MSc Artificial Intelligence
Master Thesis

ASR Personalization by Voice Conversion

by

Ties J. van Rozendaal
10077391

September 21, 2018

36 EC
September 2017 - September 2018

Supervisor:
Mr. Taco S. Cohen

Assessors:
Dr. Zeynep Akata
Prof. Dr. Max Welling

Faculteit der Natuurkunde, Wiskunde en Informatica
Abstract

Voice assistants have become deeply embedded into our everyday lives and Automated Speech Recognition (ASR) is now a key component of almost any personal device [3]. In this application, it is critical that the ASR system understands the utterances made by its owner. Improving an ASR system specifically for a single target speaker is called ASR personalization. In the case of personal devices, it is unlikely that we have labeled data of the target speaker, while it might be easy to access the raw unlabeled speech. When we combine the unlabeled speech data of the target speaker with the labeled speech in an ASR training dataset we can do semi-supervised ASR personalization.

Deep neural networks are the state of the art in ASR [2, 26, 13]. They require large amounts of data to train until convergence and fine tuning these models for a specific target speaker would require hundreds of hours of labeled data [26, 2].

We propose to use unsupervised voice conversion for semi-supervised ASR personalization. In voice conversion, we learn a model that generates audio samples in the voice of a target speaker based on input audio of a source speaker. We propose to train a voice conversion model using the target speaker data and use it to convert all of the speech in the ASR training data into the voice of the target speaker. We can then use the generated data to fine-tune the ASR model for the target speaker.

Our voice conversion model is an autoencoder with a speaker conditional-decoder. Related work used either waveform [81] or vocoder [32] audio representations, which differ from the representation used in ASR systems [2], making them less useful for ASR personalization. Furthermore, the methods in these works either have a high computational load [81] or produce low quality voice conversions [32].

We present a 2-dimensional fully convolutional model that is simpler to train than [81] yet is complex enough to achieve reasonable voice conversion performance results.

In addition, we show how different types of latent space regularization lead to different voice conversion results. We demonstrate that models without latent space regularization do not learn to condition on speaker id, while too strongly regularized models fail to generate intelligible speech. We show that variational [32] and quantized [81] autoencoders can perform voice conversion, and that they have slightly different characteristics.

Furthermore, we derive mutual information and conditional entropy metrics and show that they accurately describe the speaker conditionally of the decoder. Our metrics do not only quantify the extent of conditional dependence between speaker and generated audio, but also describe which aspects of the speaker (such as gender or accent) have the highest importance.

We show that our voice conversion model is most suited for gender conversion, and that it struggles to model accent. Furthermore, the dataset characteristics can have a large effect on the performance of our models.

Our voice conversion models do not generalize well to our ASR dataset and as such, we fail to use them for ASR personalization. We do, however, show that our voice conversion models can be used to generally increase ASR performance for all speakers.
Acknowledgements

The past months have been one of the most intense and challenging, yet inspiring ones in my academic career. Writing this thesis would not have been possible without the help and support of many. As such, I would like to take a moment to sincerely thank all of you:

I would like to thank Taco Cohen for supervising me. I really valued his feedback and I am happy that he encouraged me to dive into the unknown and follow my instincts and passion when doing research.

I am grateful to Zeynep Akata and Max Welling for being the accessors of my thesis and taking time despite their very busy schedules.

A big thanks to the people who proofread this thesis: Anne van Rozendaal, Auke Wiggers, Guillaume Saunière, Lisa Steenkamp, Markus Nagel, Mart van Baalen, Maurits Bleeker, and Taco Cohen. Listed in alphabetical order, because every one of you really took the time to read with care, and provided me with comments that were extremely helpful and insightful.

Thank you, all of my colleagues at work, for being supportive and understanding, while I was writing my thesis next to my regular job. Thanks to my managers Taco, Auke, and Peter for being accommodating in my work situation.

Last but definitely not least, I would like to thank my family and my friends, especially my parents (all three of them) and my girlfriend: Thank you for cheering me up at times when I felt challenged, and reassuring me that there is more to life than just this thesis.
# Contents

1 Introduction .............................................. 8
   1.1 Introduction ........................................ 8
   1.1.1 ASR Personalization ............................... 8
   1.1.2 Semi-supervised ASR Personalization ............. 8
   1.1.3 Voice conversion for ASR personalization ....... 9
   1.2 Overview .............................................. 10
   1.3 Contributions ....................................... 10

2 Background ............................................... 11
   2.1 Audio Representation .................................. 11
   2.2 Generative Models ..................................... 14
      2.2.1 Auto-Encoders .................................... 14
      2.2.2 Generative Adversarial Networks ............... 15

3 Framework ............................................... 16
   3.1 Levels of speaker variability ....................... 16
   3.2 Generative model for speech ....................... 17
      3.2.1 Graphical Model .................................. 17
      3.2.2 Notes ............................................. 19

4 Automated Speech Recognition ......................... 21
   4.1 Problem formulation .................................. 21
      4.1.1 Factorization ..................................... 21
      4.1.2 Alignment problem ................................ 22
   4.2 Factorized ASR Systems ................................ 22
      4.2.1 Components ....................................... 22
      4.2.2 Limitations ....................................... 23
   4.3 End to end ASR Systems .............................. 23
      4.3.1 Connectionist Temporal Classification ........... 23
      4.3.2 DeepSpeech 2 ..................................... 25
      4.3.3 Implicit Factorization ............................ 26
   4.4 Evaluation of ASR Systems .......................... 27

5 Voice Conversion .......................................... 28
   5.1 Problem formulation .................................. 28
      5.1.1 High-level formulation ........................... 28
      5.1.2 Levels of voice conversion ..................... 29
   5.2 Evaluating voice conversion ......................... 30
5.2.1 Dealing with subjectivity ........................................... 30
5.2.2 Qualities of voice conversion ..................................... 30
5.3 Related work ............................................................. 31
  5.3.1 Supervised ......................................................... 31
  5.3.2 Unsupervised ..................................................... 34

6 Experimental Setup .................................................. 37
  6.1 Datasets and preprocessing ......................................... 37
    6.1.1 Datasets ..................................................... 37
    6.1.2 Audio preprocessing ........................................ 39
  6.2 Voice Conversion Experiments .................................... 39
    6.2.1 Network Architecture ........................................ 40
    6.2.2 Latent spaces ................................................ 41
    6.2.3 Training procedure .......................................... 43
    6.2.4 Metrics ....................................................... 44
  6.3 ASR ................................................................. 45
  6.4 ASR Personalisation ................................................ 45
    6.4.1 Experiments and baselines .................................. 45
    6.4.2 Personalization procedure .................................. 46

7 Results ................................................................. 47
  7.1 Voice Conversion .................................................. 47
    7.1.1 Qualitative Results .......................................... 47
    7.1.2 Quantitative results ......................................... 51
    7.1.3 Speaker Embedding ......................................... 53
  7.2 Voice conversion for ASR personalization ....................... 54
    7.2.1 Training datasets for ASR personalization ................. 54
    7.2.2 Generalization to the ASR dataset ......................... 54
  7.3 ASR Personalization ............................................... 57
    7.3.1 Male to Female .............................................. 57
    7.3.2 Female to Male ............................................. 57
    7.3.3 Conclusions ................................................ 57
    7.3.4 Qualitative .................................................. 58

8 Conclusion ............................................................. 59
  8.1 Conclusions ........................................................ 59
    8.1.1 Voice Conversion ........................................... 59
    8.1.2 ASR Personalization ....................................... 60
  8.2 Limitations and Future Work ..................................... 60
    8.2.1 Latent space regularization ................................ 60
    8.2.2 Model architecture ......................................... 60
    8.2.3 Loss functions ............................................... 61
    8.2.4 ASR Personalization ....................................... 62

Appendices ............................................................... 64
### A Derivations of Mutual Information

- A.1 Generative model .................................................. 65
  - A.1.1 Speaker and speaker attributes .......................... 65
  - A.1.2 Joint probability ............................................ 66
- A.2 Mutual information and conditional entropy ............... 67
- A.3 Variational entropy lower bound .............................. 67
- A.4 Metrics bounds .................................................... 69
  - A.4.1 Mutual information ......................................... 69
  - A.4.2 Conditional Entropy ........................................ 69
- A.5 Implementation ..................................................... 69

### B ASR Speaker distributions

- B.1 Female to male conversion ...................................... 72
- B.2 Male to female conversion ..................................... 73

### C Model Architectures

- C.1 Encoder Architectures .......................................... 74
- C.2 Decoder Architectures .......................................... 75

### D Online Audio samples

- Glossary ................................................................. 77
- Bibliography .......................................................... 78
Chapter 1

Introduction

1.1 Introduction

1.1.1 ASR Personalization

Over the past decade, natural speech has become a major interface for a range of different personal devices. Nearly half of all Americans use a digital voice assistant [3], and the utilization of voice assistants has extended from smartphones to other applications such as cars, drones, headsets, and smart homes. Essential to the functioning of these voice assistants is a good ASR system to interpret the speech of its user. Over the last years, deep neural networks have become the state of the art in ASR [2, 26, 13]. These models reach the lowest error rates but they require lots of data (1000+ hours of annotated speech) to yield good performance.

Besides that, language complexity can pose a challenge. It is estimated that 1.5 billion people speak English, of which 75% are non-native speakers [12]. Correspondingly, the number of English accents and dialects is endless, making general ASR a difficult problem to solve. Even though deep ASR models tend to generalize well, it is impossible to collect enough annotated data to make them work perfectly for any speaker [7].

While the general goal of an ASR model is to understand any speaker the best it possibly can, we want an ASR model on a personal device to be particularly good in understanding the utterances of its owner. Improving an ASR system specifically for a single target speaker is called ASR personalization.

1.1.2 Semi-supervised ASR Personalization

A trivial solution to the ASR personalization problem is to collect enough annotated speech data for the target speaker until we can train a decent ASR model. In practice, this is an unrealistic scenario because of the effort required in annotating hundreds of hours of speech data for every possible target user. More realistic is the scenario in which we do have access to unlabeled audio
1.1. INTRODUCTION

![Figure 1.1: Our ASR personalization method. Our goal is to improve the performance of a regular ASR system on speech data of a target speaker. The regular ASR model is trained using speech $x_{src, i}$ form source speakers combined with textual transcripts $y_{src, i}$. For our target speaker, we only have speech data $x_{tgt, j}$. We train a voice conversion system using audio data of both the target and the source speaker. We then use the voice conversion model to turn each of our training speech samples $x_{src, i}$ into a new sample $\hat{x}_{src, i}$ that sounds like it was spoken by our target speaker. We then use this converted data together with the original transcripts to train or fine-tune a personalized ASR model. (Solid lines indicate training, dotted lines denote inference.)](image)

...data for the target speaker. This type of data could, for example, be obtained by storing each request that the user makes while using the ASR system.

In **semi-supervised ASR personalization**, we aim to somehow combine the unlabeled speech data $\{x_{tgt, j}\}_{j=1}^{N_{tgt}}$ of our target speaker $stgt$ with a set of labeled data $\{x_{src, i}, y_i\}_{i=1}^{N_{src}}$ by different source speakers. In practice, this could be the data that we would normally use to train an ASR model.

### 1.1.3 Voice conversion for ASR personalization

We propose to use voice conversion as a method of ASR personalization. A voice conversion model can convert the speech of a source speaker $x_{src, i}$ into a sample $\hat{x}_{tgt, i}$ which has the style of a target speaker $stgt$. We aim to train a voice conversion model using the combined audio data of our source and target speakers $\{x_{src, i}\}_{i=1}^{N_{src}} \cup \{x_{tgt, j}\}_{j=1}^{N_{tgt}}$. Because we are not using any transcriptions or information about the alignment between source and target audio, this is known as **unsupervised** voice conversion. We then use the voice conversion model to convert the audio samples in the (labeled) source dataset $\{x_{src, i}\}_{i=1}^{N_{src}}$ into the style of our target speaker $\{\hat{x}_{tgt, i}\}_{i=1}^{N_{src}}$. In the last step of our approach, we use the generated audio together with the original labels $\{\hat{x}_{tgt, i}, y_i\}_{i=1}^{N_{src}}$ to train an ASR model that is personalized for the target speaker. We could either train the model from scratch or finetune an existing ASR model. This entire...
method is summarized in figure 1.1.

1.2 Overview

The structure of this thesis is as follows. In chapter 2 we present a theoretical background in audio representation and unsupervised learning. In chapter 3 we formulate a generative model of speech, which we will use in the subsequent chapters to explain theoretic concepts and related work. We continue in chapter 4 to explain the background and related work in automated speech recognition. In chapter 5 we review the problems and related work in voice conversion, which is the main focus of this thesis. We then outline our experimental setup in chapter 6. Our results are presented in chapter 7 and we discuss our final conclusions in chapter 8.

1.3 Contributions

Our main contributions are:

- We show how different types of latent space regularization lead to different voice conversion results. We demonstrate that models without latent space regularization do not learn to condition on speaker id, while too strongly regularized models fail to generate intelligible speech. We show that variational and quantized autoencoders can perform voice conversion and that they have slightly different characteristics.

- We derive mutual information between speaker and audio and show experimentally that it is a good indicator for the extent of voice conversion. We demonstrate that we can compare the mutual information between models even if they have a different latent space formulation.

- We derive a lower bound on the conditional entropy of speech given various speaker attributes, such as gender or accent. We use this bound to reveal that gender is the most important attribute in our voice conversion models.

- We show that voice conversion can be used as a method to improve the general performance of an ASR system.

- We formulate a generative model for speech and use it to provide various insights into ASR, ASR personalization and voice conversion.
Chapter 2

Background

This chapter lays down the theoretical groundwork, which is assumed to be background knowledge in the rest of this thesis. All of the models that we train and all of the related work that we review in this thesis do use audio as their main input. We look at what an audio signal is, and in what ways we can represent audio in section 2.1. The focus of this thesis is on voice conversion, which is a form of generative modeling. We look at the basics of generative modeling in section 2.2 where we describe the two main building blocks: the variational autoencoder and generative adversarial networks.

2.1 Audio Representation

Everything we do in this thesis revolves around audio, the way that audio is represented can be critical to what we can do with it. In this section, we will answer the questions "What is an audio signal?", and "Which audio representations are most natural to represent speech?"

Waveform

What humans perceive as audio, are in fact changes in air pressure that occur at frequencies within the human hearing range (of an estimated 20 Hz - 20,000 Hz). The most direct representation of this signal is the waveform, that encodes the amplitude of the change as a function of time (see figure 2.1b-I).

Though a waveform is in reality a continuous signal, it must be quantized in order to digitize it (see figure 2.1a). The number of samples recorded per timestep (on the x-axis) is defined as the sample rate. The number of bits used to quantize the amplitude at each timestep, is called the bitdepth. For both parameters, a higher value means a more accurate representation of the signal, at the cost of a more spacious representation. Because of aliasing effects, the highest frequency in a discrete audio signal is half that of the sample rate. For this reason, 44 kHz is a common value for (high fidelity) audio. However for speech signals sample rates as low as 8 kHz still produce very natural results.

Spectrogram

When looking at waveform plots with the naked eye, it can be very difficult to interpret what we would hear if we were to listen to the sample (see figure 2.1b-II). Frequency is the most salient feature for the ear, but it can be difficult to read from a waveplot. Next to this, a natural audio
signal such as speech or music, contains many components with various frequencies that change over time. A spectrogram shows the amount of energy per frequency per timestep, and forms a very natural representation (see figure 2.1-III). In the following paragraphs we will describe how a spectrogram is created, and look at a few representations that can be derived from a spectrogram.

**Short Time Fourier Transform** The complex spectrogram is calculated by performing a short time Fourier transform or STFT. Because we are dealing with discrete, measured waveforms, we will only review the discrete STFT. In a discrete STFT, a window function $w[n]$ is convoluted with the signal $x$, yielding a sequence of frames. On each of those frames, a regular discrete Fourier transform is performed, yielding a sequence of complex numbers with a length of $N_x = |x|$. The convolution of the signal with the window function can be strided by a stride (or hop length) $h$. This yields the following definition:

$$\text{STFT}\{x[n]\} \equiv X(t,f) = \sum_{n=0}^{N_x-1} x[n + tH]w[n]e^{-2\pi i fn/N_x}$$  (2.1)

where $t, f \in \mathbb{Z}$, $t \in [0, N_t)$, $f \in [0, N_f)$ and $X \in \mathbb{C}^{N_t \times N_f}$.

In practice, window functions with a fixed windowsize $N_w$ are used, for which $w[n] = 0$ $\forall n \geq N_w$. The sum over $N_x - 1$ in equation (2.1) will then become a sum over $N_w - 1$. 

---

*(a) Quantization of an audio signal. A waveform is sampled with different sample rates for time resolution (x-axis) and bitdepths for amplitude resolution (y-axis). The source signal is shown in red, sampled signal in blue.*


---

*source: https://commons.wikimedia.org/wiki/File:C_Major_scale_(up_and_down).svg
https://commons.wikimedia.org/wiki/File:C_major.ogg*
This bounds the size of the spectrogram as follows:

\[
N_f = N_w
\]

\[
N_t = \frac{N_c}{2H}
\]

As long as the hop size and window function are such, that overlap between frames is at least 25%, the audio signal can be recovered using the overlap-add method [78], and the STFT is an invertible transformation.

**Complex components** In order to obtain the energies per frequency, we take the magnitude of the complex matrix produced by a STFT. This real-valued matrix represents the energies of each frequency-time bin, and is being referred to as a spectrogram. Note that this operation is not invertible, because we cannot reconstruct the complex numbers by only their magnitude. For this reason, the (real-valued) spectrogram is a lossy representation of the original signal.

**Magnitude scaling** The magnitude of the spectrogram is often rescaled in order to align them better with the perceived intensity. The most common transformations are taking a power of the magnitude, using a log scale (such as the decibel/dB scale) or both.

**Frequency scaling** Similarly the frequency can be converted into a log scale to align it better with human perception. The MEL-scale [69] is a logarithmic scale, that can be used to create a filterbank \( M \in \mathbb{R}^{N_f \times N_m} \) of any size \( N_m < N_f \). This filterbank is simply multiplied with the spectrogram to yield the MEL-spectrogram \( X \). This reduces the frequency dimension by a factor, but introduces additional reconstruction loss.

**Phase representation and reconstruction** In contrast with the magnitude of a complex spectrogram, its phase is a very complex feature. Phase components have very complex correlations that scatter across multiple time and frequency bins, making it seem very chaotic (see figure 2.1b-IV). For this reason, the phase is often ignored in signal processing. When recovering an audio signal from the magnitude of a complex spectrogram, one option is to take a random phase. A more sophisticated solution is using the **Griffin-Lim** algorithm [24] which iterative tries to refine an estimate of the phase.

**Vocoded features** When we want to generate speech, the phase is very difficult to predict. Even when using algorithms such as Griffin-Lim, speech reconstructed from a real-valued spectrogram may sound quite robotic and artificial.

An entire area of research exists, solely devoted to building features that better capture voice characteristics, and make it more easy to synthesize good-sounding speech. These systems are called vocoders, and are often inspired by the physiology of speech production in humans. Most vocoders combine three types of features that are all based around the spectrogram. These features are I: the magnitude of the spectrogram. II: estimated fundamental frequency (F0). III: More advanced phase representations such as derivate of phase energy or unwrapped phase. Taken together these features yield a perfect reconstruction of source audio, and make it easier to generate speech. Some well-known vocoders are STRAIGHT [40], WORLD [57] and MagPhase [16].
CHAPTER 2. BACKGROUND

2.2 Generative Models

A generative model is a model that models a (conditional) data distribution. In this section, we review two types of generative models; the auto-encoder and generative adversarial networks. As we will see later in this thesis, both models can be used as building blocks to create more complex networks.

Note that in this thesis we are interested in generative models for speech, but the principles of auto-encoders and generative adversarial networks can apply to any type of data. We will review both models here in its general form, without making any assumptions about the size or dimensionality of the data.

2.2.1 Auto-Encoders

**Vanilla auto-encoders**

In most machine learning settings we want our data to be represented in a meaningful space, where close points are semantically similar, interpolation between datapoints is smooth, and all plausible datapoints live in a smooth subspace. Such a space is referred to as a manifold, and learning it is a major goal of machine learning.

Auto-encoders are a method to learn such a manifold representation of data in the absence of labels. Auto-encoders consist of an encoder network, that maps the data to a latent feature space $z = \text{enc}(x)$ and a decoder network that tries to recover the data from this latent space $\hat{x} = \text{dec}(z)$. Autoencoders are trained by minimizing the reconstruction error:

$$\arg \min_{\theta} \| \text{dec}_{\theta}(\text{enc}_{\theta}(x)) - x \|$$  \hspace{1cm} (2.2)

This formulation can lead to the trivial solution of the encoder and decoder to learn the identity function. Vanilla auto-encoders rely on the bottleneck principle to avoid this trivial solution; by setting the size of the latent space $Z$ to be a fraction of the original representation, the network is forced to learn a compressed representation of the data.

**Denoising Autoencoders**

Another method to force auto-encoders to learn a meaningful latent space is by adding noise to the input and trying to recover the original, forcing the auto-encoder to learn structural relations in the data.

$$\arg \min_{\theta} \| \text{dec}_{\theta}(\text{enc}_{\theta}(x + \epsilon)) - x \|$$  \hspace{1cm} (2.3)

where $\epsilon \sim p(\epsilon)$ is sampled from a noise distribution.

**Variational auto-encoders**

Because denoising and vanilla auto-encoders make no assumptions about the shape of the latent space, there are no guarantees about this space; is does not have to be smooth, and it is impossible to sample from this space.

Variational auto-encoders use a fully probabilistic model for both $z$ and $x$ where the random variable $z$ is sampled form a prior $p(z)$ and the random variable $x$ is sampled from $p_\theta(x|z)$. Because we defined a prior over the latent space, its shape is now clearly defined (usually a centered isotope multivariate Gaussian is chosen for tractability).

14
2.2. GENERATIVE MODELS

When training a vanilla auto-encoder we are fitting the network to the maximum likelihood estimates \( p(x|z) \) for each \( z \). However, now that we imposed a prior over our latent space, we are interested in the modeling the maximum a posteriori (MAP) probability \( p(z|x) \). This posterior probability is intractable because when we apply Bayes rule we have to evaluate the evidence \( p(x) \) which contains an integral over \( z \).

Instead, we let our encoder network \( q \) define a variational approximation of the posterior probability \( p(z|x) \). We can then rewrite the KL divergence between our encoder and the true distribution, and introduce the evidence \( p(x) \) by multiplying with \( p(x) \) (equation 2.4). When we solve for \( p(x) \) we yield equation 2.5. Note that the only term here which is difficult to evaluate is \( KL[q(z|x)||p(z|x)] \). Because the KL divergence is larger than or equal to 0, the remaining terms form a tractable lower bound for the evidence named ELBO (evidence lower bound).

\[
KL[q(z|x)||p(z|x)] = E_{q(z|x)}[\log q(z|x)] - E_{q(z|x)}[\log p(x, z)] + \log p(x) \tag{2.4}
\]

\[
log p(x) = E_{q(z|x)}[\log p(x|z)] - KL[q(z|x)||p(z)] - KL[q(z|x)||p(z|x)] \tag{2.5}
\]

Though we cannot optimize the posterior or the evidence, we can optimize the ELBO. The expectations over \( q \) can be approximated using Monte Carlo by sampling from \( q(z|x) \).

2.2.2 Generative Adversarial Networks

Generative Adversarial Networks or GANs. \[22\] do not use a latent space but instead aim to directly model the data distribution \( p(x) \). Specifically, they aim to learn a mapping \( G \), called the generator, that maps samples from a known distribution \( z \sim p(z) \) such that they have a high probability under the data distribution \( p(x = G(z)) \). Because we do not know \( p(x) \) we cannot evaluate how well the generator is doing, and thus we cannot optimize its parameters.

Instead, we define a discriminative model that takes a sample \( x \) as its input that can either originate from the generator or from the true data distribution \( p(x) \). The discriminator outputs a single value that reflects the probability that \( x \) originated from the generator \( G \). We can easily train the discriminator using the binary cross-entropy loss (equation 2.6). Additionally, we now have a target for the generator; as we want to generate samples that are likely under the data distribution, we can adversarially optimize the loss of the discriminator. The final objective of a GAN is given by the cross-entropy loss for \( D \) which is optimized in a zero-sum two-player minimax game:

\[
\min_G \max_D V(D, G) = E_{data(x)}[\log D(x)] + E_z p_z(z)[\log(1 - D(G(z)))] \tag{2.6}
\]
Chapter 3

Framework

We speak not only to tell other people what we think, but to tell ourselves what we think. Speech is a part of thought.

Oliver Sacks - Seeing Voices

Before we attempt to find solutions for ASR personalization we need to ask ourselves the question: "What exactly is a speech accent?" and "Why does ASR work better for certain speakers and worse for others?". In this chapter, we will formulate a generative model for speech that shows how speaker characteristics are manifested in speech. We will use this model in the following chapters to explain ASR, voice conversion, ASR personalization, and how they are related to each other.

We start by analyzing the dynamics of speaker style in section 3.1, where we define three levels at which speaker identity is manifested in the speech signal. We then proceed in section 3.2 to formulate a graphical model of speech generation based on these three levels of speaker variability.

Note that we use this model to explain the concepts and difficulties in ASR, voice conversion, and ASR personalization and as such this model is meant to be as close to reality as possible (even though that can mean that it is intractable). Our final voice conversion method follows a different generative model which is described in appendix A.

3.1 Levels of speaker variability

Speaker style and accent are manifested in various ways in a speech signal. In this work, we will distinguish between the following three categories:

• **Transcription level variability:**
  Different speakers might express the same intent using different words. The variability at this level is determined by:
  - Differences in vocabulary; both of words (lexicon) and expressions (idioms).
  - Differences in mastery and use of grammar.

• **Phoneme level variability:**
3.2. GENERATIVE MODEL FOR SPEECH

These are differences below the transcript level. Though phonemes are a direct manifestation of the transcript, different accents can result in a different sequence of phonemes for the same transcription.

Examples are the differences between the British /tu’mərtəʊ/ or American /tu’mer.əʊ/ pronunciation of tomato or the phoneme deletion in ("doing" vs "doin' ").

- **Prosody level variability**: Prosodic differences include all variations in sound that cannot be defined at phoneme level. These are:
  - Differences in Voice, (including fundamental frequency, timbre/harmonics, volume/power)
  - Differences in Stress (including intonation, inflection)
  - Differences in Rhythm (including tempo, pause)

Note that some of these properties are global to the sequence (pitch), while others are time-dependent (intonation).

Some part of the prosodic information contains useful information for speech recognition. Inflection for example, can be essential to the interpunctuation of a sentence.

Other aspects represent information that is meaningful in the context of the conversation such as the emotional state of the speaker or conversational dynamics.

Though prosodic variability can be further split into meaningful sublevels, it is difficult to concisely define or evaluate them. On this ground, we do choose prosody level variability as the lowest level at which speaker variability occurs.

3.2 Generative model for speech

Now that we have defined at which levels we want to model speaker style or accent, we turn to propose a generative model for speech that incorporates all these factors.

3.2.1 Graphical Model

Our generative model is depicted in figure 3.1, we will explain the variables in this section.

**Speaker**

We do assume that speaker is an observed variable throughout this entire thesis and the same assumption is made in all the related work that we review. We define a categorical variable for speaker identity $s_i \in \{0, \ldots, N_s\}$.

A problem with this speaker identity variable is that it is difficult to generalize any function of $s$ to speakers that are not observed in the training data. For this reason, we also define a speaker embedding variable $s_i$ that is a smooth semantic representation of speaker style where $||s_i - s_j||$ is small when $s_i$ and $s_j$ sound similar. Speaker embedding can be treated as a latent variable that is learned implicitly, but there exist various methods that can be used to predict speaker embedding from the audio $p(s_i|\mathbf{x}_i)$ [85, 46, 39, 62]. We can use those methods to estimate a $s_i$ for each $s_i$ which can then be treated as an observed variable.

Figure 3.1: Generalized graphical model for speech generation. This graphical model is used to understand ASR and voice conversion methods in chapters 4 through 5. Semi-shaded nodes indicate that these quantities are observed in some situations, but are not in others. Dashed lines indicate weak dependencies. The meaning of each variable is explained in section 3.2.1. Note that we use this model to explain the concepts and difficulties in ASR, voice conversion, and ASR personalization and as such this model is meant to be as close to reality as possible (even though that can mean that it is intractable). Our final voice conversion method follows a different generative model which is described in appendix A.
3.2. GENERATIVE MODEL FOR SPEECH

Transcript

We assume each speech utterance starts with sampling a transcript $y_j$. The transcript is a variable length sequence of characters s.t. $y_j \in \mathbb{Z}^{T_j}$

Phonemes

This transcript is then converted into a sequence of phonemes $z^{(i,j)}_{\text{phone}}$. Because of the phoneme-level speaker variability (see section 3.1), $z^{(i,j)}_{\text{phone}}$ has a (weak) dependency on $s_i$. Sometimes phonemes are observed, but for many models (and datasets), phonemes are a hidden variable, we will refer to this variable by $z^{(i,j)}_{\text{phone}}$ in either case.

Prosody

The prosodic speaker variability for the sample is encoded by the variable $z^{(i,j)}_{\text{pros}}$. Recall from section 3.1 that this is where most of the speaker variability is modeled. There is still a (weak) dependency on $y^{(i,j)}$ because of inflection, interpunctuation and other prosody that is related to syntax.

As described in section 3.1, some of the prosodic elements are time-dependent (a function of $t$) while others are constant throughout the sequence. We simplify our model by grouping both into a single time-dependent latent variable.

Speech

Finally, the speech sequence $x^{(i,j)}$ is sampled from the distribution $p \left( x^{(i,j)} \middle| z^{(i,j)}_{\text{phone}}, z^{(i,j)}_{\text{pros}} \right)$. In this thesis we will assume the speech signal is a spectrogram of length $T_{(i,j)}$ with a fixed number of frequency bins $N_f$ s.t. $x^{(i,j)} \in \mathbb{R}^{N_f \times T_{(i,j)}}$. We explicitly mention the few cases where $x^{(i,j)}$ is represented as a waveform (in which case $x^{(i,j)} \in \mathbb{R}^{T_{(i,j)}}$), or where $x^{(i,j)}$ is represented by vocoder features (in which case the dimensionality is similar to a spectrogram).

3.2.2 Notes

The above variable definitions provide us with a formal language. We proceed to refine our framework by clarifying some open questions.

Sequence length

Note that the variables $y_j$, $z^{(i,j)}_{\text{phone}}$, $z^{(i,j)}_{\text{pros}}$, and $x^{(i,j)}$ are all sequences, and all have a different length for each sample. In some cases, $y_j$, $x^{(i,j)}$, and $z^{(i,j)}_{\text{phone}}$ are observed, in which case their length is known. In general, the lengths of these three variables is different, even for the same sample $(i,j)$. The timescale of $z^{(i,j)}_{\text{pros}}$ on the other hand, is not very well defined, and it could be set to any length. However, in practice, it is often chosen to be aligned with $x^{(i,j)}$, as it is very closely related to the speech signal. The same convention holds for $z^{(i,j)}_{\text{phone}}$ when it is unobserved. Aligning these sequences is a difficult problem, and so is modeling the dependencies between these sequences. We will not formalize this in our model but rather discuss explicitly how this is dealt with in each model separately.
Speaker/transcript combinations

Note that the graphical model in figure 3.1 contains an intersection between the plate over speaker $i$ and the plate over transcript $j$. However, in practice, we do not always observe a speech sample $x_{(i,j)}$ for each speaker transcript pair $(i,j)$. Instead, the combinations $x_{(i,j)}$ that are observed vary per dataset.

We often use a simplified notation where we use a single index that can be seen as the position of $x_i$ in our dataset ($i \in N_x$). We often use an identical index for different random variables to denote they are observed in pairs. For example, $\{x_i, y_i, s_i\}$ are sampled from the joint $p(x, y, s)$, whereas $\{x_i, y_i, s_j\}$ are independently sampled from $p(x, y)$ and $p(s)$.
Chapter 4

Automated Speech Recognition

To listen is an effort, and just to hear is no merit. A duck hears also.

---

Igor Stravinsky

Before we attempt to personalize any ASR system, we have to understand how ASR systems work. We use our graphical model to analyze the problem of speech recognition and show that ASR can be factorized into three separate components (section 4.1). We then continue to review classical ASR systems that implement these components separately in section 4.2. We proceed to review the next generation of end-to-end ASR systems in section 4.3 and we conclude with evidence that shows that even end-to-end systems implicitly learn a factorization (section 4.3.3).

4.1 Problem formulation

In this section, we will formalize the objective of automated speech recognition using our generative model of speech (from section 3.2.1). This formulation will be used in the subsequent section to describe the ASR systems.

4.1.1 Factorization

The task of speech recognition is finding the most likely transcript $y$ for a given speech sample $x$. We are looking for:

$$y^* = \arg \max_y p(y|x)$$  \hfill (4.1)

Using Bayes theorem we know:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$  \hfill (4.2)

$$\propto p(x|y)p(y)$$  \hfill (4.3)

We can factorize $p(x|y)$ using our the graphical model in figure 3.2.1 yielding:
This equation reveals the components that are required for speech recognition. In traditional ASR systems, these components are separately implemented and trained. We will review these explicitly factorized ASR systems in section 4.2. In the next section, (section 4.3) we will describe end-to-end ASR systems that directly optimize the ASR objective in equation 4.1.

4.1.2 Alignment problem

The main problem in ASR is dealing with the nonlinear alignment between phonemes \( z_{\text{phone}} \) and speech \( x \). Not only do the audio \( x \) and the phonemes \( z_{\text{phone}} \) have different lengths, but the alignment between them is highly nonlinear because of the large variability in phoneme length. Although the alignment is nonlinear, we do know that it is monotonic, because the ordering of elements does not differ between phonemes and speech.

4.2 Factorized ASR Systems

Traditionally, ASR systems explicitly followed the factorization of equation 4.4. In this section, we describe how each component is implemented and how the ASR system is trained.

4.2.1 Components

Language model

The language model is a prior over sentences. The common choice for this component is a simple n-gram model over words based on (smoothed) frequencies in the training data [18], though more complex language models such as recurrent neural networks can be used [54].

Pronunciation model

The pronunciation model defines the probability of phonemes given a sequence of characters. It typically consists of a predefined dictionary [88] that maps each word in \( y \) independently to a single sequence of phonemes \( z_{\text{phone}} \), making integral over \( z_{\text{phone}} \) in equation 4.4 disappear.

This dictionary based pronunciation model can be extended to take into account that the same word can have different pronunciations by adding multiple entries \( z_{\text{phone}} \) for each \( y \). Because in this case \( p(z_{\text{phone}}|y) \) will still be very sparse, the integral over \( z_{\text{phone}} \) in equation 4.4 is reduced to a sum over a the few sequences \( z_{\text{phone}} \) which have a nonzero probability.

Phonetic transcriptions require expert annotations and are expensive to obtain. Most training data is therefore in textual form. For this type of data it is conventional to use the pronunciation model to convert the training data into a phonetic form.

Acoustic model

At the heart of the factorized ASR system we find the acoustic model that performs the complex task of aligning the phonemes with the audio. The most popular class of acoustic models are (linear) hidden Markov models (HMMs). They are well suited to model the likelihood of the audio \( x \) given the phonemes \( z'_{\text{phone}} \) (which are unobserved during inference). HMMs can
4.3. END TO END ASR SYSTEMS

efficiently be fit using the forward-backward algorithm \[4\] and yield good performance on little data \[19\]. The hidden states in a HMM acoustic model reflect the phoneme sequence, and each observed value is one of the spectral frames of the audio signal \[19\]. The most difficult task of HMMs is to model the emission probabilities that define the probability of a spectral frame $x^t$ given the phoneme $z_{phone,t}$. Because phonemes have wildly varying spectral manifestations (see section 3.1), they are not easily modeled.

A traditionally cited emission probability model is a Gaussian mixture model (GMM) \[62\]. The GMM allows for different clusters of phonetic manifestations allowing the HMM to learn a better alignment. Recently, deep neural networks have been outperforming GMMs in modeling emission probabilities \[28\].

Inference

During inference, a Viterbi algorithm \[17\] can be used to efficiently find the sequence of characters that maximizes the posterior probability in equation 4.4 \[18\].

4.2.2 Limitations

Factorized (GMM)-HMM models have been the state of the art in ASR for a long time. They are very data efficient, but they do require a lot of domain specific tuning \[19, 28\].

4.3 End to end ASR Systems

An alternative to the factorized ASR systems in section 4.2 are end-to-end ASR systems. These models are straightforward to train and currently hold the state-of-the-art in speech recognition. In this section we review the DeepSpeech 2 ASR model, which is currently the best performing ASR system, achieving a word error rates as low as 5.33%. An essential component of DeepSpeech2 is the connectionist temporal classification loss. We discuss the necessity for this loss, and how it works in section . We then describe the architecture of DeepSpeech2 and its limitations in section \[2\].

4.3.1 Connectionist Temporal Classification

In this section, we review the connectionist temporal classification loss (CTC loss) \[23\] which circumvents the alignment problem by summing all possible alignments between audio $x$ and text $y$. This is achieved by introducing a many to one mapping of alignments to targets. By carefully choosing the way alignments are represented, the sum of all alignments for a single transcription can be efficiently computed using a dynamic computing algorithm.

Alignment

In general, the input sequence of an ASR model (e.g. a sequence of input frames) contains more elements than the output sequence (which is a sequence or characters). This means one can define an alignment by assigning at most one output character to each input frame. In CTC this is achieved by allowing character repetitions and by introducing the blank character “_” in the output of the model.

The transcription belonging to a sequence of characters predicted by the model is then found by using the two following to step process:
CHAPTER 4. AUTOMATED SPEECH RECOGNITION

Figure 4.1: Example of the CTC loss function. The speech audio sample $x$ is shown segmented into chunks. For each chunk, the network predicts a probability distribution over all single characters (including the blank token '_'). The CTC loss defines the probability of the ground truth 'HELLO WORLD' as the sum over all possible alignments that lead to this transcript. These alignments are shown above the audio sample where a single character is assigned to each chunk of the audio. An alignment is transformed into a transcript by first removing all duplicate characters and then removing all occurrences of the blank (see section 4.3.1 for more detail).

1. Remove all repeated subsequent characters (e.g. "H_EELL_L_O" → "H_EL_L_O");

2. Remove all blank characters "_" (e.g. "H_EL_L_O" → "HELLO")

For example the following sequences all map to the transcript "HELLO WORLD":

- "H_EELL_L_O _WO_RLD_"
- "HH_EL_L_O WOR_L_D"
- "__ELLO WWOOR_LD"

While the sequence "__ELLOWWOOR_LD" maps to the transcript "HELOWORLD". An example for this alignment process is shown in figure 4.1.

Loss function

The CTC loss function can be used in any network that outputs a probability distribution over all characters and the blank character for each frame in the audio signal (or for each $k$ number of frames). The CTC loss for a given target transcript is defined as the negative log likelihood of the target transcription, which in turn is defined as the sum over the probabilities of each alignment that leads to that transcript. See figure 4.1 for an example.

The number of possible alignments for a given transcript is exponential in the length of the audio sample, but fortunately, the summation over all possible alignments can be calculated in an efficient way using a forward-backward algorithm [4, 23]. The reader is referred to [25] for a thorough explanation. The CTC loss is fully differentiable, allowing it to be used to train neural
Networks. The forward-backward algorithm can only be used during training, when the target transcript is known. During inference, the CTC loss has to be optimized using beam search or greedy search [89].

Limitations

The main limitation of the CTC loss is that characters are conditionally independent. This is especially a problem when multiple modes occur (e.g. two different correct spellings of the same word). If a few characters have been predicted that belong to one of the two models, we would want all probability for all remaining characters to peak at that mode, however, due to the conditional independence, the probabilities will remain the same for both modes. As a result of the conditional independence assumption, the CTC loss is not very well suited for learning a flexible language model [89]. For this reason a separate language model is often added during decoding [26]. In this case the ASR model can be seen as a combination of the acoustic model and pronunciation model in equation 4.4.

Alternatives to CTC

There are several alternative ways to deal with alignment in classical sequence-to-sequence learning. These methods can be applied to ASR, but are not as suitable for ASR as the CTC loss. We will explain why in the following paragraphs.

Fixed length encoding

The most simple method is the encoder-decoder architecture that maps each input sequence into a fixed length context vector, and then predicts the output sequence from this vector [72]. Though this method has been applied to ASR [48], it requires many tweaks to work well and does not perform well on longer sequences. Fixed size encodings are too limited to capture an accurate ASR alignment, especially for longer sentences.

Attention

A more flexible alignment architecture is attention. In attention-based models, at each output timestep the decoder has to predict attention weights over all input steps. These attention weights are normalized and multiplied with the input, which is then summed and given as input to the decoder. As a result, the decoder is able to focus at one specific input for each output timestep. ASR systems can be trained using attention [11, 5]. Nonetheless, attention is a too flexible model for alignment, because it allows for non-monotonic mappings.

4.3.2 DeepSpeech 2

The end-to-end deep neural ASR models DeepSpeech [26] and its successor DeepSpeech2 [2] have become the state of the art in ASR with word error rates as low as 5.33 on LibriSpeech (see section 6.1.1 for more information on ASR datasets). In contrast with factorized ASR systems, the DeepSpeech2 model architecture does not require adaptation when applied to different datasets or even different languages. Exemplary of this is the high performance for both English and Mandarin Chinese using the exact same architecture and training procedure [2]. We describe the model architecture of DeepSpeech 2 and its limitations in the following paragraphs.

Network architecture

DeepSpeech2 uses a spectrogram audio representation that is followed by two convolutional layers milliseconds. Because of strides in the convolutions, one character is predicted for every two spectral frames. The convolutional layers are followed by a series of stacked bidirectional
CHAPTER 4. AUTOMATED SPEECH RECOGNITION

Figure 4.2: DeepSpeech2 model architecture. Figure source: [2]

recurrent layers and the final prediction layer is a fully connected softmax layer that is repeated for each timestep. DeepSpeech and DeepSpeech2 are trained using the CTC loss function (see section 4.3.1). The architecture of DeepSpeech2 is summarized in figure 4.2.

Limitations

Though DeepSpeech2 reaches state-of-the-art performance on several datasets, the model requires a lot of data to perform well. Table 10 in their paper shows that at least 2400 hours are required to reach a WER of 11.65 % on LibriSpeech. Performance quickly degrades when fewer than 1200 hours are used.

4.3.3 Implicit Factorization

We have reviewed factorized ASR systems (section 4.2) and end-to-end models (section 4.3) separately, but we hypothesize that end-to-end models implicitly learn a similar factorization. We will present evidence for this hypothesis for the DeepSpeech2 model.

The first convolutional layers of DeepSpeech2 can be viewed as a phoneme classifier (acoustic model in equation 4.4). They have short temporal dependencies in line with the observations that phoneme classification is largely context-independent. This is further confirmed by [6] who show that the lower convolutional layers of DeepSpeech are most predictive for phoneme class.

The higher recurrent layers of DeepSpeech can be thought of as the pronunciation and language model (in equation 4.4). Recurrent neural networks can model long temporal dependencies and are the common choice for language models [72, 49]. In line with this theory, cold-fusion, a method for integrating an external language model into a pre-trained DeepSpeech model, shows best results on the upper layers [68].
4.4 Evaluation of ASR Systems

The most commonly used metrics to quantify ASR performance are character- and word error rate (CER and WER respectively). The error rates are calculated by finding the optimal alignment between the predicted transcript (hypothesis) and the ground-truth (reference). The optimal alignment is defined by the Levenstein (or edit) distance [45], which encodes the minimal number of edits one has to make to the hypothesis in order to obtain the reference. An edit can either be a substitution, deletion, or an insertion. The error rate between hypothesis $h$ and reference $r$ is defined as the Levenstein distance normalized by reference length:

\[
 e = \frac{\text{lev}(h, r)}{|r|} = \frac{s + i + d}{n} \quad (4.5)
\]

where $\text{lev}$ denotes the Levenstein distance, and $s, i, d$ denote the number of substitutions, insertions and deletions respectively required to obtain $r$ from $h$. $n$ equals the number of units in the reference. In the CER, each unit is a character whereas for WER each word is considered a single unit. Consequently, the word error rate is more stringent, as even when only one character is predicted incorrectly, the entire word is counted as wrong.
Chapter 5

Voice Conversion

Give every man thy ear, but few thy voice.

William Shakespeare - Hamlet

Voice conversion is the key component in our ASR personalization scheme (summarized in figure 1.1). In this chapter, we will give a comprehensive overview of the field of voice conversion and discuss how it fits into our personalization setup. Our goal is to find the best voice conversion model for use in ASR personalization.

We begin by outlining the problem of voice conversion in section 5.1 using our graphical model from chapter 3. In the next section (section 5.2), we discuss the problem of evaluating voice conversion systems, and we highlight which aspects of voice conversion we consider to be the most important. We then review the related work in both supervised and unsupervised voice conversion in section 5.3.

5.1 Problem formulation

Although the question "What is voice conversion?" may be easy to answer intuitively, it is difficult to formalize this notion (the notion being "voice conversion", or the question?). We will do so in this section, starting with a high-level formulation in section 5.1.1 which we will then proceed to refine in section 5.1.2 (using our framework from section 3). These definitions will guide us through the rest of this chapter.

5.1.1 High-level formulation

Voice conversion, also known as voice conversion, is a form of style transfer, where we want to generate a target audio sample \(\hat{x}_{tgt,n}\) that has the semantic content \(C\) of a source sample \(x_{src,n}\), but exhibits the style of a target speaker \(s_{tgt}\). By this definition, the converted sample \(\hat{x}_{tgt,n}\) is conditionally independent of the source sample \(x_{src,n}\) given the semantic content \(C\):

\[
\hat{x}_{tgt,n} \sim p(\hat{x}_{tgt,n}|x_{src,n}, s_{tgt}) = \int p(\hat{x}_{tgt,n}|C, s_{tgt})p(C|x_{src,n})dC
\] (5.1)

We can view voice conversion as a two-stage process were we use ancestral sampling to sample from the joint \(p(\hat{x}_{tgt,n}, C|x_{src,n}, s_{tgt})\) (and we discard the semantic content \(C\)):
5.1. PROBLEM FORMULATION

\[ \hat{C} \sim p(C|x_{src,n}) \] (5.2)
\[ x_{tgt,n} \sim p(x_{tgt,n}|\hat{C}, s_{tgt}) \] (5.3)

5.1.2 Levels of voice conversion

Equation 5.1 provides with a formalization of the voice conversion objective, but it involves a semantic content variable \( C \) which is not well defined. In this section, we will use our framework form section 3.2.1 to complete the objective of voice conversion.

Semantic content can be defined at various levels. We will look at the three levels of speaker variability from section 3.1. At each level, we will define semantic content \( C \) as one of the nodes in the graph in figure 3.1. We will factorize the voice conversion objective at each level, and discuss the dynamics it imposes on voice conversion.

- **Intent level consistency** \( C = z_{\text{intent}} \), allowing for transcription level variability
  
  When we want to allow for transcription level speaker variability, we have to define consistency at a higher, intentional level.
  
  As a result, the converted sample \( x_{tgt} \) could contain different words than \( x_{src} \) to express the same intent.
  
  Note that this is usually seen as an undesirable property, and none of the voice conversion models described operate on this level.
  
  This level of voice conversion cannot be modeled by our graphical model in figure 3.1 because the intent variable is not modeled.

- **Transcription level consistency** \( C = y_n \), allowing for phoneme level variability
  
  At this level, we consider two audio samples semantically identical iff they have same transcript \( y_n \).
  
  The objective of voice conversion in equations 5.1 and 5.3 factorize as follows:
  
  \[ \hat{y}_n \sim p(y_n|x_{src,n}) \] (5.4)
  \[ x_{tgt,n} \sim p(x_{tgt,n}|z_{\text{pros}_{n,tgt}}, z_{\text{phone}_{n,tgt}}) \]
  \[ \times p(z_{\text{pros}_{n,tgt}}|y_n, s_{tgt}) \]
  \[ \times p(z_{\text{phone}_{n,tgt}}|y_n, s_{tgt}) \] (5.5)
  
  or when ignoring the weak relations:
  
  \[ x_{tgt,n} \sim p(x_{tgt,n}|z_{\text{pros}_{n,tgt}}, z_{\text{phone}_{n,tgt}}) \] \[ \times p(z_{\text{pros}_{n,tgt}}|s_{tgt}) \]
  \[ \times p(z_{\text{phone}_{n,tgt}}|y_n) \] (5.6)
  
  Note that equation 5.5 essentially defines an ASR system.
  
  Voice conversion systems at this level can model speaker variability in phoneme and prosody.

- **Phoneme level consistency** \( C = z_{\text{phone}} \), allowing for prosody level variability:
  
  At this level semantic consistency is defined as two audio samples having the same sequence of phonemes \( z_{\text{phone}} \), allowing only for variations in prosody.
  
  The objective of voice conversion in equations 5.1 and 5.3 factorize as follows:
  
  \[ \hat{z}_{\text{phone}} \sim p(z_{\text{phone}}|x_{src,n}) \] (5.7)
  \[ x_{tgt,n} \sim p(x_{tgt,n}|z_{\text{pros}_{tgt}}, z_{\text{phone}_n}) \]
  \[ \times p(z_{\text{pros}_{tgt}}|s_{tgt}) \] (5.8)
5.2 Evaluating voice conversion

Before we turn to the voice conversion literature and start to compare models we need to be able to answer the question "What is a good voice conversion model?". In this section, we will look at the difficult problem of evaluating a voice conversion system. In section 5.2.1, we explain why evaluation is inherently subjective and how we will deal with that fact throughout this work. In the next section (section 5.2.2) we describe different properties of voice conversion models and motivate which of them are important in our ASR personalization setup.

5.2.1 Dealing with subjectivity

Evaluating a voice conversion system is inherently a task of a subjective nature. The main problem is that there are no ground truth samples for voice conversion, as there is not a single "correct" conversion, but rather many possible solutions that can all be equally "good".

Mean opinion scores As most voice conversion qualities cannot be measured by any quantitative metrics, a common way to quantify these properties is to take the mean opinion score (MOS) of a group of human raters. MOS are the standard for evaluation in the voice conversion literature even though they are noisy, expensive to obtain and highly dependent on the pool of reviewers.

References to audio samples In this work we will mainly resort to manual subjective interpretation of voice conversion results. We will try to convince the reader of our interpretation by providing original sounds samples wherever possible. We will refer to sound samples on the web using the following icon: \(\text{volume_up}i\). Clicking the icon will take you to the sound samples, a footnote on the page will show the full URL to the files.

5.2.2 Qualities of voice conversion

There are many different qualities that describe the performance of a voice conversion system. The two main qualities on which a voice conversion system is typically rated are voice quality and conversion quality, though both can be divided into various sub-scales [76, 47].

We selected three qualities of voice conversion that are most relevant when we want to apply it for ASR personalization. In the next paragraphs, we will describe each metric, list alternatives to the MOS to measure them, and explain their relevance to ASR personalization.

Target Speaker Similarity Target speaker similarity describes how close the converted voice is to the target speaker. There are several quantitative measurements for target speaker similarity. One group of metrics directly compare statistics (such as pitch, or spectral envelope) of the converted audio signal to statistics of the target speaker [70, 32]. Another approach are adversarial metrics, that measure how well the converted sample can fool a network trained on a speaker classification or verification task. The accuracy of the adversarial metrics depends largely on the quality of the speaker verification model, and its ability to generalize [47].

Because tuning, using and interpreting these automatic metrics is an extensive nontrivial task, we will resort to manual subjective evaluation for target speaker similarity. High target speaker similarity is essential in our ASR personalization scheme because the converted data has to match the distribution of our target speaker.
5.3. RELATED WORK

Voice Intelligibility  Voice Intelligibility is a sub-scale of voice quality that describes how comprehensive the speech is. A voice sample can have a high voice quality, but a low voice intelligibility. A proxy for voice intelligibility is the error rate of an ASR model on the converted sample. This measure depends on the quality of the ASR model and how well it generalizes to the speaker in question.

This ASR metric is not directly applicable in our scenario because we want to use voice conversion to generate audio of accented voices, voices on which our ASR system is known to perform badly. However, the increase in ASR performance after our voice conversion data augmentation gives us some indication of voice intelligibility. Voice conversion systems should have high voice intelligibility when used in ASR personalization, as we will use the converted audio as ground truth when training our personalized ASR system.

Voice Quality  This is a sub-scale of voice quality that describes the quality of the voice itself and which includes aspects such as audio quality and naturalness. A speech sample can have high intelligibility but low audio quality but they do require a ground truth and are thus not applicable in voice conversion.

Although voice quality plays some role in our data augmentation scheme, it is generally not very important. As long as the converted samples are intelligible and have high target speaker similarity, the ASR system should improve when trained on them. In fact, low voice quality could help improve the model, as it is a form of noise augmentation.

5.3 Related work

Now that we have given a problem definition and criteria for evaluation we have set the right scope to review the related work in voice conversion. The most relevant to our ASR personalization pipeline are the supervised voice conversion methods, which we review in section 5.3.2. We will first review the supervised voice conversion methods, as there are many parallels to unsupervised voice conversion and because this area has traditionally been the focus of the voice conversion literature.

5.3.1 Supervised

In this section, we will give a general overview of supervised voice conversion methods and highlight the link with unsupervised voice conversion. There are two supervised scenarios which we review separately in the following paragraphs.

The first section will discuss the setting in which we do have parallel audio data, where for each \( x_{src,n} \) we have a \( x_{tgt,n} \) s.t. they are semantically identical. Next, we will discuss the scenario where we do not have parallel audio data, but instead we are in the possession of semantic labels (either transcripts \( y_n \) or phonemes \( z_{phone,n} \)) for each \( x_n \).

Voice Conversion by using a parallel corpus

Voice conversion using parallel audio data has traditionally been the focus of the voice conversion literature. In this scenario, a corpus of parallel audio is available, i.e. the data consists of pairs \( \{x_{src,n}, x_{tgt,n}\} \).

---

CHAPTER 5. VOICE CONVERSION

General setup  In the classical setup, the problem is split into two stages: (1) finding a (frame level) alignment between the source and target audio and (2) learning a function to map source audio frames to target audio frames. During inference, the mapping function is directly applied to the source audio (hence the alignment is only required for training).

Aligning the audio  Aligning the audio typically happens at the frame level, where each spectral frame $x_{\text{src},n}^t$ of the source audio is paired with a frame of the target audio $x_{\text{tgt},n}^t$. The most common alignment algorithm that is used for this purpose is dynamic time warping [83].

Learning the mapping  Next, the conversion function is learned. This function maps the source audio frames directly into target audio frames.

$$\theta^* = \arg \min_\theta p \left( x_{\text{tgt},n}^t \mid \phi \left( x_{\text{src},n}^t , s_{\text{tgt}}, \theta \right) \right)$$ (5.9)

The conversion is fitted by minimizing the likelihood (or the posterior under some prior over $\theta$) in equation 5.9 using the source-target frame pairs obtained by the alignment. Note that instead of using the spectral frames directly, more advanced features can be obtained using a basis function $\phi$. This basis function can be either static or learnable. The most common audio representation in this voice conversion setup are vocoder features, which can also be viewed as a basis function [56].

Inference  During inference, the target audio is predicted by applying the conversion function to each frame of the source audio independently:

$$x_{\text{tgt},n} \sim p \left( x_{\text{tgt},n} \mid x_{\text{src},n}, s_{\text{tgt}} \right) = \prod_{t=1}^{[x_{\text{src},n}]} p \left( x_{\text{tgt},n}^t \mid \phi \left( x_{\text{src},n}^t , s_{\text{tgt}}, \theta^* \right) \right)$$ (5.10)

Limitations  The assumption that frames are independent in equation 5.10 severely limits the model capacity of these models. In order to allow for some longer range (supra-frame) dependencies, some models update the feature learning function $\phi$ such that it incorporates information of neighboring frames (see [55] for an overview).

As phonemes typically span multiple spectral frames, frame-level conversion only allows for minor prosodic level variability (see section 5.1.2). As a result, both voice quality and target speaker similarity tend to be low [55].

Relation to ASR personalization  In the setup of ASR personalization we usually do not have access to parallel training data.

There are some works that use unsupervised methods to align single source frames with target frames in the absence of parallel data, but they do not perform well when the source and target speaker have very different voice dynamics [55]. The frame-level conversion function can also be learned in an unsupervised manner as will be described in section 5.3.2. The vocoder features $\phi$ that are developed for supervised frame level conversion are being reused in those works.

https://www.youtube.com/watch?v=GNobjtN9IU
5.3. RELATED WORK

Voice Conversion by mapping between audio and transcript.

We now review the second supervised scenario where instead of parallel audio, we have access to semantic labels. We first review how these models are implemented and conclude with the relevance it has to ASR personalization.

Though these semantic labels could be textual transcripts (y_n in figure 3.1) they usually are phonetic (z_phone in figure 3.1). We will only consider the latter case in this paragraph. Note that in order to be consistent with our framework, we will use the label z_phone for phonemes, even though this variable is considered observed in this paragraph. When we treat z_phone as an observed value, we can factorize equation 5.8 into a speech recognition model and a (speaker conditional) speech synthesis model:

\[
p(x_{tgt,n}|x_{src,n},s_{tgt}) = p(x_{tgt,n}|z_{pros_{tgt}},z_{phone_{n}})p(z_{pros_{tgt}}|s_{tgt}) p(z_{phone_{n}}|x_{src,n})
\]

We will now review how these models are usually implemented and then highlight the relation with ASR personalization.

Speech recognition model The first stage of a typical voice conversion model is to train a phoneme classifier that outputs a matrix of posterior phoneme probabilities \( p(y_n|x_{src,n}) \) for each spectral frame (known as phonetic posteriogram or PPG [27]). Generally, the phoneme classifier is a speaker-independent model to allow for many-to-one or many-to-many conversion.

Speech synthesis model The next step is to train a generative model that predicts the converted audio sample given the predicted transcript \( p(x_{tgt,n}|y_n, s_{tgt}) \). This is essentially a text-to-speech (phoneme-to-speech) problem and recurrent neural networks have been successful at this task [65]. The best performing model in the voice conversion challenge 2018 was such a model (the N10 system) [47].

The speech synthesis model can either model just a single speaker, or it can be conditional on speaker (id or embedding) to learn a one-to-many or many-to-many mapping [37]. Recent work in work in text to speech (TTS) has shown that speaker-embedding conditional TTS models can extend well to unseen speakers [37, 87, 66].

Relation to ASR Personalization In the case of ASR personalization, the audio data of our source speakers is labeled \( x_{src,i}, y_{src,i} \), while we only have unlabeled speech for our target speakers \( x_{src,j} \). As the speech recognition model (equation 5.11) is supposed to be speaker-independent, we could use it to generate transcripts for our target speakers \( y_{tgt,j} = p(y|x_{tgt,j}) \) that we can then use to train the speech synthesis model.

Though this is indeed possible for voice conversion [74] [37] [33] it is not suitable for ASR personalization. The reason being that the assumption that the speech recognition model is speaker-independent does not hold for ASR personalization. In fact, we know that the ASR system does not give accurate predictions for our target speaker, which is why we want to personalize it in the first place.

---

49 https://datashare.is.ed.ac.uk/handle/10283/3061
5.3.2 Unsupervised

We now review unsupervised voice conversion methods. We divide them in a similar manner to the supervised voice conversion methods. We first review the models that directly learn a mapping and finish with the models that learn an intermediate representation.

**Cycle-GAN for images**  The first work to directly learn a mapping in an unsupervised manner introduced a model typically referred to as cycle-GAN [91] (this work is similar to the earlier published DISCO-GAN paper [41]). The paper reported results on images, but in this paragraph, we will describe the general principle.

In the cycle-GAN approach, both a $\mathbf{x}_{\text{src}} \rightarrow \mathbf{x}_{\text{tgt}}$ mapping, modeled by network $F$, and a $\mathbf{x}_{\text{tgt}} \rightarrow \mathbf{x}_{\text{src}}$ mapping, modeled by a network $G$ are learned simultaneously. This is accomplished by combining a GAN loss with a cycle consistency loss. The cycle-consistency loss (equation 5.12) is a regular autoencoder loss that ensures that mapping from src to tgt and then back to src should yield the same sample. The GAN loss (equation 5.13) requires that the converted sample $F(\mathbf{x}_{\text{src}})$ should be indistinguishable from the true target distribution $p(\mathbf{x}_{\text{tgt}})$.

\[
L_{\text{cyc-src\rightarrow\text{tgt}}} = \mathbb{E}_{\mathbf{x}_{\text{src}} \sim p(\mathbf{x}_{\text{src}})} \left[ \| F(G(\mathbf{x}_{\text{src}})) - \mathbf{x}_{\text{src}} \|_1 \right] 
\]
\[
L_{\text{GAN-src\rightarrow\text{tgt}}} = \mathbb{E}_{\mathbf{x}_{\text{tgt}} \sim p(\mathbf{x}_{\text{tgt}})} \left[ \log D_{\text{tgt}}(\mathbf{x}_{\text{tgt}}) \right] + \mathbb{E}_{\mathbf{x}_{\text{src}} \sim p(\mathbf{x}_{\text{src}})} \left[ \log [1 - D_{\text{tgt}}(G(\mathbf{x}_{\text{src}}))] \right] 
\]
\[
\mathcal{L} = \mathcal{L}_{\text{cyc-src\rightarrow\text{tgt}}} + \lambda \left( \mathcal{L}_{\text{GAN-src\rightarrow\text{tgt}}} + \mathcal{L}_{\text{GAN-tgt\rightarrow\text{src}}} \right) 
\]

The final loss is a weighted combination of both objectives (equation 5.14). Note that it is composed of a cycle consistency loss for each (direction of) mapping, as well as a GAN loss for each (direction of) mapping. The objective is minimized with respect $F$ and $G$ while is updated for both discriminators $D_{\text{src}}$ and $D_{\text{tgt}}$.

**cycle-GAN on audio**  Two works have applied the cycle-GAN principle on audio [20, 30]. Though in essence, the method is the same as cycle-GAN, several adjustments are required to extend the work for speech. Firstly the main difference between speech and images is that speech is a variable length sequence, while the images in cycle-GAN are of a fixed size. However, because the mapping networks $F$ and $G$ are fully convolutional (U-net [64]), they can be adopted to work on variable length sequences [20, 30]. Secondly, translational invariance along both axis is a desirable property in image modeling, whereas in audio the absolute positions on the frequency axis convey important information, and models should not be invariant along this axis.

**Voice GAN**  VoiceGAN [20] is similar to cycle-GAN. In addition to the cycle-GAN losses (equation 5.14) Gao et al. [20] add another GAN loss (equation 5.16) and an identity loss (equation 5.15).

\[
L_{\text{const-src\rightarrow\text{tgt}}} = \mathbb{E}_{\mathbf{x}_{\text{src}} \sim p(\mathbf{x}_{\text{src}})} \left[ \| F(G(\mathbf{x}_{\text{src}})) - \mathbf{x}_{\text{src}} \|_1 \right] 
\]
\[
L_{\text{GAN-src\rightarrow\text{tgt}}} = \mathbb{E}_{\mathbf{x}_{\text{tgt}} \sim p(\mathbf{x}_{\text{tgt}})} \left[ \log D_{\text{tgt}}(\mathbf{x}_{\text{tgt}}) \right] + \mathbb{E}_{\mathbf{x}_{\text{src}} \sim p(\mathbf{x}_{\text{src}})} \left[ \log [1 - D_{\text{tgt}}(G(\mathbf{x}_{\text{src}}))] \right] 
\]

The authors claim that these losses are required in order to retain linguistic information, but they do not provide any evidence or ablation studies. The paper shows a qualitative evaluation consisting of only two samples on a male-female conversion task.
5.3. RELATED WORK

Multi-Discriminator CycleGAN  Hosseini-Asl et al. [30] do not modify the cycle-GAN losses. However, instead of having one discriminator, they split the frequency axis into three bins and introduce a separate discriminator for each section. They report qualitative results on a male-female conversion task and in an ablation study, they show that the three-discriminator cycle-GAN yields a higher voice quality than the regular cycle-GAN.

Limitations  Though cycle-GAN approaches are a good approach for unsupervised voice conversion, they have one big drawback that limits their use for ASR personalization. In its current form, cycle-GAN only allows learning a one-to-one mapping, as do both works that apply cycle-GAN to voice conversion. In our setup (figure 1.1), we want to be able to convert the audio of several source speakers into a single target speaker. This would mean that we have to train a cycle-GAN model for each speaker in the training data. Extending the cycle-GAN framework to be able to learn a speaker-independent mapping is highly non-trivial. Some works have extended the method to learn a conditional mapping, but the adjustments are very domain specific [9, 51]. Next to this, there are several known situations in which the cycle consistency loss and the GAN loss are not nicely behaved [11].

By learning an intermediate representation

In the absence of phonetic or textual labels, we can treat semantic content $C_n$ as a latent variable $z_n$. The generative process for voice conversion in equations 5.2 and now 5.3 become:

$$\hat{z}_n \sim p(\hat{z}_n | x_{\text{src}, n})$$

$$\hat{x}_{\text{tgt}, n} \sim p(\hat{x}_{\text{tgt}, n} | \hat{z}_n, s_{\text{tgt}})$$

These distributions define an autoencoder with a speaker conditional decoder (equation 5.18). We choose this autoencoder model for voice conversion because it allows us to learn a many-to-many mapping. There are a few earlier works that have used a conditional autoencoder for voice conversion which we will review in the next paragraphs.

VAE and VAE-WGAN: frame level conversion  Hsu et al. [32] used a variational autoencoder with a speaker dependent decoder for voice conversion. Their model is frame-based, making the same frame-independence assumption that classical supervised methods make (see equation 5.10). In a follow-up paper, they add a GAN loss to the decoder and show that this slightly improved their results [33]. Their results have high speech intelligibility, but their voice quality and target similarity are low, probably due to the frame independence assumption. Hsu et al. [32] use vocoder features whereas we train our model on spectral audio representations. Furthermore, we do not make the frame independence assumption, allowing us to learn a more flexible voice conversion model.

VQVAE: waveform level conversion  van den Oord et al. [81] applied the speaker-conditional autoencoder on waveform level. Their encoder is a simple convolutional network that is strided such that one latent is predicted for every 128 input samples. Their decoder is a powerful WaveNet model, that is conditioned globally on speaker id and locally on the latents [75]. Instead of a variational autoencoder, they use a vector quantized autoencoder where the latents that are predicted by the encoder are quantized according to a vector quantization scheme.

https://einstein.ai/research/a-multi-discriminator-cyclegan-for-unsupervised-non-parallel-speech-domain-adaptation
The audio samples that the authors present are very impressive, voice and audio quality are incredibly high as is target speaker similarity. Furthermore, voice conversion occurs at the phonetic level, leading to rich voice conversions that properly transfer accent. The authors also provide evidence that suggests that the learned latent space is speaker-independent and that it is close to the actual phonetic space. On a dataset with labeled phonemes, they show that the latent vectors predict phonemes with 49.3 % accuracy (vs 7.2 % for a random baseline).

van den Oord et al. [81] use a WaveNet decoder which is computationally expensive to train and is very slow during inference. We use a fast 2-dimensional convolutional decoder that directly predicts the sequence of spectral frames given the sequence of latents.

https://avdnoord.github.io/homepage/vqvae/#voice-style-transfer
Chapter 6
Experimental Setup

Before anything else, preparation is the key to success.

Alexander Graham Bell

This chapter will lay out the experiments that we conducted in detail. Our experimental setup can be grouped into three stages; (1) unsupervised training of a voice conversion model, (2) training an ASR model, and (3) using the voice conversion model to personalize the ASR model for a certain target speaker (this pipeline is summarized in figure 1.1).

6.1 Datasets and preprocessing

Dataset and audio preprocessing are critical to the success of voice conversion and ASR experiments. We describe the datasets that we used and the rationale for using them in section 6.1.1. Although we use different datasets for our voice conversion and for our ASR experiments, we adopt a single audio representation in all of our experiments. The problems that this posed and the settings that we settled upon are described in section 6.1.2.

6.1.1 Datasets

The dataset desiderata are quite different between ASR, voice conversion, and ASR personalization experiments. In the next paragraphs, we describe which datasets we used for which experiments and why we made these choices. An overview of the datasets and their statistics are shown in table 6.1.

VCTK

The VCTK corpus [82] is a professionally recorded dataset with speakers of all native English speaking countries. The dataset is fairly balanced in terms of gender (see table 6.1) and includes metadata about the gender, accent, and origin region of each speaker. We choose to use VCTK to train and analyze our voice conversion models since the meta-data allows us to look into the speaker dynamics of our models. Considering that the VCTK corpus is quite small and has a low amount of data per speaker, we do not use this dataset in our ASR personalization experiments.
LibriVox

LibriVox is a public domain dataset of license-free audiobooks that are read by volunteers. Because the dataset is open source, there is a great variety of speakers, recording conditions, and audio quality. LibriVox is extremely large and various datasets have been derived from it. We will discuss the derivative datasets that we used in the following paragraphs.

LibriSpeech

LibriSpeech is a subset of LibriVox where the text of the audiobooks has been aligned with the audio [58]. As a result, LibriSpeech contains speech-transcript pairs that can be used to train an ASR model. LibriSpeech contains several folds of data totaling around 1000 hours, making it a popular choice for training deep ASR models. Accordingly, we observed that LibriSpeech was the only dataset that was big enough to train our ASR model on.

Because we focus on inter-gender ASR personalization, we split LibriSpeech based on gender and train two separate ASR models, one for the male and one for the female fold.

LibriVox 2004-2679 (+)

While in ASR we want to see as many speakers as possible in order to generalize well, for our voice conversion model we are solely interested in the ability to model the target speaker. As such, we do not want the amount of available data for our target speaker to be a limiting factor in our setup and we create our own dataset for ASR personalization based on LibriVox and LibriSpeech.

Specifically, we select speakers from the LibriSpeech dataset and set aside the labeled LibriSpeech data for use in later evaluations (we can use it to measure the ASR performance of our personalized ASR models). We then proceed to download all audio data that is available for these speakers on LibriVox but that is not included in LibriSpeech and use it to train our unsupervised voice conversion model. We select one male speaker (with id 2679) and a female speaker (with id 2004) for our female-to-male and male-to-female voice conversion experiments respectively. We will refer to this dataset as LibriVox 2004-2679.

We discovered that a voice conversion model trained on the LibriVox 2004-2679 dataset does not generalize well to LibriSpeech and in that view, we decided to extend our dataset. We randomly sample 25 more speakers (13 male and 12 female) from the LibriSpeech dataset and collect their speech data from LibriVox. We name this extended dataset LibriVox 2004-2679+.

<table>
<thead>
<tr>
<th>dataset</th>
<th>size [h]</th>
<th># speakers</th>
<th>size / speaker [min] [max] [mean]</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCTK [82]</td>
<td>44.05 h</td>
<td>109</td>
<td>9.58 m 35.46 m 24.25 m</td>
</tr>
<tr>
<td>LibriSpeech [58]</td>
<td>982.42 h</td>
<td>2484</td>
<td>1.92 m 30.27 m 23.73 m</td>
</tr>
<tr>
<td>LibriVox 2004-2679</td>
<td>20.22 h</td>
<td>2</td>
<td>3.46 h 16.76 h 10.11 h</td>
</tr>
<tr>
<td>LibriVox 2004-2679+</td>
<td>221.91 h</td>
<td>27</td>
<td>48.05 m 58.57 h 8.22 h</td>
</tr>
</tbody>
</table>

Table 6.1: Datasets used in our experiments. Table shows dataset size and distribution of data over speakers.
6.2. VOICE CONVERSION EXPERIMENTS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampling rate</td>
<td>16kHz</td>
</tr>
<tr>
<td>stft window type</td>
<td>hanning</td>
</tr>
<tr>
<td>stft window size</td>
<td>320 (0.02s)</td>
</tr>
<tr>
<td>stft window stride</td>
<td>160 (0.01s)</td>
</tr>
<tr>
<td>Magnitude scaling</td>
<td>log</td>
</tr>
<tr>
<td>Pre-emphasis</td>
<td>0.97</td>
</tr>
<tr>
<td>spectrogram framerate</td>
<td>100 Hz</td>
</tr>
<tr>
<td>spectrogram frequency bins</td>
<td>161</td>
</tr>
<tr>
<td>spectrogram frequency range</td>
<td>0 - 8kHz</td>
</tr>
</tbody>
</table>

Table 6.2: Audio preprocessing settings. These settings are used in all ASR and voice conversion experiments presented in this thesis. Top rows show parameters and their values; bottom rows show resulting spectrogram properties (see section 2.1 for more information about spectrogram parameters).

6.1.2 Audio preprocessing

The way audio is represented is a difficult part of speech modeling because it confronts us with a broad range of hyperparameters that we all have to set and that can greatly impact performance if not set correctly (see section 2.1 for more details). The optimal audio representation highly depends on the specific task, and in general, speech generation requires a different representation than speech recognition.

In speech generation, the goal is to produce high-quality audio and as a result, denser signals (higher sampling rates, higher bit depths, and lower frame hop sizes) are desirable. In speech recognition, on the other hand, we can choose settings that greatly decrease the perceived audio quality, as long as the speech is still intelligible.

We choose our audio representation based on our ASR model [2]. Even though this means that the speech generated by our voice conversion model sounds of low quality, we intend to feed it as the input to our ASR model and thus it is superfluous to generate the audio in another space (of higher quality).

Because we discovered that some of the settings of [2] are sub-optimal in the training of generative models, we experimented with different settings for several of the hyperparameters, and deviated from the settings of [2] in some regards. This mainly considered settings relating to amplitude scaling and normalization. Our final audio representation parameters are shown in table 6.2.

6.2 Voice Conversion Experiments

In our voice conversion experiments, we aim to develop a good unsupervised voice conversion system that we can later use for ASR personalization.

We describe and motivate the architecture of our voice conversion model, a fully-convolutional encoder-decoder, in section 6.2.1. Encoder-decoder voice conversion models that have been reported so far have been using different types of latent regularization [31 32]. We aim to investigate the role of latent regularization in voice conversion and experiment with several types of latent spaces. The models that we use are described in section 6.2.2. In section 6.2.4 we derive some metrics that we use to perform a quantitative analysis of our voice conversion models. We derive metrics based on entropy and mutual information and describe how we can use and interpret them. We conclude with a paragraph on our training setup which details all hyperparameters used and the minutiae required in getting our setup to work.
6.2.1 Network Architecture

Our model is a fully convolutional autoencoder, with a decoder that is conditioned locally on the encoders’ latents, and globally on the speaker identity. We choose this autoencoding setup because we want our model to be able to map many voices to many for ASR personalization.

General Architecture

We use a 2-D fully convolutional architecture for both our encoder and decoder. Analogous to [33, 32, 15], our encoder outputs a latent vector \( z_t \) for each spectral frame \( x_t \) (though we also experimented with models that predict a latent every \( d \) timesteps). The convolutions are zero-padded along the time dimension to make sure that each \( z_t \) is well aligned with the input and output frames.

We want to reproduce the results of [81], but as we are working on spectrogram inputs instead of raw waveforms we cannot adopt their architecture. We do try to maintain the same characteristics as their model, which we summarize in table 6.3. We used the architecture of [86] as a starting point, and made several changes that we describe below. Table 6.3 gives a high-level summary of these two models and compares them with our own network.

Encoder

We extended the 1-D (spectral frame-wise) convolutional architecture of [33] to 2D convolutions in the interest of increasing the receptive field size of the encoder (see table 6.3). We experimented with recurrent encoders [50, 49] but found that they resulted in worse reconstruction and less stable training. We also tried to use the setup of the (lower) convolutional layers of the DeepSpeech2 model [2] but found that they led to worse reconstruction. The exact layer and filter sizes for our convolutional network are displayed in table C.1 in appendix C.

Decoder

The decoder is essential in our setup because is is the generative component of our network. It is conditioned locally on the sequence of latent vectors \( z \) and globally on the speaker id \( s \).

Experiments with autoregressive decoders

It is argued in [81] that a flexible, autoregressive decoder is essential to their setup. We did conduct experiments with two types of autoregressive models but concluded that it was more straightforward to train a convolutional decoder.

<table>
<thead>
<tr>
<th>Model</th>
<th># layers</th>
<th># parameters</th>
<th>receptive field</th>
<th>latent space</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enc dec</td>
<td>enc dec total</td>
<td>enc dec</td>
<td>freq ratio</td>
</tr>
<tr>
<td>Ours</td>
<td>5 7</td>
<td>3.41e+05 4.25e+05 7.66e+05</td>
<td>11 (110 ms) 29 (290 ms)</td>
<td>100 Hz 1</td>
</tr>
<tr>
<td>Ours+</td>
<td>5 7</td>
<td>1.39e+06 7.86e+06 9.25e+06</td>
<td>15 (150 ms) 31 (310 ms)</td>
<td>100 Hz 1</td>
</tr>
<tr>
<td>VQVAE</td>
<td>6 30</td>
<td>3.29e+05 3.42e+07 1.31e+07</td>
<td>190 (12 ms) 6139 (384 ms)</td>
<td>250 Hz 64</td>
</tr>
<tr>
<td>VAWGAN</td>
<td>5 4</td>
<td>9.24e+04 5.54e+04 1.48e+05</td>
<td>1 (25 ms) 1 (25 ms)</td>
<td>40 Hz 1</td>
</tr>
</tbody>
</table>

Table 6.3: Model architecture summary. We compare high-level statistics of our model to the two most closely related methods in the literature. Receptive field size is a measurement for the size of the conditional dependence of each latent (in the case of the encoder) or for each output sample (in the case of the decoder). Receptive field is expressed in number of samples, and is converted to a duration in milliseconds (within brackets) to facilitate comparison. Note that our model and the VAWGAN model work on spectral features whereas VQVAE uses a waveform audio representation.
6.2. VOICE CONVERSION EXPERIMENTS

<table>
<thead>
<tr>
<th>model</th>
<th>dim(z)</th>
<th>latent regularizer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>128</td>
<td>-</td>
</tr>
<tr>
<td>AE-B</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>VAE</td>
<td>128</td>
<td>KL [q(z</td>
</tr>
<tr>
<td>VQAE</td>
<td>128</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Latent space types. We experiment with four different kind of latent space regularization. The models are explained in section 6.2.2.

The first recurrent decoder that we tried was the decoder of the text-to-speech model Tacotron [86]. We found that it kept introducing artifacts such as repeating frames leading to slurry and unintelligible speech. It also failed to learn an accurate attention map. Secondly, we conducted experiments with temporal convolutional networks [44]. We found that their performance was highly dependent on teacher forcing [77], and that their behavior during training did not generalize well to inference due to error accumulation.

Our convolution decoder Given our negative results for autoregressive decoders, we proceeded with a fully convolutional decoder. We started with the 1D convolutional architecture of [33], and modified it for 2D convolution. We then did several experiments to find the optimal network architecture, drawing inspiration from the autoregressive models described in the previous paragraph. The layer and filter sizes of our final decoder architecture is shown in table C.3 in appendix C.

Global speaker conditioning We use a learnable speaker embedding to convert the one hot speaker identity vectors I_s into continuous speaker embedding vectors s_i. These embedding vectors were chosen to be of the same dimensionality as the latent space and were concatenated to the latent vectors along the channel axis.

This setup is different from [75, 79], where the one-hot speaker identity vector is injected into each layer of the conditional WaveNet model. We did experiment with injecting the speaker embedding at two stages into our model but did not find it to have any effect on reconstruction or voice conversion.

6.2.2 Latent spaces

Our two main references [81, 82] both use a different type of latent space regularization. We want to investigate what the effect of latent regularization is on voice conversion, which is why we run experiments with the same model under different latent spaces.

General setup

Though the training procedure and latent regularization are slightly different for each latent space type, we use a general loss function and we use the same encoder and decoder models in all of our experiments.

Loss function The loss function consists of a reconstruction loss \(L_x\), that is constant across latent space types, and a latent loss \(L_z\) that differs between them. The latent loss is weighted by a scalar \(\beta\) (equation 6.2). We choose the L1 distance as our reconstruction loss (equation 6.1) as it is reported to give the best results for spectrogram modeling [86, 60, 82].
Both the reconstruction loss and the latent loss are element-wise losses (over $x_i$ and $z_i$ respectively). In reality, spectral frames $x_i$ and latent variables $z_i$ are a sequence of vectors over time, and we average the losses over time and over a mini-batch of samples (using masking to deal with variable sequence length).

Note that we use a use simplified notation for all our expressions in this section, where we only define losses and functions for a single element $i$. We do not distinguish between time and frequency in this notation. For example, we say that $z_i = \text{enc}(x_i)$ while in practice $z_i$ might be dependent on multiple frames of $x$, and it might be possible that there is only one $z_j$ for every $k$ frames in $x$. We can make this simplification because our losses are element-wise and our network architecture is such that input $x$ and predictions $\hat{x}$ are always aligned.

$$L_x = ||x_i - \hat{x}_i||_1 \quad (6.1)$$

$$L(x_i, s_i) = L_x(x_i, s_i) + \beta L_z(x_i, s_i) \quad (6.2)$$

**Encoder and Decoder** All of our models are trained with the same encoder and decoder networks, which we will refer to as $\text{enc}(x_i)$ and $\text{dec}(z_i, s_i)$ respectively. During training we always decode using the source speaker id $s_i$ for each speech sample $x_i$, i.e. we never do any voice conversion during training.

**AE: Regular autoencoder**

Our baseline model is a vanilla autoencoder with no regularization over the latent space. We predict a sample by encoding the input (equation 6.3) and decoding the latent with the speaker id (equation 6.4). The loss is $L$ is identical to the reconstruction loss $L_x$.

$$z_i = \text{enc}(x_i) \quad (6.3)$$

$$\hat{x}_i = \text{dec}(z_i, s_i) \quad (6.4)$$

$$L_z(x_i, s_i) = 0 \quad (6.5)$$

**AE-B: Bottleneck autoencoder**

We add another unregularized autoencoder, but with a severely decreased latent space size to enforce a stronger informational bottleneck. The default setting for latent space size is 128, but the AE-B model has a dimensionality of 64.

**VAE: Variational autoencoder**

We use a standard implementation of our variational autoencoder, where our prior and posterior distributions are isotropic multivariate Gaussians [42]. We add two fully connected layers $f_\mu$ and $f_\sigma$ to our encoder so that it outputs the mean and variance of a Gaussian distribution respectively (equations 6.6 and 6.7). We sample a $\tilde{z}_i$ from this distribution which we use for decoding (equation 6.8 - 6.9). We can still back-propagate trough the sampling operation by using the reparameterization trick [42]. Our loss over the latent space is defined as the KL divergence between our encoder and our prior (see section 2.2 for more details on variational autoencoders). We choose a zero-centered Gaussian with unit variance as our prior and as such can use the derivation of [42] to evaluate equation 6.10.
6.2. VOICE CONVERSION EXPERIMENTS

\[ \begin{align*}
    z_i^\mu &= f_\mu(\text{enc}(x_i)) \\
    z_i^{\sigma^2} &= f_{\sigma^2}(\text{enc}(x_i)) \\
    \hat{z}_i &\sim N\left(z_i^\mu, z_i^{\sigma^2}I\right) \\
    \hat{x}_i &= \text{dec}(\hat{z}_i, s_i) \\
    \mathcal{L}_z(x_i, s_i) &= \text{KL}\left[N\left(z_i^\mu, z_i^{\sigma^2}I\right) \mid \mid N(z; 0, I)\right]
\end{align*} \] (6.6-6.10)

VQAE: Vector quantized autoencoder

Our VQAE model uses the vector quantization scheme of [81]. We initialize a codebook \( C = \{c_0, \ldots c_k\} \) that consists of \( k \) centroids that have the same dimensionality as our latent space \( Z \). We then quantize each latent vector \( z_i \) to an integer \( z_i q \) that represents the closest centroid in the codebook (see equation 6.12). The decoder looks up that index in the same codebook (equation 6.13) and uses it as a proxy \( \hat{z}_i \) for the original latent vector \( z_i \).

\[ \begin{align*}
    z_i &= \text{enc}(x_i) \\
    z_i^q &= \arg\min_j ||z_i - c_j||_2 \\
    \hat{z}_i &= Q(z_i) = c_{z_i^q} \\
    \hat{x}_i &= \text{dec}(\hat{z}_i, s_i) \\
    \mathcal{L}_z(x_i, s_i) &= ||z_i - \hat{z}_i||_2^2
\end{align*} \] (6.11-6.15)

When we want to update the parameters of the encoder with respect to reconstruction loss (equation 6.1), we need to be able to evaluate \( \frac{\delta Q}{\delta z} \) in order to apply the chain rule. Because of the argmax in equation 6.12 this gradient is undefined. Akin to [81], we approximate the gradient of \( Q \) with the identity:

\[ \frac{\delta Q}{\delta z} = I \] (6.16)

Both the encoder and decoder are updated with respect to \( \mathcal{L}_x \) using this approximate gradient to update the encoder. Only the encoder and the codebook are updated with respect to \( \mathcal{L}_z \), which is effectively pulling the encoders prediction \( z_i \) and the selected codebook vector \( \hat{z}_i \) closer together.

Note that [81] name their method vector quantized variational autoencoder because they do some experiments where they learn a prior on top of their autoencoder. They do not use this prior for their voice conversion and nor do we, which is why we name this method VQAE.

6.2.3 Training procedure

In this section, we recite all our hyperparameters settings and all specifics required to make our experiments work.
CHAPTER 6. EXPERIMENTAL SETUP

Optimization  We use the ADAM optimizer with a constant learning rate. We varied the learning rate depending on model size and used a rate of 3e-4 for the model described in section 6.2.1. We use a minibatch-size of 32 and randomly crop 5 seconds from each audio file before doing the STFT. In order to maintain a balanced distribution over speakers, we used oversampling and undersampling of our speech samples. We train our models until we observed convergence on the validation fold which usually takes between 50-200 epochs (5h - 3 days using a K80 GPU) depending on the dataset and model architecture.

Hyperparameters and tweaks  We use batchnorm in all our models and apply weight decay of 1e-5 for the smaller models and 1e-4 for the larger ones. We set $\beta$ to be 0.25 for our VQAE models and 1 for our VAE model. We clip the gradient norm to a value of 5 which especially helps the convergence for our VQAE models. We found that the number of centroids in our codebook that are actually used by the model converged to a stable value of around 60-90. Accordingly, we set the codebook size $k$ to be 256 which we always found big enough in practice. We found that codebook initialization is a very important factor in training the VQAE model. If set incorrectly, the model would only use a few values of the codebook. We got the best results with a uniform initialization in the interval $[-\frac{1}{k}, \frac{1}{k}]$.

6.2.4 Metrics

In this thesis, we aim to compare various voice conversion models with each other. Though we look at qualitative samples, we also use some metrics to facilitate an objective comparison. In the next paragraphs, we derive these quantitative metrics and describe how we use them in our analysis.

Mutual Information

Because we have no explicit loss for voice conversion, there is no reason that the decoder should learn a mapping that is dependent on speaker $s_i$. In fact, we observed that some decoders did indeed not seem to be conditional on speaker.

We use mutual information between speaker id and audio $I(x, s)$ to quantify to what extent the decoder is speaker dependent, and to see which models learn a stronger dependence. We derive a variational expression for mutual information in appendix A and prove that we can use it to compare models even if they use different latent spaces.

Conditional Entropy

We also look at which aspects of speaker, such as gender or accent, are most important in the decoder conditionally. We derive mutual information between these attributes $a$ and audio, but we show that we cannot compare these values between attributes (appendix A). Instead, we derive a lower bound for conditional entropy $H(x|a)$ and prove that we can use it to compare different speaker attributes to each other.

Reconstruction loss

Reconstruction loss is a metric for the quality of autoencoded audio, though it is not well aligned with perception. Though good reconstruction does not imply good voice conversion at all, we found that under a certain threshold it was indicative for model performance.
6.3 ASR

<table>
<thead>
<tr>
<th>Experiment</th>
<th>ASR trained on</th>
<th>ASR finetuned on</th>
<th>Model type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male to Female</td>
<td>LibriSpeech M</td>
<td>LibriSpeech M</td>
<td>baseline</td>
</tr>
<tr>
<td></td>
<td>VC[LibriSpeech M → 2679 (M)]</td>
<td>src conversion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VC[LibriSpeech M → 2004 (F)]</td>
<td>tgt conversion</td>
<td></td>
</tr>
<tr>
<td>Female to Male</td>
<td>LibriSpeech F</td>
<td>LibriSpeech F</td>
<td>baseline</td>
</tr>
<tr>
<td></td>
<td>VC[LibriSpeech F → 2004 (F)]</td>
<td>src conversion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VC[LibriSpeech F → 2679 (M)]</td>
<td>tgt conversion</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5: Overview of ASR personalization experiments. Table shows what ASR data each model was trained and finetuned on. We use VC[dataset → s] to denote that a voice conversion model has been used to convert the audio of dataset into the voice of speaker s. For each finetuning experiment we combine the data that the ASR will be finetuned on with the data that the ASR model was originally trained on because it led to better performance.

Latent loss

The VAE and VQAE models both have a latent loss $L_z$. For the VAE this loss has a nice probabilistic definition which it does not for the VQAE. In our result section, we show some evidence that suggests that latent losses can be indicative for dataset perplexity.

6.3 ASR

We use DeepSpeech2 [2] as our ASR model. This section will summarize the procedure of training the ASR model while the exact experiments that we conduct and the datasets that we used are described in section 7.3. We use SGD with Nesterov momentum of 0.9 to optimize the model. We use a learning rate schedule where we start with a learning rate of 1e-3 and anneal with 1.05 after every epoch. We train the model for 30 epochs (4 days using 8 K80 GPUs). We use a minibatch-size of 128, and clip the gradient norm at 400. We use batchnorm and we apply weight decay of 1e-3.

6.4 ASR Personalisation

In the previous sections, we described how we train our voice conversion and ASR model. In this section, both are united as we use our voice conversion model to personalize the ASR model. We run two main experiments, one for male-to-female and one for female-to-male personalization.

6.4.1 Experiments and baselines

We now describe the exact experiments that we run and the baselines for each experiment. An overview is listed in table 6.5.

**Target conversion model** The target conversion model is the ASR model that is personalized for our target speaker. We use a speaker of the opposite gender to ensure that the possible gain is as large as possible.

**Baseline model** Our vanilla baseline is an ASR model that has not seen any voice converted data. To hold all other factors constant we use the exact same personalization procedure as for
the target conversion model. Instead of using voice converted data, we use the original audio
data to finetune the ASR model.

**Source conversion baseline**  If the target conversion model shows improvement compared to
the baseline model we cannot be certain that this improvement is specific to the target
speaker personalization. For this reason, we add another baseline in which we finetune the ASR model
for a different speaker. We use a speaker that is in the source dataset (e.g in the male-to-female
experiment we convert to a male speaker) which is why we call this the source conversion baseline.
We use the exact same personalization procedure as for the target conversion.

### 6.4.2 Personalization procedure

**Voice conversion data generation**  For all of the experiments that involve voice conversion
(VC in table 6.5) we use the same voice conversion model which is the VQAE model trained
on the LibriVox 2004-2679+ dataset. We use the voice conversion model to create audio in the
target speakers voice and combine the speech with the original transcripts to obtain an ASR
dataset (figure 1.1).

**Finetuning the ASR model**  When finetuning the ASR model we add the finetuning dataset
to the dataset that the ASR model was originally trained on. For the voice conversion experi-
ments, this means that half the data is voice converted while the other half is the original audio.
For the baseline model, it means that we do duplicate the dataset. Note that it is important
that we actually do this because of the learning rate schedule. We finetune the ASR model for
30 more epochs using the same learning rate and learning rate schedule that we used to train
the ASR model (which is described in section 6.3).
Chapter 7

Results

If we knew what it was we were doing, it would not be called research, would it?

Albert Einstein

In this chapter, we present our results. We start with a thorough analysis of our voice conversion models in section 7.1, where we show that the quantized and variational autoencoder do learn a voice conversion model, which is mainly limited to gender-based conversion. We then continue in section 7.2 to investigate how our voice conversion model performs on the ASR personalization dataset. We conclude that it does not generalize well and we investigate the causes. We still proceed to use our voice conversion model for ASR personalization and present the results in section 7.3. We show that our voice conversion method can help to increase general ASR performance, but it fails to increase performance specifically for our target speaker.

7.1 Voice Conversion

We will now evaluate our voice conversion model. Recall from section 5.2 that our goal is to achieve a high target speaker similarity and speech intelligibility, while we are indifferent to the voice quality of our generated samples as we will feed the output to an ASR model.

We perform a qualitative (section 7.1.1) and a quantitative (section 7.1.2) analysis, and we look at the learned speaker embedding (section 7.1.3). Our conclusions are summarized in section 7.1.3. All results reported in this section are from models that are trained on the VCTK dataset.

7.1.1 Qualitative Results

Tables 7.1 and 7.2 show spectrograms of speech samples generated by our voice conversion model. These samples are handpicked to represent typical behavior of the models. We encourage the reader to listen to these samples online when reading this section.

Reconstruction quality The first row in table 7.1 and 7.2 shows the reconstruction results ($s_{tgt} = s_{src}$) for the source audio sample $g1$. 

[http://masterthesis.tivaro.nl/audiosamples/#table-of-contents]
### Table 7.1: Converted male audio samples.

Presented samples originate from the test fold of the VCTK dataset. Different latent space types are shown in each column (see Table 6.4 for acronym definitions). The GT column shows a ground truth sample for each target speaker. The sample in the first row is used as the source sample for all conversions. Annotations in the spectrograms are used in the interpretation in section 7.1.1.

Transcript of the input sample: "It will include the Black Watch, the Royal Highland regiment."

Listen to these samples here: [http://masterthesis.tivaro.nl/audiosamples/#male-conversion](http://masterthesis.tivaro.nl/audiosamples/#male-conversion)

<table>
<thead>
<tr>
<th>src</th>
<th>tgt</th>
<th>GT</th>
<th>AE</th>
<th>AE-B</th>
<th>VAE</th>
<th>VQAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>p345 (M)</td>
<td>p345 (M)</td>
<td><img src="image1" alt="g1" /></td>
<td><img src="image2" alt="a1" /></td>
<td><img src="image3" alt="b1" /></td>
<td><img src="image4" alt="v1" /></td>
<td><img src="image5" alt="q1" /></td>
</tr>
<tr>
<td>p345 (M)</td>
<td>p304 (M)</td>
<td><img src="image1" alt="g2" /></td>
<td><img src="image2" alt="a2" /></td>
<td><img src="image3" alt="b2" /></td>
<td><img src="image4" alt="v2" /></td>
<td><img src="image5" alt="q2" /></td>
</tr>
<tr>
<td>p345 (M)</td>
<td>p277 (F)</td>
<td><img src="image1" alt="g3" /></td>
<td><img src="image2" alt="a3" /></td>
<td><img src="image3" alt="b3" /></td>
<td><img src="image4" alt="v3" /></td>
<td><img src="image5" alt="q3" /></td>
</tr>
</tbody>
</table>
### Table 7.2: Converted female audio samples.

Presented samples originate from the test fold of the VCTK dataset. Different latent space types are shown in each column (see Table 6.4 for acronym definitions). The GT column shows a ground truth sample for each target speaker. The sample in the first row is used as the source sample for all conversions. Annotations in the spectrograms are used in the interpretation in Section 7.1.1.

Transcript of the input sample: "I, in the meantime, had been far from chilling."

Listen to these samples here: [http://masterthesis.tivaro.nl/audiosamples/#female-conversion](http://masterthesis.tivaro.nl/audiosamples/#female-conversion)

<table>
<thead>
<tr>
<th>src</th>
<th>tgt</th>
<th>GT</th>
<th>AE</th>
<th>AE-B</th>
<th>VAE</th>
<th>VQAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>p265</td>
<td>p265</td>
<td>g1</td>
<td>a1</td>
<td>b1</td>
<td>v1</td>
<td>q1</td>
</tr>
<tr>
<td>(F)</td>
<td>(F)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p265</td>
<td>p248</td>
<td>g2</td>
<td>a2</td>
<td>b2</td>
<td>v2</td>
<td>q2</td>
</tr>
<tr>
<td>(F)</td>
<td>(F)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p265</td>
<td>p345</td>
<td>g3</td>
<td>a3</td>
<td>b3</td>
<td>v3</td>
<td>q3</td>
</tr>
<tr>
<td>(F)</td>
<td>(M)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*kHz*
The first thing we note is the low perceived quality of our audio representation. Listening to the groundtruth samples (spectrograms $g_1$, $g_2$ and $g_3$) reveals that they sound very robotic and noisy. This is due to the fact that we use a lossy magnitude spectrogram representation that does not contain phase information. More information about spectrogram representations can be found in section 2.1.

We observe that the AE model learns a near-perfect reconstruction, while the AE-B does not learn to produce any speech at all. This is illustrated by the fact that the groundtruth sample ($g_1$) is indistinguishable from the reconstruction by the AE model ($a_1$). The AE-B model (whose reconstruction is shown in $b_1$) failed to reconstruct the original sample. While failure modes of audio generative models in the literature often resemble babbling [79], the reconstructed sample of the AE-B model is unintelligible and does not sound like human speech.

The VAE and VQAE model both generate noisy yet intelligible reconstructions that preserve most of the voice characteristics. Note that the reconstructions of the VAE model ($v_1$) and the VQAE model ($q_1$) both sound noisier and more robotic than the groundtruth sample ($g_1$), but still preserve the phonetic content. If we compare the spectrograms we can see that a lot of the finer detail of the voice has disappeared, but the global harmonic patterns are properly reconstructed. The male reconstruction (table 7.1) seems to be lossier than the female reconstruction (table 7.1).

In some of the reconstructions, the VQAE model introduces minor distortions. An example of this is shown in table 7.2-$q_1$. The section highlighted in the green rectangle contains spectral frames that look natural, but upon closer inspection are substantially different from the original (table 7.2-$g_1$). When listening to this sample, the stress of this sentence appears to have changed. These "substitutions" occurred periodically for the VQAE model and resulted in slight variations in perceived in intonation.

**Inter-gender voice conversion**  Inter-gender voice conversions are shown in the third row of table 7.1 and 7.2. The input source sample ($g_1$) is converted by decoding with a different target speaker id. The target speaker is of the opposite gender, and the specific target speaker id is shown in the $tgt$ column. An example audio fragment of the target speaker is given in spectrogram $g_3$.

The AE model did not learn to output a different sample when a different speaker id was provided. This can be seen by comparing the converted sample by the AE model ($a_3$) with its reconstruction ($a_1$). For the AE-B model, the conversion ($b_3$) sounds as unintelligible as the reconstruction ($b_1$).

The VAE and VQAE model both were able to capture the harmonic distortion necessary to convert the gender. For the male to female conversion (table 7.1) we saw an expansion of the harmonics for both the VAE ($v_1$ vs $v_3$) and the VQAE ($q_1$ vs $q_3$) model. For the female to male conversion (table 7.2), a contraction of the harmonics can be observed for both models ($v_1$ vs $v_3$ and $q_1$ vs $q_3$ respectively). When listening to these spectrograms we mainly perceive a change in pitch, while phonemes and intonation still closely resemble the original audio sample ($g_1$).

Both the VAE and VQAE model displayed some cases of incomplete voice conversion. An example can be seen in the female to male conversion in table 7.2. The yellow rectangles in spectrograms $v_3$ and $q_3$ indicate parts of the spectrogram where the harmonics have not been compressed. As a result, the voice sounds inconsistent across the spectrogram.

The VAE model tended to produce samples with very linear formants that sounded monotone, whereas the VQAE produced samples with more variation in intonation that sounded more natural. A prime example of this is the male to female conversion by the VQAE model, shown in table 7.1-$q_3$. The green rectangle shows the part of the spectrogram which demarks the rise
7.1. VOICE CONVERSION

in intonation at the phoneme "a" in "watch". Listening to this sample reveals the richness in intonation which is in sharp contrast with the monotone audio of the same sample converted by the VAE model (table 7.1-v3). A more subtle manifestation of the same phenomenon is highlighted by the green rectangle in table 7.2 spectrogram q3.

Intra-gender voice conversion  Samples for intra-gender voice conversion are shown in the second row in table 7.1 and 7.2. The target speaker has the same gender as the source speaker, and an example spectrogram of the target speaker is shown in spectrogram g2.

In correspondence with our results for inter-gender voice conversion, the AE model did not learn a conversion (a2 vs a1) and the AE-B model produced unintelligible speech (b2).

For the VAE and the VQAE model, the voice conversion is very subtle, and we do not observe the big changes in harmonic pattern that we saw for inter-gender voice conversion. For male-to-male conversion (table 7.1) the harmonics produced by the VQAE model are slightly more compressed in the converted sample (q1) than they are in the reconstruction (g1), as indicated by the green line. Apart from this, we mainly observe changes in the distribution of background energies. For example, the converted examples v2 and q2 in table 7.1 have a higher density of frequencies below 4 kHz (red line) than the reconstructions v1 and q1. If we listen closely, we can hear that the voices do sound somewhat different for the reconstruction and the conversion.

Conclusion  Taken altogether, we conclude that the AE model does yield very accurate reconstruction but does not exhibit any voice conversion. The AE-B model is unusable in general as its speech samples are unintelligible. Both the VAE and the VQAE model display subpar reconstruction of the original signal compared to the AE, but they are able to perform voice conversion when conditioned on a different target speaker. This shows that we can use the VAE and VQAE model in the context of ASR personalization.

Inter-gender voice conversion is the most apparent with changes in harmonics patterns according to the gender of the target speaker. Intra-gender voice conversion is a lot more subtle, with changes in pitch but relatively unchanged harmonics. In general, both the VAE and VQAE model show voice conversion that is limited to the level of voice texture, where phonetic information and even intonation remains unaltered between reconstructed and converted audio. The harmonics of converted samples by the VAE model are very linear, while those of the VQAE model show more variation. As a result, the intonation in the VQAE model is more natural. Because of this, the VQAE model shows more promise for use in ASR personalization, although the differences with the VAE model are small.

7.1.2 Quantitative results

The quantitative results are shown in table 7.3. We will first compare the extent of voice conversion between each of the different latent space types, and we finish by looking at the importance of various aspects of speaker in the learned voice conversion. Refer to appendix A for derivations of the metrics that we use in this section.

Latent space type

Reconstruction The average reconstruction loss is equal to the conditional entropy lower bound $H_b(x|s)$ that we derive in section A.2 of appendix A.

When we compare the conditional entropy between the different models (each row in table 7.3 represents a model), we see that it is ranked as follows:

$H_b_{AE}(x|s) < H_b_{VAE}(x|s) < H_b_{VQAE}(x|s) < H_b_{AE-B}(x|s)$  \hspace{1cm} (7.1)
This ranking is in line with our qualitative results; the AE-B showed the worst reconstruction, while the reconstruction of the AE model was the most accurate. The reconstructions of the VAE and VQAE models are intermediate. The VAE model has a slightly better reconstruction than the VQAE model, which is not something we directly observed in our qualitative analysis. However, we did see artifacts in the samples generated by our VQAE model which might explain the difference.

**Mutual Information** Mutual information between speaker and audio tells us to what extent the decoder is conditional on speaker. Even though the speaker id is given as input to the decoder in all of our models, the decoder could just learn to ignore it. When we look at the mutual information between speaker and speech, we observe the following ordering:

\[
I_{AE}(x, s) < I_{AE-B}(x, s) \ll I_{VQAE}(x, s) < I_{VAE}(x, s)
\] (7.2)

This observation is in line with our qualitative results, where the models that had the strongest voice conversion were the VAE and VQAE model, while the AE and AE-B model did not show voice conversion.

**Speaker attributes**

In our qualitative evaluation, it seemed that voice conversion was mainly dependent on gender. We now look at the conditional entropy of \(x\) given gender, and compare it with other conditional entropies, to see if this trend is general. We can see in Table 7.3 that the following inequality holds for all models:

\[
H_b(x|s) < H_b(x|\text{gender}) < H_b(x|\text{accent}) \preceq H_b(x)
\] (7.3)

This confirms that our decoder is more dependent on gender than on accent.

Furthermore we observe the following approximate equalities for the AE and AE-B model:

\[
H_{AE(\cdot,B)}(x|\text{gender}) \approx H_{AE(\cdot,B)}(x|\text{accent}) \approx H_{AE(\cdot,B)}(x)
\] (7.4)

Or in other words:

\[
I_{AE(\cdot,B)}(x, \text{gender}) \approx 0
\] (7.5)

\[
I_{AE(\cdot,B)}(x, \text{accent}) \approx 0
\] (7.6)

|     | \(I(x, s)\) | \(I(x, \text{gender})\) | \(I(x, \text{accent})\) | \(H_b(x|s)\) | \(H_b(x|\text{gender})\) | \(H_b(x|\text{accent})\) | \(H_b(x)\) |
|-----|-------------|----------------|----------------|-------------|----------------|----------------|-------------|
| AE  | 2.43e-04    | 5.41e-05       | 4.17e-05       | 5.51e-02    | 5.54e-02       | 5.54e-02       | 5.54e-02    |
| AE-B| 4.25e-04    | 2.07e-04       | 1.06e-05       | 1.22e-01    | 1.23e-01       | 1.23e-01       | 1.23e-01    |
| VAE | 2.20e-03    | 1.62e-03       | 4.38e-04       | 8.19e-02    | 8.25e-02       | 8.36e-02       | 8.41e-02    |
| VQAE| 2.15e-03    | 1.25e-03       | 3.65e-04       | 1.04e-01    | 1.05e-01       | 1.06e-01       | 1.06e-01    |

Table 7.3: **Mutual Information and entropy per latent space type**. \(I\) is mutual information, \(H_b\) is a lower bound on entropy (see appendix A for derivations). Reported results are on the test fold of the VCTK dataset. Each row presents a different latent space model, refer to table 6.4 for acronym definitions. A derivation of these metrics is given in appendix A.

52
Which confirms that the decoder in these models has almost no dependence on speaker.

Finally, we note that if we order the speaker attributes by their mutual information, we yield a similar ordering to equation 7.3 (At least for the VAE and VQAE model, where the mutual information is significantly larger than 0). This suggests that mutual information is not bounded by speaker attribute (see section A.4 in appendix A for more information).

### 7.1.3 Speaker Embedding

Our decoder model is conditioned on speaker ids through a learnable speaker embedding $s$ (see our method in section 6.2.1 for details). We visualize the learned embeddings in this section and show that they confirm the previous results.

The learned speaker embeddings show a clear separation between the two genders for all models except for the AE model. Speaker accent, on the other hand, is not separated by any of the models. To visualize the speaker embedding matrices we reduce their dimensionality using PCA and t-SNE. We then plot each speaker in our dataset on this coordinate system in figure 7.1a (using PCA) and 7.1b (using t-SNE). In the first row of these figures, the speakers are colored by gender. We see that gender is not separated for the AE model (figure 7.1a-I and 7.1b-I) while the other models seem to separate gender fairly well. When we look at the second row, where the speakers are colored by accent, we do not see a meaningful clustering for any of the models.

![PCA speaker embedding.](image1)  ![t-SNE speaker embedding.](image2)

**Figure 7.1: Speaker embedding visualizations.** The speaker embeddings are plotted for each model (see table 6.4 for acronym definitions). The dimensionality of the embeddings is reduced from 32 to two, and each speaker in the VCTK dataset is plotted along those two axes. The first row is colored by gender and the second row is colored by accent. Plots are rotated manually to facilitate easy comparison.

### Conclusion

Our results show that only the VAE and VQAE models learn a speaker-conditional decoder that can be used for voice conversion. The AE model does learn to reconstruct the audio better but its decoder is independent of speaker, the AE-B model does not learn any realistic generative model at all.

The voice conversion is prosodic, and only changes the style of the voice while maintaining a similar phonetic sequence and intonation. As a result, the voice conversion is strongest for inter-gender pairs. The converted speech generated by the VAE model sounds monotonous while the VQAE model has slightly more variation in intonation in its converted samples.
CHAPTER 7. RESULTS

7.2 Voice conversion for ASR personalization

In the previous section, we successfully trained a voice conversion model on the VCTK dataset. We wanted to use the same model for ASR personalization, but we discovered that doing voice conversion on the ASR dataset is a lot more difficult. We discuss the problems that we faced and motivate the choice of our final model for ASR personalization in section 7.2.1. In section 7.2.2 we analyze how well this model performs on the LibriSpeech dataset, and conclude that it does not generalize well.

7.2.1 Training datasets for ASR personalization

In our ASR personalization scheme (summarized in figure 1.1) we propose to train the voice conversion model on the combination of the (audio in the) ASR dataset and the audio data that we have for our target speaker. In our setup, this would mean that we train our voice conversion model on the male and female folds of LibriSpeech.

Unfortunately, we found it very difficult to train a sufficiently good voice conversion model on the LibriSpeech dataset. Due to its open-source nature, LibriSpeech has a large variety of speakers, speech quality and recording conditions. It seemed really difficult for our decoder to learn to accurately predict this wide a variety of speech samples. Both reconstructions and converted samples were very blurred, and as a result, they sounded quite unintelligible. Furthermore, the harmonic patterns seemed to change multiple times within a single audio sample, making it sound like the speaker suddenly changed. We observed these artifacts for any model architecture and latent space type that we trained on LibriSpeech.

We tried to eliminate these problems in the LibriVox2004-2679+ dataset, by decreasing the number of speakers and increasing the amount of data per speaker (see section 6.1.1 for more details). However, we observed the same problems occurred when trying to train models on this dataset.

As a first iteration, we decided to continue our ASR personalization experiments with a voice conversion model trained on the LibriVox-2004-2769 dataset. We chose the VQAE model because it had a slightly better target speaker similarity and a more natural voice than the VAE model (see section 7.1.1).

7.2.2 Generalization to the ASR dataset

We use a VQAE model trained on the LibriVox-2004-2769 dataset for our ASR personalization experiments. As this dataset only contains data for two speakers (the female with id 2004 and the male with id 2769) we anticipate that our model might not generalize well to the LibriSpeech dataset. Manual inspection of the results shows that there are indeed a lot of unintelligible samples in the converted ASR data. However, it is difficult to separate the bad conversions from the good ones.

We hypothesize that bad conversions occur because the input sample $x_i$ is out of distribution with the training data. As a result, we expect that the latents $z_i$ for these samples will also be out of distribution and this would mean that we could use the latent loss $L_z$ as a proxy for conversion quality. For both the VAE and the VQAE model, the latent loss $L_z$ can be viewed as a proxy for latent probability. For the VAE model, the latent loss is defined as KL-divergence between $q(z_i|x_i)$ and $p(z_i|x_i)$. Although this is not the case for the VQAE model, the latent loss still is a proxy for "perplexity" because the distance between $z_i$ and $\hat{z}_i$ is minimized during training, so high values indicate unlikely latents.

High latent loss seemed to be indicative for samples with bad voice conversion quality. Table 7.4 shows the mean latent losses for the voice conversion and ASR datasets. We can indeed see
7.2. VOICE CONVERSION FOR ASR PERSONALIZATION

<table>
<thead>
<tr>
<th>VC data</th>
<th>ASR data</th>
</tr>
</thead>
<tbody>
<tr>
<td>(LibriVox-2004-2769)</td>
<td>(LibriSpeech)</td>
</tr>
<tr>
<td>Male fold</td>
<td>0.03323</td>
</tr>
<tr>
<td>Female fold</td>
<td>0.03320</td>
</tr>
</tbody>
</table>

Table 7.4: Latent losses on the voice conversion and ASR dataset. Losses are shown for the VQAE voice conversion model trained on the LibriVox-2004-2769 dataset. The latent loss for the VQAE model is defined in equation 6.15. Losses are averaged over all data and all timesteps. The VC data column shows the losses on the test fold of the voice conversion dataset while the ASR data column reports the losses on the train fold of the ASR dataset (we use the train fold here because we do voice conversion on this fold). Both datasets are separated into a male and female fold which are shown per row.

that for both the male and the female fold the latent losses on LibriSpeech are at least a factor 2 higher than the losses on the LibriVox-2004-2769 dataset, which suggests that our model does not generalize well to LibriSpeech. We further investigate this relation in table 7.5. We show samples that are selected to have a low, median and high latent loss based on the percentile of their loss value in the dataset (we note that the latent loss values were nicely lognormally distributed). Although we do not find big differences in intelligibility between low and median z losses, we do notice that samples with a high z loss are often unintelligible, as shown by the samples in table 7.5.

Within a sample, the latent loss seemed to be indicative for unintelligible parts of the speech. The penultimate row in table 7.5 shows the latent loss over time. We note that especially for the high z sample, the peaks in the latent loss correspond well to artifacts in the converted speech sample (that render those parts unintelligible).
<table>
<thead>
<tr>
<th></th>
<th>low $z$</th>
<th>median $z$</th>
<th>high $z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>loss $z$ mean</td>
<td>0.194</td>
<td>0.497</td>
<td>1.208</td>
</tr>
<tr>
<td>loss $z$ percentile</td>
<td>9.9 %</td>
<td>50.0 %</td>
<td>90.1 %</td>
</tr>
</tbody>
</table>

**Table 7.5: Latent losses for female to male voice conversion samples.** The speech samples in this table were selected to have a low, median or high latent loss $L_z$ (averaged over all timesteps per sample) as defined by their percentile in the dataset distribution. Source samples are selected from the LibriSpeech male train fold, and the VQAE model trained on the LibriVox-2004-2679 dataset was used to convert the samples to the target speaker (with id 2004). The penultimate row shows the latent loss values per timestep. Note that latent loss seems to be a good indicator for speech intelligibility. Listen to these samples here: [here](http://masterthesis.tivaro.nl/audiosamples/#z-loss-male-to-female) and here for female to male conversions: [here](http://masterthesis.tivaro.nl/audiosamples/#z-loss-female-to-male).
7.3 ASR Personalization

Even though we observed that our voice conversion model does not generalize well to LibriSpeech, we investigate whether using them in our ASR personalization setup can still lead to increased ASR performance. We conclude that we can use our voice conversion model to increase the general ASR performance, but that it does not succeed in personalizing the ASR model for a specific target speaker.

7.3.1 Male to Female

Table 7.6a shows the average CER scores for the baseline and personalized ASR models that were trained on only male ASR data. As expected, the error rate for females (row all F) is higher than the error rate for males (row all M) for the baseline method. Contrary to our expectations, the lowest error rates for our female target speaker (2004) are achieved by the 2769(male)-personalized ASR model, and not by the 2004(female)-personalized model. The same result is true for the error rates of the target gender (female), where the best performance is again achieved by the 2769-personalized model. We also note that both voice conversion models greatly improve the overall ASR performance, with over 10% decrease in error rate. This can be seen clearly in figure 7.2a-I.

7.3.2 Female to Male

The average CER scores for the ASR model trained on female data are shown in table 7.6b. As expected the females have a lower error rate than the males for the baseline method. We can see that 2769(male)-personalized ASR model does reach the lowest error rates for both the target speaker (2679) and the target gender (male) compared to both baselines. However, unexpectedly we find that 2769(male)-personalized also achieves the lowest error rates for female ASR.

7.3.3 Conclusions

Although the best performing ASR model is a personalized model for both male and female ASR, we failed to increase error rates specifically for our target speakers. Interestingly, the female to male voice conversion worked the best in all cases, while in section 7.2.2 we showed that our female to male conversion is a lot worse than our male to female conversion (see table 7.4). This suggests that the main effect of voice conversion on ASR model is achieved by adding noise and so regularizing the model. For a more finegrained overview of the change in speaker dynamics.

<table>
<thead>
<tr>
<th>ASR model</th>
<th>(a) trained on M data</th>
<th>(b) trained on F data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>baseline to 2004 (F)</td>
<td>to 2769 (M)</td>
</tr>
<tr>
<td>2004 (F)</td>
<td>11.03</td>
<td>7.00</td>
</tr>
<tr>
<td>2769 (M)</td>
<td>19.51</td>
<td>7.93</td>
</tr>
<tr>
<td>all F</td>
<td>30.05</td>
<td>20.10</td>
</tr>
<tr>
<td>all M</td>
<td>26.36</td>
<td>13.81</td>
</tr>
<tr>
<td>all</td>
<td>27.76</td>
<td>16.61</td>
</tr>
</tbody>
</table>

Table 7.6: ASR scores of baseline and personalized ASR models. (a) shows CER scores for ASR models that were trained on female data and (b) shows scores for ASR models that were trained on male data only. Green and orange rows highlight the scores for the target speaker while blue and pink rows highlight the gender of the target speaker (which was opposite to the gender of the ASR training data).
for each of the ASR models we refer to figure 7.2a and 7.2b and to the speaker histograms in appendix B.

7.3.4 Qualitative

(a) Male to 2004 (female) personalization. Target speaker is displayed as a yellow cross.

(b) Female to 2769 (male) personalization. Target speaker is displayed as a green cross.

Figure 7.2: ASR Improvement plots. Plots show the change in error rate for each speaker (in the LibriSpeech ASR test fold) after ASR personalization. Each speaker is presented as a dot in a two-coordinate axis system. The y-axis shows the error rate for the ASR model that is finetuned for the target speaker, while the x-axis shows the error rate for two different ASR baselines. In plot I, the x-axis shows the performance of a baseline (non-personalized) ASR model and in plot II the x-axis shows performance for an ASR model that is personalized for a speaker whose gender is opposite to that of the target speaker. Black line denotes $x = y$, indicating equal performance. Speakers are colored by gender, and the two target speakers are plotted in green (2769, male) and yellow (2004, female). See section 6.4.1 for more information about the ASR models and baselines.
Chapter 8

Conclusion

In this thesis, we proposed to use voice conversion as a method for ASR personalization. In this final chapter, we summarize our conclusions (in section 8.1) and discuss limitations and suggestions for future work (in section 8.2).

8.1 Conclusions

8.1.1 Voice Conversion

In our voice conversion experiments, we trained a convolutional autoencoder with a decoder that is conditioned on the speaker id. We did experiment with different types of regularization of the latent space.

We showed that latent space regularization is required in order to learn a model with a dependence on the speaker, and that the specific type of regularization is important. The unregularized autoencoder did learn to properly reconstruct the audio, but it did not learn any voice conversion. The bottleneck-autoencoder on the other hand did not even learn a good generative model as it produced highly unintelligible audio that only remotely sounded like speech. The only models that learned a good voice conversion were the variational and quantized autoencoder.

Although the performance of these two models was very similar, there were subtle differences in the audio that they produced. The variational autoencoder tended to produce very monotone sounding samples, while the quantized autoencoder did produce samples with more variety in intonation. However, the quantized autoencoder did introduce some minor artifacts, that sporadically rendered parts of the speech slightly less intelligible. In general, the voice conversion that was learned by both models was limited to changes in pitch and in patterns of the harmonics. The timing, accent, and intonation of the converted speech samples remained very close to the original.

We derived expressions for mutual information and conditional entropy between speech and speaker and demonstrated that they were very descriptive in practice. As expected, mutual information was an indicative metric for speaker conditionally and could be compared across different latent space types. The values of conditional entropy of speech given speaker attributes confirmed that gender was modeled well by our voice conversion models while accent was not.

We discovered that training our voice conversion model became more difficult as the speaker variability in the dataset increased. We found that it was relatively easy to train a model on the VCTK dataset, which is rather small in size, but has very consistent (and high quality) recording conditions. In contrast, we failed to train any model on LibriSpeech that was able to produce
samples of a reasonable quality. Even training on a smaller subset of 27 speakers in LibriSpeech with a large amount of data per speaker took us elaborate effort. None of the models that we trained on various datasets generalized well to LibriSpeech during inference. In conclusion, we failed to train a model that we could use for voice conversion on LibriSpeech.

8.1.2 ASR Personalization

Our voice conversion models did not generalize well to LibriSpeech (our ASR dataset) and as such, we failed to use them for ASR personalization. However, we did show that we could still use our voice conversion models to increase the general ASR performance for all speakers.

8.2 Limitations and Future Work

8.2.1 Latent space regularization

In this study, we compared autoencoders with different kinds of latent regularization (unregularized, bottleneck, variational and quantized), and showed that they have different properties of voice conversion.

A limitation of this study is that it is difficult to make a fair comparison between the models with different types of latent regularization. Specifically, it is difficult to differentiate the type of regularization from the amount of regularization. Because of the additional regularization in our quantized and variational autoencoder, the dimensionality of the latent space is not a sufficient statistic to express the effective bottleneck size. Instead, the conditional entropy $H(z|x)$ gives a more accurate measure. From an information theoretic perspective, the conditional entropy can be seen as the expected number of bits (or nats) that need to be used (on average) to transmit the conditional latents produced by the encoder (using entropy or arithmetic coding). The conditional entropy is intractable because it contains an integral over $z$. It can be approximated in some cases [90], but it will most likely involve the prior distribution $p(z)$ which is not defined for our deterministic autoencoders. As such, we are not able to compare the conditional entropy between models with different types of latent space.

Another area for improvement is the quantization method used in our quantized encoder. We use the vector quantization method of [80] because we want to be as close as possible to their setup. A downside of this scheme is that index collapse can occur, in which all of the encoded latents $z$ get assigned to the same codebook vector $z_q$, rendering the quantized latents uninformative [38]. Although we did not observe this often during our experiments, it did happen more frequently as we increased the size of our models. The main problem with this type of vector quantization is that only the selected codebook vector $c_{z_q}$ receives a gradient, even if there are multiple codebook vectors with an identical distance. Alternative quantization methods that do not have this property include Gumbel softmax sampling [36] or soft-to-hard vector quantization [1]. How these different quantization methods affect voice performance is a question that should be answered by future research.

8.2.2 Model architecture

The results that we present in this work only cover a single model architecture; a 2-dimensional fully convolutional autoencoder. We conducted experiments with different layer configurations and with different model architectures that are not reported. In this section we reflect on our architecture and discuss which aspects of network architecture are important for voice conversion.
At the moment of writing, the best voice conversion results are reported by van den Oord et al. [81]. The main difference between their setup and ours, is that their decoder of is an autoregressive WaveNet. This WaveNet differs from our decoder in three ways: (1) the audio is represented as a waveform instead of a spectrogram, (2) their decoder is autoregressive, (3) their decoder has roughly 4 times as many parameters as the largest model that we successfully trained (and about 100x as many parameters as the model that we report our results on). It is difficult to specify how much each of these differences contributed to the gap between our and their results, but it is likely that the autoregressive decoder is the key difference. The biggest advantage of an autoregressive decoder over a joint decoder is that it can deal substantially better with multimodal distributions. A joint generative model would predict the mean of all the modes while the autoregressive model would stochastically sample from one of the modes during inference.

Assuming that an autoregressive decoder is a critical component for good voice conversion, the question arises what a good autoregressive decoder to model spectrogram audio would be. The WaveNet architecture is heavily tied to the properties of waveforms and it is unclear how to modify the network for usage on spectrogram data. Not a lot of research has been conducted in the area of autoregressive autoencoding on spectrograms, likely due to their low audio quality. There are papers that use LSTM autoencoders [53, 52] but their main goal is representation learning, and they do not report on audio reconstruction. Instead, we could look at supervised generative models for speech. The best example is Tacotron [86], a text to speech synthesis model that makes spectrogram predictions. Tacotron consists of a text encoder and an attention-based decoder that recurrently predicts \(k\) spectral frames of speech per timestep, and produces very natural sounding speech (apart from spectrogram artifacts). In a recent extension, Tacotron has been modified to apply prosody transfer [66] and voice transfer [37] by conditioning it on a prosody embedding or a speaker embedding [84], respectively. The speaker-conditional Tacotron shows the potential for voice conversion. We ran several experiments with a Tacotron decoder but found that it produced more artifacts than our convolutional networks (such as stuttering and repetition of frames). These artifacts were likely due to the attention mechanism and it could be worthwhile to continue research into adopting Tacotron for voice conversion.

Lastly, we remark that our network architecture has several limitations that could constrain its performance. We conducted various experiments to increase the size of our models, but we found that the larger models became increasingly more unstable to train until convergence. Our network architecture is relatively simple and we expect that there are many possible modifications that would increase the network stability. For example, adding residual connections is known to be critical in training deep networks, as it improves the gradient flow [73]. Furthermore, the speaker dependence in the decoder was implemented by concatenating the latents with the speaker embedding vector. It could be more powerful to inject the speaker embedding into each layer of the network, as was done in [75].

### 8.2.3 Loss functions

In all our experiments, the loss function was composed solely of a reconstruction loss \(\mathcal{L}_x\) and a latent loss \(\mathcal{L}_z\). This loss function does not specify that the decoder has to do voice conversion when a different target speaker id is given, and we did indeed observe that for some models this did not happen. In this section, we discuss alternative loss functions that might match better with the voice conversion objective.

A clear extension to our method is to add a GAN loss to our objective function in order to encourage that the converted speech samples \(p(x_{tgt}|x_{src, tgt})\) are indistinguishable from the target speaker distribution \(p(x_{tgt})\). This presents us with the difficulty that we should be able to
discriminate between the converted and true distributions for each speaker \(s_{tgt}\) in our dataset. We could either add a separate discriminator for each speaker \(D_s(x)\), or use a conditional discriminator \(D(x|s)\). Hsu et al. [34] have a similar setup to our method except that they perform voice conversion on individual frames. They show that adding a GAN loss with speaker-conditional discriminator improves the sharpness of the spectral features. This suggests that this approach may be fruitful, although it remains to be seen how well it generalizes to datasets with a large speaker variability.

Another option to improve our loss function would be to maximize the mutual information \(I(x,s)\). We showed in this work that mutual information can be evaluated for different latent space formulations and that it seems to correlate well with decoder conditionally (as expected by its definition). By adding \(-I(x,s)\) to our loss function we enforce the mutual information between \(x\) and \(s\) to grow. Note that we do not have to put any further restrictions on this mutual information loss, because it is bounded by \(\min(H(x), H(s))\). We expect that \(H(s)\) is substantially lower than \(H(x)\) because \(s\) is discrete while \(x\) is continuous multivariate. In fact, converging to \(I(x,s) = H(s)\) would be very desirable, as it means that \(x\) and \(s\) are a function of each other, in other words, we can distinguish the speaker \(s_i\) for all converted samples \(x_i\). We note that in our experiments \(I(x,s)\) seemed not to be bounded by \(s\) and as such, there is still room for improvement in our models. One potential obstacle in this approach is that the entropy definitions required to calculate the mutual information (equations A.14 - A.16 in appendix A) do contain a summation over speakers which prevents it from generalizing to large datasets. However all the summations can be rewritten as expectations (as they are of the form \(\sum_s p(s)f(s)\)) and as such can be estimated by their Monte Carlo approximation. Taken altogether, the derivations and experimental evidence that we have presented in this work suggest that mutual information is a likely candidate to improve the voice conversion objective.

Finally, the autoencoding architecture with a speaker-dependent decoder is sub-optimal in our voice conversion for ASR personalization setup. This is due to the decoder (the generative component of our model) being trained to optimize \(p(x_{src}|z_{src}, s_{src})\) for all speakers \(s_{src}\), while during inference we will only use it to decode using the target speaker id \(p(x_{tgt}|z_{src}, s_{tgt})\). This means that a lot of the model capacity of the decoder is wasted on modeling speech for all other speakers. A simple solution to this problem would be to train two separate decoders that share the same encoder. During training, we use \(\text{dec}_{src}(z_{src}, s_{src})\) to decode all of the samples by the source speakers \(s_{src} \neq s_{tgt}\) and \(\text{dec}_{tgt}(z_{tgt}, s_{tgt})\) to decode the samples of the target speaker. During inference, we can discard the source decoder, because we only will use the target decoder. Future research should show how feasible this approach is for specific modeling of a single target speaker.

### 8.2.4 ASR Personalization

In our ASR personalization experiments, we found that voice conversion failed to personalize the ASR system for a specific target speaker. We did we see that finetuning the ASR model using voice converted data increased the general ASR performance.

The failure of our voice conversion models to perform ASR personalization is not surprising because our models did not generalize well to the ASR training dataset (LibriSPeech). We managed to train a model that generalized better to this dataset (the Ours+ VAE model), but due to resource constraints, we were unable to use it for ASR personalization. Instead of improving our voice conversion model, we could try to select the samples where voice conversion worked successfully and use only those for ASR personalization. We have provided evidence that the latent loss is indicative of the conversion quality and as such we could use a threshold on the latent loss to separate the successfully converted samples from the rest. It would be interesting to
see what the ASR performance of a model that was finetuned with those specific samples would be as it would give an estimate to the potential of voice conversion in ASR personalization.

Interestingly, our voice conversion model did manage to increase general ASR performance, even though it did not generalize well to the ASR dataset. One explanation is that by adding noise these samples helped to increase regularization. Several noise augmentations are known to increase model performance such as adding background noise \cite{2,67}, morphing temporal structure \cite{43}, and auditive effects such as reverb \cite{31} or vocal tract length perturbation \cite{35}. Closer to our setup is DeVries and Taylor \cite{14}, who show that training an autoencoder and using sampling from the feature space can increase performance for ASR. More baselines should be included in future analyses to distinguish the effects of voice conversion from other augmentation effects.

Finally, we note that there are not many publications on ASR personalization and that there are no standard datasets. In fact, the dataset desiderata for training an ASR model and training a voice conversion model are in conflict. In an ASR dataset, it is desirable to have as many speakers and noise conditions as possible, in order to increase robustness (note that LibriSpeech does satisfy this requirement). For a voice conversion dataset, on the other hand, we generally want consistent and clean speech because it makes it easier to model the target speaker. Correspondingly, we found it difficult to train voice conversion models on larger speaker datasets. Hosseini-Asl et al. \cite{30}, the main other reference for ASR personalization by voice conversion reported results on the TIMIT \cite{21} and the WSJ datasets \cite{59}, which are substantially cleaner and smaller than LibriSpeech. However, these datasets are too small to train our ASR model on \cite{2}. Finally, they grouped all male and female speakers into two categories and only focused on gender conversion. This is a big limitation as we showed that this type of conversion is the easiest to achieve. Personalizing an ASR model for a specific accent instead of a specific gender remains an open goal in ASR personalization.
Appendices
Appendix A

Derivations of Mutual Information

In this appendix, we derive some probabilistic metrics that we use in the analysis of our voice conversion models. We are especially interested in quantifying how dependent on speaker our learned decoder model is and which aspects of speaker are most important in this conditioning. We show that we can interpret all the different latent space models in a single probabilistic framework and we derive mutual information and conditional entropy bounds that describe the quantities that we are interested in.

A.1 Generative model

In this section, we show how our encoder and decoder define a joint probabilistic model. In section A.2 we will continue to use this model to derive our metrics of interest.

A.1.1 Speaker and speaker attributes

We first define the random variables that describe a speaker and its characteristics. Each speaker is encoded by a unique speaker id $s_j$, and has known attributes such as gender or accent. There are multiple attributes, but we will only introduce a single random attribute variable $a$. We will derive general equations that can be applied to any attribute $a$. The attributes are discrete class-like variables, and each speaker can only belong to a single class for each attribute.

Formally, let $A(s_j)$ be the function that returns the attribute class for speaker $s_j$ then:

$$p(a_i|s_j) = \begin{cases} 1 & \text{iff } a_j = A(s_j) \\ 0 & \text{otherwise} \end{cases} \quad \text{(A.1)}$$

From which it follows that the marginal probability of a class $a_i$ is simply given by the sum of the probabilities of the speakers that belong to that class.

$$p(a_i) = \sum_{s_j \in a_i} p(s_j) \quad \text{(A.2)}$$

Both speaker id $s$ and attribute are discrete variables which means that summations over them are tractable.
A.1.2 Joint probability

In our graphical model, our encoder and decoder specify \( p(z_i|x_i) \) and \( p(x_i|z_i, s_j) \) respectively. Together, they specify the joint probability \( \{x_i, s_i, a_i\}^N_{i=1} \) which we will use in later sections to derive our metrics of interest. We now explain the generative model of the joint probability.

**Decoding distribution** The log probability of \( p(x|z, s) \) is defined by our reconstruction loss \( L_x \).

\[
- \log p(x_i|z_h, s_j) = ||x_i - \text{dec}(z_h, s_j)||_1
\]  

(A.3)

Note that this distribution is properly normalized because we use the L1 loss, but all of our derivations can also be applied for non-normalized losses by introducing a normalization constant.

**Encoding distribution** For reasons that will be clear in section A.3, we adopt the model of [42] were we view our encoder as a variational approximation \( q(z_h|x_i) \) to the distribution \( p(z_h|x_i) \). We will now show how we can define \( q \) for each of our different latent space models. For our VAE models, we have already defined \( q \):

\[
q_{\text{VAE}}(z_h|x_i) = \mathcal{N}(z_h|z_i^e, z_i^e \mathbf{I})
\]  

(A.4)

However, we can also define \( q \) for our deterministic AE and AE-B models:

\[
q_{\text{AE}}(z_h|x_i) = q_{\text{AE-B}}(z_h|x_i) = \begin{cases} 
1 & \text{iff } z_h = \text{enc}(x_i) \\
0 & \text{otherwise} 
\end{cases}
\]  

(A.5)

And similarly for our VQAE model:

\[
q_{\text{VQAE}}(z_h|x_i) = \begin{cases} 
1 & \text{iff } z_h = Q(\text{enc}(x_i)) \\
0 & \text{otherwise} 
\end{cases}
\]  

(A.6)

We will continue our derivations with the general distribution \( q \), so that we can substitute the appropriate \( q \) for each latent space type.
A.2 Mutual information and conditional entropy

We are interested in the mutual information $I(x, a)$ and $I(x, s)$ and the conditional entropy $H(x|a)$ and $H(x|s)$ because they all quantify the extent to which $x$ is dependent on speaker (attribute). We use the following factorizations:

$$I(x, s) = H(x) - H(x|s)$$  \hspace{1cm} (A.7)

$$I(x, a) = H(x) - H(x|a)$$  \hspace{1cm} (A.8)

$$H(x|a) = H(x, a) - H(a)$$  \hspace{1cm} (A.9)

In equations A.13 - A.16 we rewrite all the entropies in the equations above. We express the entropies in terms of their expectation, and we rewrite any probability of $x_i$ in terms of the conditional probability $p(x_i|s_j)$ in order to be closer to our decoding distribution (equation A.3). For the entropy $H(x)$, we can rewrite $p(x_i)$ by marginalizing over $s$ yielding equation A.14. The entropy $H(a)$ can by found by substituting A.2 to yield A.16.

Finally in order to find $H(x, a)$ we rewrite the joint probability $p(x, a)$ by: (A.10) marginalizing over speaker id, (A.11) factorizing according to our graphical model in figure A.1 and (A.12) using the sparseness of the conditional distribution $p(a_i|s_j)$ in equation A.1 to rewrite the sum.

$$p(x, a) = \sum_{s_j} p(x, a, s_j)$$  \hspace{1cm} (A.10)

$$= \sum_{s_j} p(x_i|s_j)p(a_i|s_j)p(s_j)$$  \hspace{1cm} (A.11)

$$= \sum_{s_j \in a} p(x_i|s_j)p(s_j)$$  \hspace{1cm} (A.12)

Yielding the following equations for all of the relevant entropies:

$$H(x|s) = E_{p(x, s)} \left[ - \log p(x_i|s_i) \right]$$  \hspace{1cm} (A.13)

$$H(x) = E_{p(x)} \left[ - \log \sum_{s_j} p(x_i|s_j)p(s_j) \right]$$  \hspace{1cm} (A.14)

$$H(x, a) = E_{p(x, a)} \left[ - \log \sum_{s_j \in a} p(x_i|s_j)p(s_j) \right]$$  \hspace{1cm} (A.15)

$$H(a) = E_{p(a)} \left[ - \log \sum_{s_j \in a} p(s_j) \right]$$  \hspace{1cm} (A.16)

A.3 Variational entropy lower bound

In order to calculate the entropies in equations A.13 - A.16 we need to be able to evaluate $p(x_i|s_j)$. However, our decoding distribution in equation A.3 is conditional on $z_h$, and integrating over $z_h$ is intractable. In this section, we will derive a tractable variational lower bound.

Analogous to [42], we start by rewriting the KL divergence between $q(z|x)$ and $p(z|x)$ such that it includes $p(x_i|s_j)$ (equation A.17). We then solve for $p(x_i|s_j)$ and rewrite the expectations once more to yield equation A.18. We refer to 2.2 for more details on these steps. Equation
\[ A.18 \] now defines \( p(x_i | s_j) \) in terms of our decoding distribution. We introduce a bound on this distribution by grouping all terms that are a function of \( x_i \), but that are constant in terms of \( s_j \) into a "constant" function \( c(x_i) \).

Note that this bound is less tight than the evidence lower bound (ELBO) of [22]. We choose to include \( KL [q(z | x_i)|p(z)] \) into our constant function because the prior \( p(z) \) is not defined for our deterministic auto encoding models.

\[
\begin{align*}
KL [q(z | x)|p(z)] &= E_{q(z_i|x_i)} [\log q(z_i|x_i)] - E_{q(z_i|x_i)} [\log p(x_i, z_i | s_j)] + \log p(x_i | s_j) \\
&= \log p(x_i | s_j) = E_{q(z_i|x_i)} [\log p(x_i | z_i, s_j)] - KL [q(z | x_i)|p(z)] - KL [q(z | x_i)|p(z|x_i)]
\end{align*}
\] (A.17)

\[
\log p(x_i | s_j) = E_{q(z_i|x_i)} [\log p(x_i | z_i, s_j)]
\] (A.18)

We also define \( C(x) = E_{p(x_i)} [\log c(x_i)] \) in equation \[ A.20 \] as it will be useful later on. Note that \( C(x) \) is a function of the random variable \( x \), while \( c(x_i) \) is a function of an observed value for that variable.

Because \( C(x) \) is defined as the sum of two KL divergences, which are both positive semidefinite, we know that \( C(x) \) is positive semidefinite itself:

\[
C(x) = E_{p(x_i)} \left[ KL [q(z_i | x_i)|p(z)] + KL [q(z | x_i)|p(z)] \right] (A.20)
\]

\[
C(x) \geq 0 (A.21)
\]

Because we can evaluate the RHS of equation \[ A.19 \] we can use it to derive tractable bounds on the entropies in equations \[ A.13- A.15 \]. For example, we can introduce the constant \( c(x_i) \) into equation \[ A.14 \] (yielding equation \[ A.22 \]), because \( c(x_i) \) is independent of \( s_j \), we can take it out of the sum and use the log-quotient rule to move it out of the expectation to arrive at \[ A.24 \]. The remaining expression is now a bound \( H_b(x) \) of the original entropy. Because \( C(x) \geq 0 \) we know that \( H_b(x) \) is a lower bound.

\[
H(x) = E_{p(x_i)} \left[ -\log \sum_{s_j} \frac{c(x_i)}{c(x_i)} p(x_i | s_j) p(s_j) \right] (A.22)
\]

\[
= E_{p(x_i)} \left[ -\log \sum_{s_j} c(x_i) p(x_i | s_j) p(s_j) \right] + C(x) (A.23)
\]

\[
\geq H_b(x) (A.24)
\]

In a similar fashion, we derive the bounds for the entropies in equations \[ A.13- A.15 \]

\[
H(x) = H_b(x) + C(x) (A.25)
\]

\[
H(x | s) = H_b(x | s) + C(x) (A.26)
\]

\[
H(x, a) = H_b(x, a) + C(x) (A.27)
\]
A.4 Metrics bounds

Now that we have derived bounded metrics that we can evaluate numerically, we continue to interpret how each metric is bounded and what this tells us about how we can use them to make fair comparisons.

A.4.1 Mutual information

We note that we can use the bounded entropy to get an exact expression for the mutual information by substituting the identities of A.13 – A.15 into A.9:

\[
I(x, a) = (H_b(x) + C(x)) - (H_b(x|a) + C(x)) = H_b(x) - H_b(x|a)
\]  

which also holds for \(I(x, s)\). This means that we can safely compare mutual information between models that have different distributions \(q\), even though they potentially have a very different \(C(x)\).

Mutual information is bounded as follows:

\[
0 \leq I(x, a) \leq \min(H(x), H(a))
\]

Because \(H(a)\) is different for each attribute, this means that we cannot safely compare the mutual information between different attributes, as the upper bound might be dependent on the attribute entropy.

A.4.2 Conditional Entropy

We want to be able to compare the extent to which the decoder is dependent on \(a\) between different speaker attributes, but we cannot use the mutual information \(I(x, a)\) as described in the previous paragraph. In this paragraph, we show that we can use the conditional entropy for this comparison.

The conditional entropy is bounded as follows:

\[
0 \leq H_b(x|a) \leq H_b(x) \leq H(x)
\]

Because \(H_b(x|a)\) is not bounded by attribute entropy \(H(a)\) we can safely compare it between different attributes, and even between attribute \(H(x|a)\) and speaker id \(H(x|s)\). This only holds within a latent space type, and we should be careful when comparing absolute values of \(H_b(x|a)\) across different distributions of \(q\), as the upper bound \(H(x)\) might differ between them.

A.5 Implementation

In section A.2 and A.3 we defined the quantities required to calculate the mutual information and conditional entropy. Here we discuss practical considerations in evaluating them.

Because our data comes in pairs \(\{x_i, s_i, a_i\}_{i=1}^N\), we can approximate the expectations in equations A.13 – A.16 by the Monte Carlo estimate over the joint \(p(x_i, s_i, a_i)\) (and discarding any variables that are not used in the expectation). The expectation over \(z\) in equation A.19 can be calculated exactly for our deterministic AE, AE-B and VQAE models:
For our VAE model, we could use the Monte Carlo estimate to approximate the expectation over $z$. However, we decide to deterministically use the mean prediction $\mu_i$ in order to be more consistent with our other latent space models. Though this might give us a biased estimate, so could the Monte Carlo estimate, as it is nested within another expectation [60, 61, 29].

We found it necessary to work in log space as much as possible in order to maintain numerical stability. Using the log-sum-exp trick for all summations over $s$ in equations [A.13] - [A.16] eliminated all numerical instabilities. We use the frequencies of the speakers in our training data to estimate $p(s_i)$. 

$$q(z_i|x_i) = \begin{cases} 1 & \text{iff } z_i = \text{enc}(x_i) \\ 0 & \text{otherwise} \end{cases} \quad (A.32)$$

$$c(x_i)p(x_i|s_i) = p(x_i|z = \text{enc}(x_i), s_i) \quad (A.33)$$
Appendix B

ASR Speaker distributions

This section contains the histograms that show the distribution of error rates across speakers for the different ASR models that we trained.
APPENDIX B. ASR SPEAKER DISTRIBUTIONS

B.1 Female to male conversion

Figure B.1: Speaker histograms female to male conversion. Speaker CER distributions for (a) baseline ASR model (female), (b) ASR model personalized for speaker 2769 (male), and (c) ASR model personalized for speaker 2004 (female). Speakers are ordered by average CER score (x-axis) and CER values are binned (y-axis). CER count is coded by color intensity s.t. darker colors mean higher values. Speakers are colored by gender, and the target speaker (2769) is displayed in green. Dots indicate averages per speaker.
B.2 Male to female conversion

Figure B.2: Speaker histograms male to female conversion. Speaker CER distributions for (a) baseline ASR model (male), (b) ASR model personalized for speaker 2004 (female), and (c) ASR model personalized for speaker 2769 (male). Speakers are ordered by average CER score (x-axis) and CER values are binned (y-axis). CER count is coded by color intensity s.t. darker colors mean higher values. Speakers are colored by gender, and the target speaker (2004) is displayed in yellow. Dots indicate averages per speaker.
Appendix C

Model Architectures

In this appendix, we describe the layer and filter sizes of our encoder and decoder networks.

C.1 Encoder Architectures

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel size</td>
<td>(3, 3)</td>
<td>(3, 3)</td>
<td>(3, 3)</td>
<td>(3, 3)</td>
</tr>
<tr>
<td>stride</td>
<td>(1, 1)</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
</tr>
<tr>
<td>padding</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
</tr>
<tr>
<td>dilation</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>channels</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>128</td>
</tr>
</tbody>
</table>

Table C.1: Model architecture Ours encoder.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel size</td>
<td>(7, 7)</td>
<td>(5, 5)</td>
<td>(4, 3)</td>
<td>(3, 3)</td>
</tr>
<tr>
<td>stride</td>
<td>(3, 1)</td>
<td>(2, 1)</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
</tr>
<tr>
<td>padding</td>
<td>(0, 3)</td>
<td>(0, 2)</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
</tr>
<tr>
<td>dilation</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>channels</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>256</td>
</tr>
</tbody>
</table>

Table C.2: Model architecture Ours+ encoder.
C.2 Decoder Architectures

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel size</td>
<td>(4, 3)</td>
<td>(3, 3)</td>
<td>(3, 3)</td>
<td>(126, 3)</td>
<td>(3, 3)</td>
<td>(5, 5)</td>
<td>(5, 5)</td>
</tr>
<tr>
<td>stride</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>padding</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(1, 1)</td>
<td>(2, 2)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>dilation</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>channels</td>
<td>64</td>
<td>32</td>
<td>32</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

Table C.3: Model architecture Ours decoder.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel size</td>
<td>(1, 1)</td>
<td>(3, 3)</td>
<td>(4, 3)</td>
<td>(5, 5)</td>
<td>(7, 3)</td>
<td>(62, 3)</td>
<td>(5, 5)</td>
</tr>
<tr>
<td>stride</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
<td>(3, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>padding</td>
<td>(0, 0)</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(0, 2)</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>dilation</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>channels</td>
<td>512</td>
<td>512</td>
<td>256</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>1</td>
</tr>
</tbody>
</table>

Table C.4: Model architecture Ours+ decoder.
Appendix D

Online Audio samples

All the audio samples that are presented in this thesis can be viewed and listened to online. The url to the webpage containing the samples is [http://masterthesis.tivaro.nl/audiosamples](http://masterthesis.tivaro.nl/audiosamples).
The credentials to the website are:

- **Username**: reader
- **Password**: Thesis

In addition to the figures in this thesis, we provide a comparison between our method and the method of van den Oord et al. [1]
Glossary

AE, 42
AE-B, 42
ASR personalization, 8

bitdepth, 11
bottleneck, 14

CER, 27
CTC, 23
cycle-GAN, 34

ELBO, 15, 68
direct-to-end ASR systems, 23

frames, 12

GAN, 15
GMM, 23
Griffin-Lim, 13

HMM, 22

Intent level consistency, 29

MAP, 15
MOS, 30

Phoneme level consistency, 29
Phoneme level variability, 16
phoneme level variability, 29
phonetic posteriogram, 33
Prosody level variability, 17
prosody level variability, 29

sample rate, 11
semi-supervised ASR personalization, 9
speaker embedding, 17
spectrogram, 12
STFT, 12
style transfer, 28

Target Speaker Similarity, 30
Transcription level consistency, 29
Transcription level variability, 16
transcription level variability, 29
TTS, 33

VAE, 42
Voice Intelligibility, 31
Voice Quality, 31
VQAE, 43

WER, 27
Bibliography


[34] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. September 2017.


80


BIBLIOGRAPHY


