "BN", "GLU", "DeConv2d", "S", "C", "T" denote the 2d convolution layer, the batch normalisation layer, the gated linear unit layer, the transposed 2d convolutional layer, the number of speakers, the feature dimensionality and the time length, respectively.

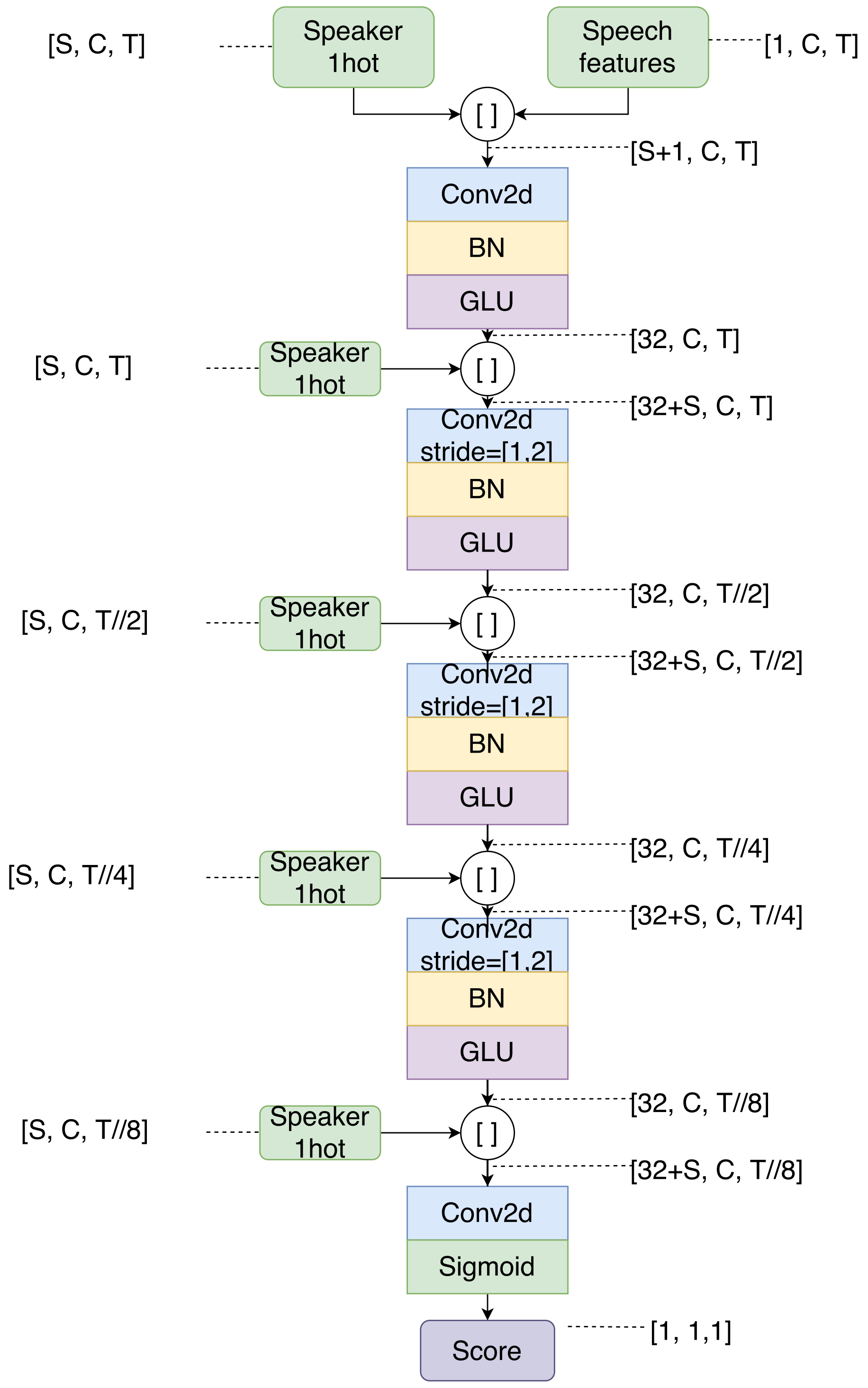

Fig. B.2 The model configurations of the discriminator of the StarGAN-VC model, where "Conv2d", "BN", "GLU", "S", "C", "T" denote the 2d convolution layer, the batch normalisation layer, the gated linear unit layer, the number of speakers, the feature dimensionality and the time length, respectively.

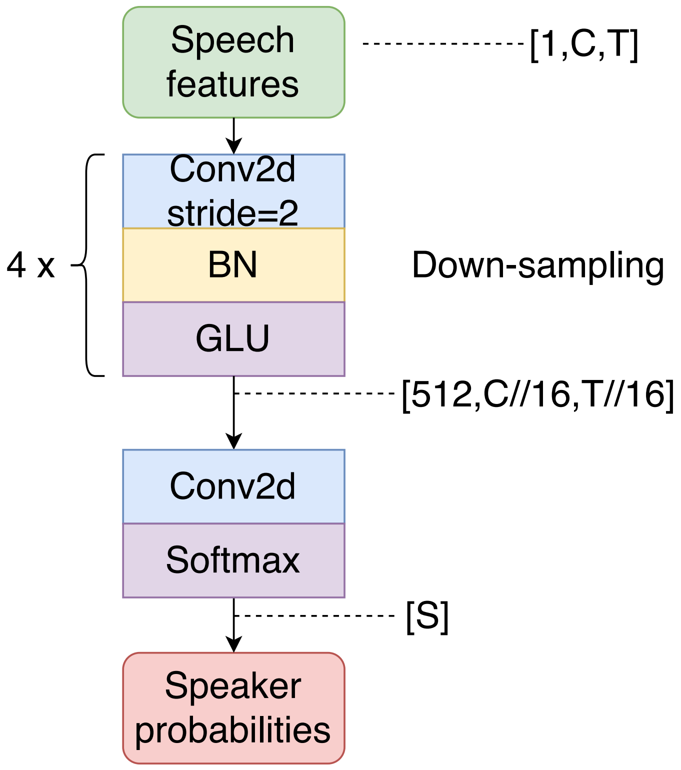

Fig. B.3 The model configurations of the speaker classifier of the StarGAN-VC model, where "Conv2d", "BN", "GLU", "S", "C", "T" denote the 2d convolution layer, the batch normalisation layer, the gated linear unit layer, the number of speakers, the feature dimensionality and the time length, respectively.

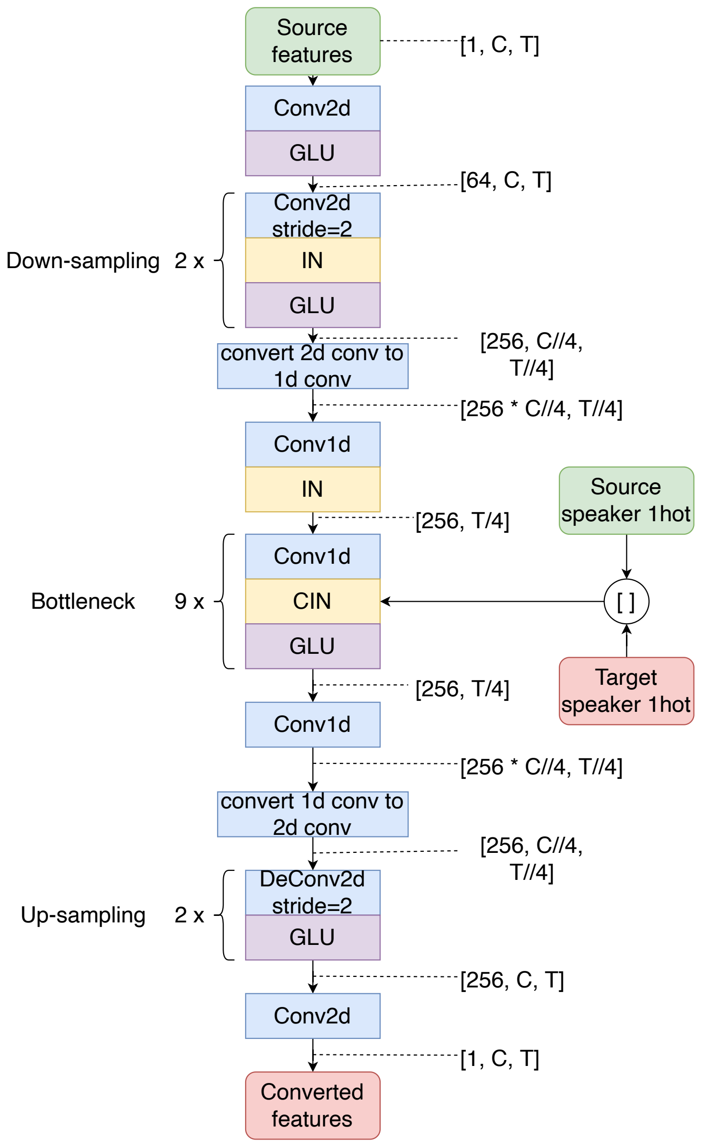

Fig. B.4 The model configurations of the generator of the StarGAN-VC2 model, where "Conv2d", "IN", "GLU", "DeConv2d", "S", "C", "T" denote the 2d convolution layer, the instance normalisation layer, the gated linear unit layer, the transposed 2d convolutional layer, the number of speakers, the feature dimensionality and the time length, respectively.

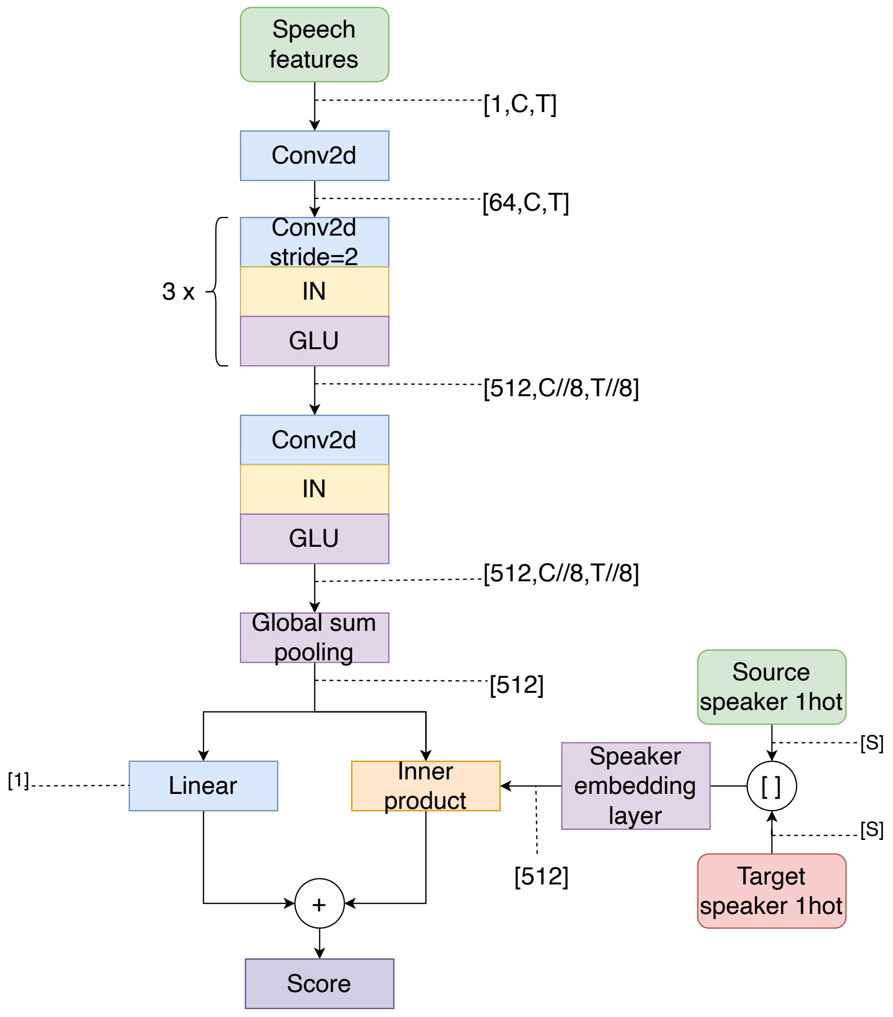

Fig. B.5 The model configurations of the discriminator of the StarGAN-VC2 model, where "Conv2d", "IN", "GLU", "S", "C", "T" denote the 2d convolution layer, the instance normalisation layer, the gated linear unit layer, the number of speakers, the feature dimensionality and the time length, respectively.

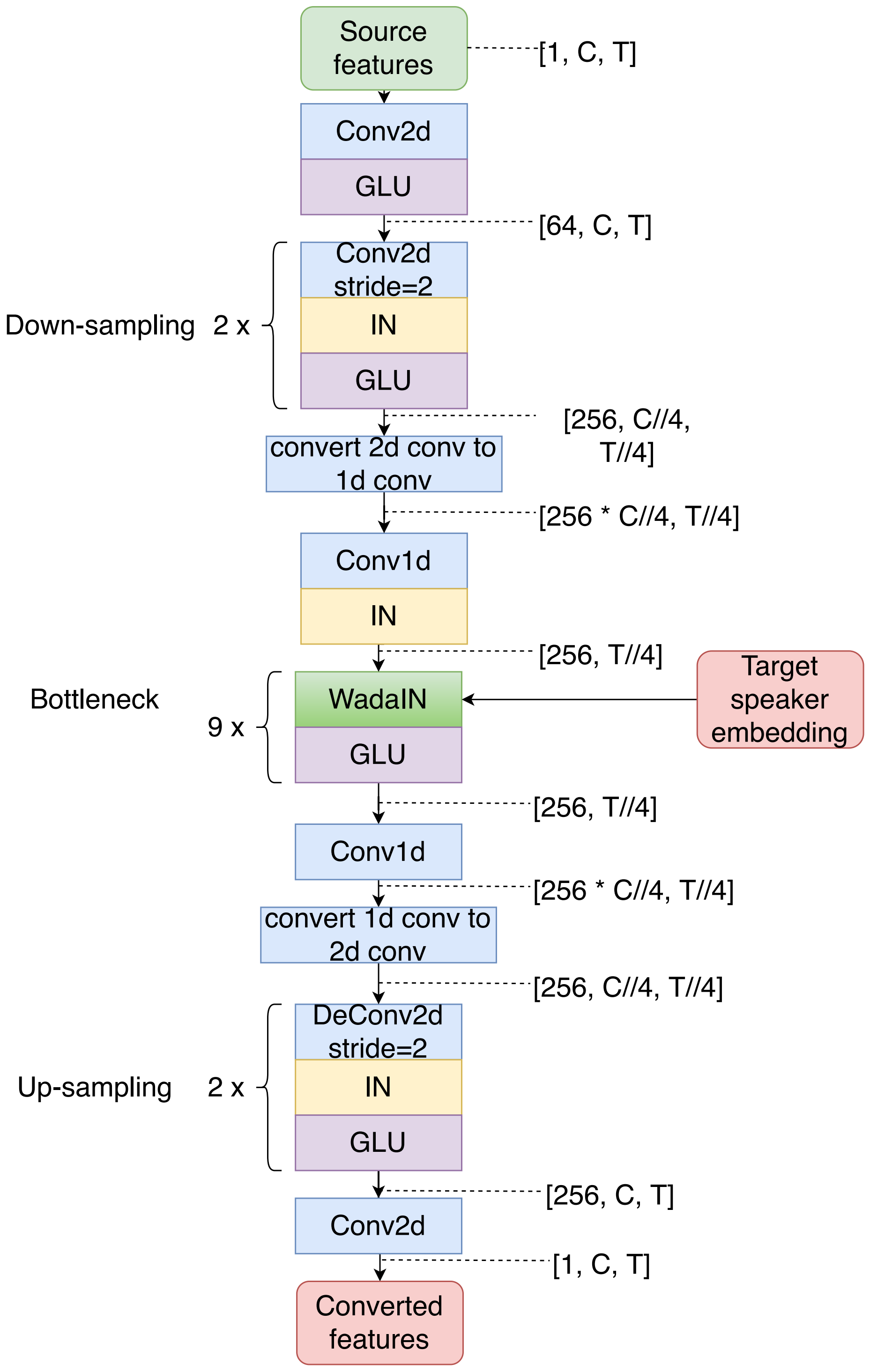

Fig. B.6 The model configurations of the generator of the WAGAN-VC model, where "Conv2d", "IN", "GLU", "DeConv2d", "S", "C", "T" denote the 2d convolution layer, the instance normalisation layer, the gated linear unit layer, the transposed 2d convolutional layer, the number of speakers, the feature dimensionality and the time length, respectively.

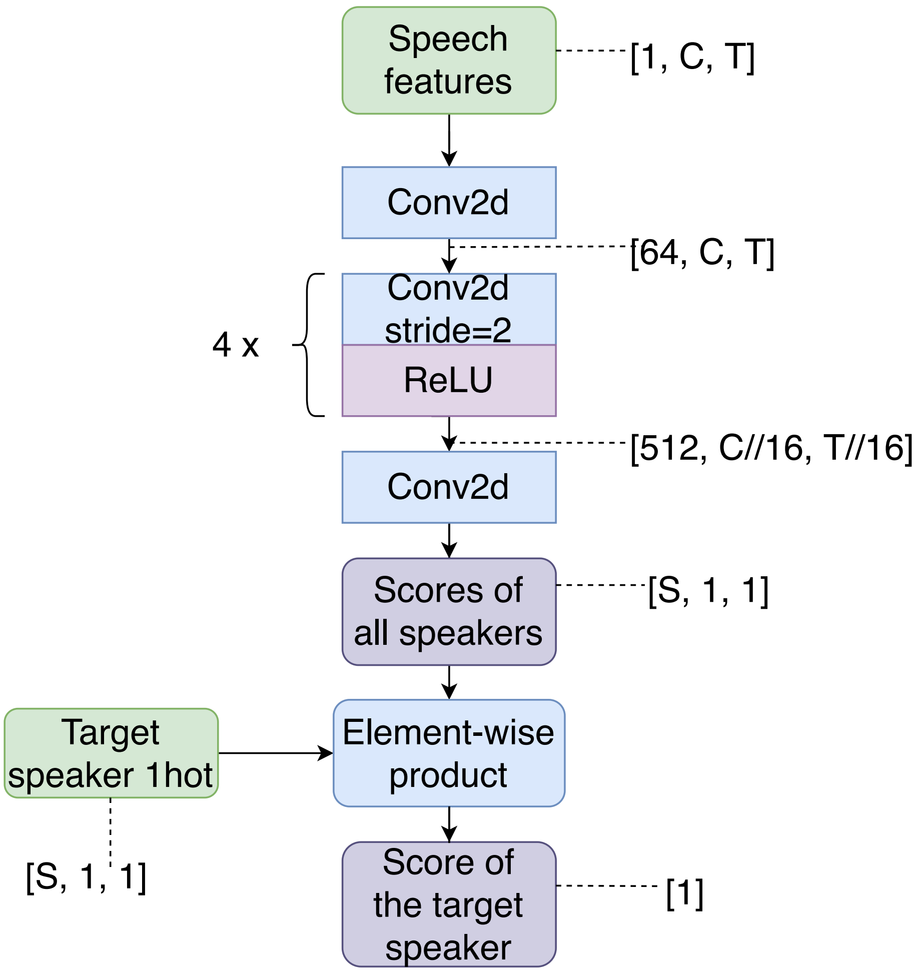

Fig. B.7 The model configurations of the discriminator of the WAGAN-VC model, where "Conv2d", "GLU", "S", "C", "T" denote the 2d convolution layer, the gated linear unit layer, the number of speakers, the feature dimensionality and the time length, respectively.

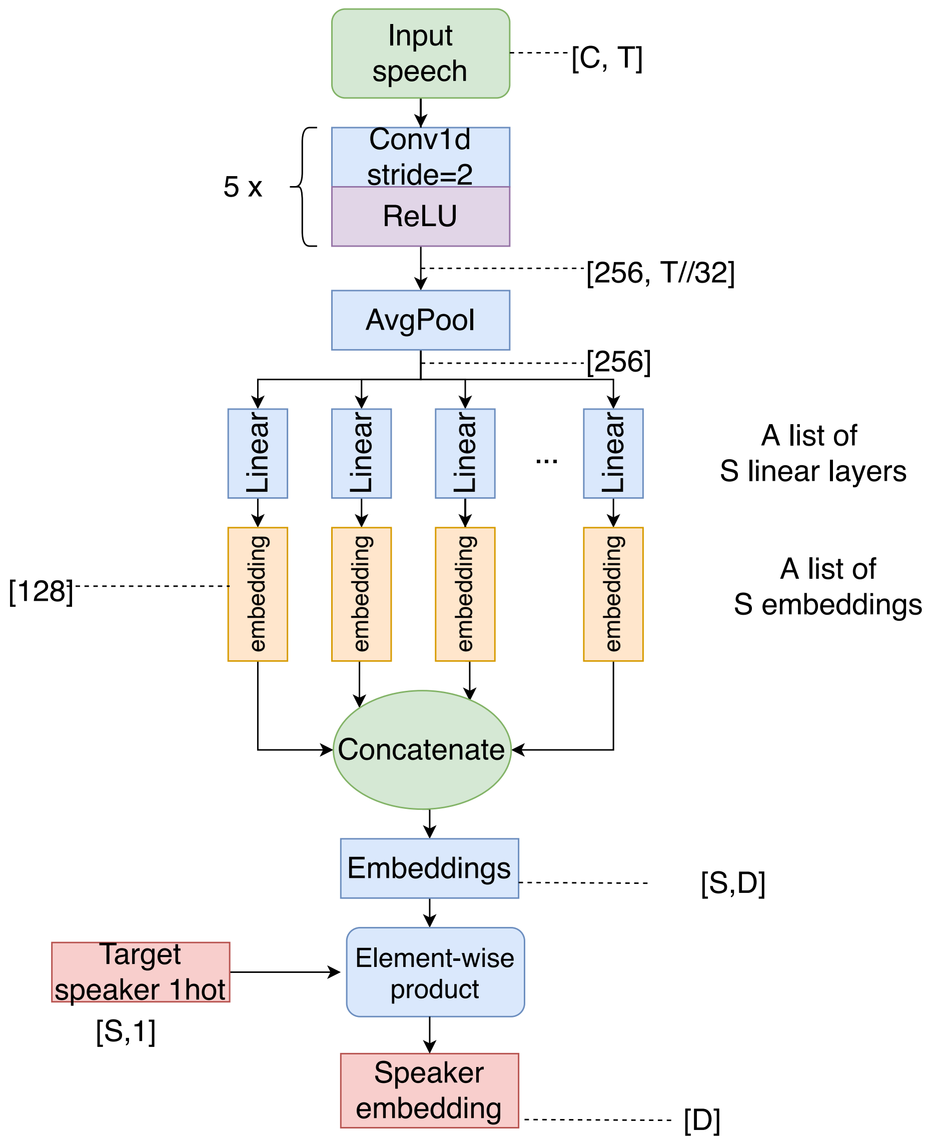

Fig. B.8 The model configurations of the speaker encoder of the WAGAN-VC model, where "Conv1d", "ReLU", "S", "C", "T", "AvgPool" denote the 1d convolution layer, the rectified linear unit layer, the number of speakers, the feature dimensionality, the time length and the average pooling layer respectively.