THE UNIVERSITY
of EDINBURGH

Learning Disentangled Speech Representations

Jennifer A. Williams

Doctor of Philosophy
The University of Edinburgh
February 10, 2022

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text. In such cases where the work was completed as part of a group I have made substantial contributions to the work and the contributions are clearly indicated. The work in this thesis has not been submitted for any other degree or professional qualification.

(*Jennifer A. Williams*)

Abstract

A variety of informational factors are contained within the speech signal and a single short recording of speech reveals much more than the spoken words. The best method to extract and represent informational factors from the speech signal ultimately depends on which informational factors are desired and how they will be used. In addition, sometimes methods will capture more than one informational factor at the same time such as speaker identity, spoken content, and speaker prosody.

The goal of this dissertation is to explore different ways to deconstruct the speech signal into abstract representations that can be learned and later reused in various speech technology tasks. This task of deconstructing, also known as disentanglement, is a form of distributed representation learning. As a general approach to disentanglement, there are some guiding principles that elaborate what a learned representation should contain as well as how it should function. In particular, learned representations should contain all of the requisite information in a more compact manner, be interpretable, remove nuisance factors of irrelevant information, be useful in downstream tasks, and independent of the task at hand. The learned representations should also be able to answer counter-factual questions.

In some cases, learned speech representations can be re-assembled in different ways according to the requirements of downstream applications. For example, in a voice conversion task, the speech content is retained while the speaker identity is changed. And in a content-privacy task, some targeted content may be concealed without affecting how surrounding words sound. While there is no single-best method to disentangle all types of factors, some end-to-end approaches demonstrate a promising degree of generalization to diverse speech tasks.

This thesis explores a variety of use-cases for disentangled representations including phone recognition, speaker diarization, linguistic code-switching, voice conversion, and content-based privacy masking. Speech representations can also be utilised for automatically assessing the quality and authenticity of speech, such as automatic MOS ratings or detecting deep fakes. The meaning of the term 'disentanglement' is not well defined in previous work, and it has acquired several meanings depending on the domain (e.g. image vs. speech). Sometimes the term 'disentanglement' is used interchangeably with the term 'factorization'. This thesis proposes that disentanglement of speech is distinct, and offers a viewpoint of disentanglement that can be considered both theoretically and practically.

Acknowledgements

There are many people to thank who have supported me during the creation of this thesis. It has been an interesting personal and professional journey. I owe my gratitude to my supervisor, Simon King. Simon has been the best PhD supervisor that I could have ever hoped for. His encouragement all throughout my PhD helped me to overcome a variety of obstacles. Simon has been an excellent coach and mentor. Being able to get Simon's perspective on a topic is something that I will always cherish. I admire that Simon is a role model of values like ethics, equity, inclusivity, and technical excellence. This thesis would not have been possible without Simon's guidance, patience, and encouragement.

During the course of my PhD, there have been several faculty who supervised me and contributed to the development and progression of my work. For the entire second year of my PhD, I stayed in Tokyo on a lab visit with the group of Junichi Yamagishi at National Institute for Informatics where I gained valuable experience under his mentorship. The eventual direction of my work as well as the impact would not have been possible without Junichi's mentorship. I also enjoyed working with Junichi's postdoctoral researchers, Erica Cooper and Zhao Yi, who helped me elevate my technical work. Thank you to Catherine Lai for being part of my PhD journey at Edinburgh from day one. Catherine has been a tremendous resource of encouragement and technical depth. I am grateful to Steve Renals who early on gave me the advice that I needed as I transitioned into speech processing. Steve has provided valuable insight throughout the development of this work. I would like to thank Sotorios Tsaftaris in the Institute of Digital Communications for allowing me to informally join his research group during my first year so that I could gain insight into methods for image segmentation.

I am grateful to have worked with incredible students and postdocs at various stages of this work. From chats in the Informatics Forum to long hours spent in coffee shops, this work would not have been possible without the insight and technical guidance of my colleagues. I deeply appreciate the experience of working closely with technical collaborators and co-authors who have taught me so much about the value of teamwork and the shared spirit of curiosity: Joanna Rownicka, Pilar Oplustil-Gallegos, Jason Fong, Paul-Gauthier Noé, Cassia Valentini Botinhao, Erica Cooper and Zhao Yi. At the National Institute for Informatics in Tokyo, Japan, I thank all of my colleagues for their thoughtful help and feedback. At the Centre for Speech Technology Research in Edinburgh, I would like to thank all of my colleagues for their support and encouragement. I also want to thank my CDT/Data Science cohort and administrators.

I am grateful for the support of my family throughout this journey. Despite living far apart, we made the most of our video calls and text messages. I enjoyed sharing my work and accomplishments with them, and they have celebrated the milestones with me along the way.

List of Tables

List of Figures

List of Publications

1. **Jennifer Williams**, and Joanna Rownicka. (2019). "Speech Replay Detection with x-vector Attack Embeddings and Spectral Features." In *Proceedings of Interspeech 2019*, pages 1053-1057.

2. **Jennifer Williams**, and Simon King. (2019). "Disentangling Style Factors from Speaker Representations." In *Proceedings of Interspeech 2019*, pages 3945-3949.

3. Pilar Oplustil-Gallegos, **Jennifer Williams**, Joanna Rownicka, and Simon King. (2020). "An Unsupervised Method to Select a Speaker Subset from Large Multi-Speaker Speech Synthesis Datasets." In *Proceedings of Interspeech 2020*, pages 1758-1762.

4. **Jennifer Williams**, Joanna Rownicka, Pilar Oplustil-Gallegos, and Simon King. (2020). "Comparison of Speech Representations for Automatic Quality Estimation in Multi-Speaker Text-to-Speech Synthesis." In *Proceedings of Odyssey 2020 - The Speaker and Language Recognition Workshop*, pages 222-229.

5. **Jennifer Williams**. (2020). "End-to-End Signal Factorization for Speech: Identity, Content, and Style." In *International Joint Conference on Artificial Intelligence (IJCAI)*, Doctoral Consortium.

6. Zhao Yi, Haoyu Li, Cheng-I. Lai, **Jennifer Williams**, Erica Cooper, and Junichi Yamagishi. (2020). "Improved Prosody from Learned F0 Codebook Representations for VQ-VAE Speech Waveform Reconstruction." In *Proceedings of Interspeech 2020*, pages 4417-4421.

7. **Jennifer Williams**, Zhao Yi, Erica Cooper, and Junichi Yamagishi. (2021) "Learning Disentangled Phone and Speaker Representations in a Semi-Supervised VQ-VAE Paradigm." In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7053-7057.

8. **Jennifer Williams**, Jason Fong, Erica Cooper, and Junichi Yamagishi. (2021). "Exploring Disentanglement with Multilingual and Monolingual VQ-VAE." In *Proceedings of ISCA Speech Synthesis Workshop (SSW11)*.

9. Nicolas E. Müller, Franziska Dieckmann, Pavel Czempin, Roman Canals, Konstantin Böttinger, and **Jennifer Williams**. (2021). "Speech is Silver, Silence is Golden: What do ASVspoof-trained Models Really Learn?" In *Proceedings of ISCA ASVspoof 2021 Workshop*.

10. Jason Fong, **Jennifer Williams**, and Simon King. (2021). "Analysing Temporal Sensitivity of VQ-VAE Sub-Phone Codebooks." In *Proceedings of ISCA Speech Synthesis Workshop (SSW11)*.

11. **Jennifer Williams**, Junichi Yamagishi, Paul-Gauthier Noé, Cassia Valentini-Botinhao, Jean-François Bonastre. (2021). "Revisiting Speech Content Privacy". In *1st ISCA Symposium of the Security & Privacy in Speech Communication.*

Contents

Chapter 1

Introduction

1.1 Motivation

The human speech signal is a very rich and dynamic source of different types of informational factors. A single short recording of speech reveals much more than spoken words. It also provides information about a speaker's vocal organs, native accent, and prosody, as well as features that correlate with age, gender, and emotional affect, to name a few. In fact, the speech signal also contains information about the recording environment such as the microphone and room characteristics. Speech science seeks to model and utilize the different types of information for different purposes. Some information can be extracted directly from the speech signal as low-level acoustic properties, such as signal energy or power. Other information can be revealed by applying a series of mathematical transforms to extract features which are based on local context, such as the phones or content. There are some features, such as speaker identity, that must be modeled using a context that is more global by nature and consistent across time because the speaker identity remains invariant for the duration of an utterance no matter how short or long.

The best method to extract and represent informational factors from the speech signal ultimately depends on which informational factors are desired and how they will be used. In addition, sometimes methods will capture more than one factor at the same time such as speaker identity, spoken content, and speaker prosody. In speech science it is not yet fully understood exactly which informational factors in the speech signal can be fully separated "by definition" (e.g., can speaker accent be completely separated from speaker identity?). When it comes to separating informational factors for speech technology that require them (such as speaker and content for voice conversion), it has been an open question as to the best method. This thesis addresses the core of that problem by learning how to *disentangle* informational factors through representation learning and then utilizing the disentangled representations in downstream speech technologies.

Recent advances in machine learning have opened up new ways of thinking about speech technology. For example, it is possible to develop deep learning methods that model speech signal content but require little or no supervision [van den Oord et al., 2017; Baevski et al., 2020; Shor et al., 2020]. It is also possible to generate synthetic speech with high enough quality that it is difficult to distinguish from natural speech. Some neural network architectures have

characteristics that allow specific information to be represented in various key places in the network and this is an advantage for representation learning. Architectures that learn distributed representations are attractive candidates for disentanglement. This thesis will explore techniques for disentanglement in later chapters. To begin, this introductory chapter discusses three main viewpoints of disentanglement (transformational, factorizational, and functional) and adopts some principles of disentanglement that will be used in later chapters to assess the proposed techniques. Deep learning in the image domain is significantly mature and techniques are often borrowed from the image domain to be applied to speech processing problems. A comparison and contrast between definitions from the image domain will be presented in the next section in order to establish the approach to speech disentanglement that is used in this thesis.

1.2 Approaches to Disentanglement

One of the challenges for developing disentangled speech representations is that the concept of *disentanglement* is not well-defined across domains such as the image domain and the speech domain. While it is reasonable to draw analogies between image and speech disentanglement, the inconsistent definitions are problematic. This section aims to first explore several definitions and then adopt one in particular for speech processing. Intuitively speaking, if information has been disentangled then this should mean that information of one type is found in one representation **A** and information of a different type is found in a separate representation **B**. For speech it is not known how many distinct types of information are contained in the speech signal and it is not clear if there are limits regarding which types of information can be completely disentangled. For example, speaker accent is related to both the speaker's identity as well as how phones are pronounced. In order to disentangle speaker identity and accent, an assumption would have to be made that the speaker identity involves only the voice (due to the vocal tract and nasal cavities) but does not involve any nuances about how sounds are articulated or realized as phones. The problem of finding a theoretical limit to speech disentanglement is a very large research problem that cannot be answered in this thesis. However, this section explores some ideas of what it means in principle to achieve disentangled representations.

1.2.1 Transformational Disentanglement

Disentanglement could be thought of in terms of which values can and cannot be transformed in a data representation. Transformational disentanglement is based on the idea that some information can be changed (variant) while other information cannot (invariant). Conceptually, this is similar to painting a house: a blue house could be painted red but it would still remain the same house. The house represents information that is invariant while the colour represents information that is variant. In another analogy, it might be difficult to identify variant and invariant information between a 1-storey house and a 3-storey house - both are houses but perhaps they have differing immutable qualities. This analogy with houses captures some of the difficulty of using a transformational definition of disentanglement, and more examples are provided below and discussed in the contexts of image and speech processing.

1.2.1.1 Image Style Transfer

A widely-circulated example of transformational disentanglement comes from the image processing domain wherein an image has been stylized to reflect the artwork of a famous painter, as in Figure 1.1. In this example, the invariant properties are considered to be the canal and buildings (content) and the variant properties are considered to be the colours and artistic rendering of those colours and patterns (style) [Gatys et al., 2016]. By combining the style of a famous painter from one image with the content of a different image, the result is a new image that appears as if it had been painted by a famous painter. To accomplish image style transfer, neural networks learned how to extract a "style representation" and a "content representation" as well as how to re-combine them.

Figure 1.1: Image from Gatys et al. [2016] demonstrating transformational disentanglement. Neural networks can be used to combine factors of variation (style) with invariant factors (content) to achieve a transformed or stylized image.

1.2.1.2 Speech Style Transfer

The example image displayed in Figure 1.1 requires some aspect of disentanglement between content and style, but according to the authors the problem of disentanglement is not well-defined [Gatys et al., 2016]. The process of learning to transform an image requires defining style from one image and content from another image but the process does not address how to recover those representations from a single stylized image. Likewise, the style transfer analogy from image processing is not straightforward when applied to speech. This is due to poor definitions of speech style as well as the fact that speech is a dynamic sequence whereas images are static. Speaking style will be explored later in this thesis, especially in Chapter 4. However in this thesis the definition of speaking style is taken to be *how a speaker adapts their speaking manner according to the context* and it is a pragmatic definition meant to account for differences between the manner of conversational and read speech. Other forms of speech style could involve patterns of expressivity or predictable prosodic changes. However since speech is a dynamic sequence it is not necessarily meaningful to transform an utterance into multiple prosodic styles using style transfer (with the image analogy) because the meaning of an utterance changes at the same time that the prosody changes as well.

1.2.1.3 Intelligent Agent Interaction

Consider the work of Higgins et al. [2018] and their proposed view that the physical world has certain properties that can be changed while other properties remain invariant. For example, a vehicle can move but a road cannot move, therefore a vehicle and a road can be disentangled. The authors build upon the analogy to the physical world to develop a set of principles of

transformational disentanglement. The principles seek to find underlying structure in the world that can be used to exploit transformational properties of data. Overall the principles they present are focused on mathematical definitions to define the achievement of disentanglement using group theory rather than processes that accomplish disentanglement. All of the analogies they provide are related to a scenario wherein an intelligent agent is interacting with the world. Disentanglement is based on whether the intelligent agent can perform an action (specifically a transformational action) that changes the world. The authors offer three characteristics of disentangled representations: compactness, modularity, and explicitness. Compactness measures whether a single informational factor is encoded as a single latent dimension, rather than multi-dimensional. Modularity involves whether or not a single factor of variation corresponds to a single dimension of the latent representation. Explicitness means that if a representation has been disentangled then all of the informational factors are modeled generatively and this information is also linearly decodable.

While the definitions proposed by Higgins et al. [2018] may have proven useful for intelligent agents interacting in a physical world, it is not clear how these principles could be adapted for disentangled speech representations. First, it is not known how many informational factors are contained within the speech signal and this alone makes it difficult to follow all three principles of modularity, compactness, and explicitness. It is also possible that information in speech-based representations is distributed across multiple dimensions. The number of informational factors may be dependent on the speech processing task or richness of the speech dataset. For example if a speech dataset has been labeled with metadata for speaker characteristics (age, gender, accent, etc) as well as utterance characteristics (emotion, expressivity, emphasis, etc) then the number of informational factors that can be modeled may depend on the dataset annotations. In that case, speech disentanglement may end up reflecting the limits of data labels rather than fundamental principles of speech itself.

1.2.2 Factorizational Disentanglement

A term commonly used in the image domain is *factorization* and this describes the act of separating information within a single representation using the indices of the representation. It is related to the transformational principles described earlier (compactness and modularity) but factorization is more general and allows for information to be encoded across multiple dimensions of a latent representation. Often in the image domain the term *factorization* is used interchangeably with the term *disentanglement*. This approach to disentanglement is based on how the information in a representation can be separated into distinct regions. As in Figure 1.2, the distinct regions of the latent representation (red values in z) correspond to distinct regions of the image (red object in image). Some of the assumptions that may be made for factorization could be the quantity or type of colours (e.g., black and white or coloured), the shapes of edges (e.g., round or sharp), or the number of target factors (e.g., two red circles). These assumptions contribute to a type of bias called the *inductive bias*. The example shown in Figure 1.2 comes from Locatello et al. [2020b] who insist that disentanglement is only achieved if inductive bias is used, otherwise disentanglement would be theoretically impossible. Their argument regarding the necessity for inductive bias is also maintained in their additional work on the topic (see: Locatello et al. [2019] and Locatello et al. [2020a]).

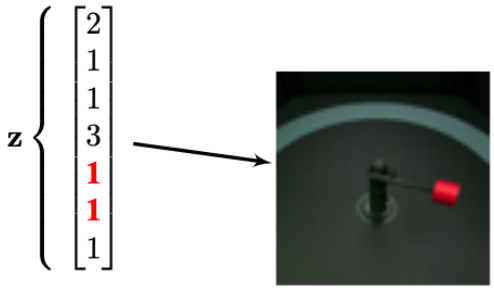


Figure 1.2: Image from Locatello et al. [2020b] demonstrating factorizational disentanglement. Neural networks can be used to identify specific factors of a representation (indices in z shown in red) that correspond to parts of an image (red object).

1.2.2.1 Image Segmentation

Factorization is widely used to perform image segmentation. An example of this comes from Chartsias et al. [2019] who used disentanglement to identify specific components of cardiac images for diagnostic purposes. The image segmentation is exact enough that the segmented image can be further processed for diagnosis such as measuring the narrowing of an artery or heart valve. Using disentanglement for image segmentation is based on an important assumption about how separable the components of the image are. In this case, the components of the image are 100% separable both in theory and in practice.

A common method to evaluate factorizational disentanglement is to attempt to decode from a latent representation while traversing the latent representation one dimension at a time. This evaluation technique is referred to as *latent traversal* [Higgins et al., 2016]. A limitation of this approach is that it relies heavily on the assumption that a single dimension corresponds to one factor of variation. If information is distributed across multiple dimensions in a representation, and the goal is to evaluate disentanglement between two representations **A** vs **B**, then latent traversal would not be relevant.

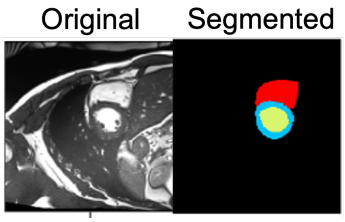


Figure 1.3: Image from Chartsias et al. [2019] demonstrating factorizational disentanglement for image segmentation. In this example an image of a heart is segmented into three anatomical structures: left ventricular cavity, myocardium, and right ventricular cavity. Using the segmented image, the sizes of the anatomical structures can be measured for the purpose of diagnosing a health condition.

1.2.2.2 Factorization in Speech

The work of Luo et al. [2019] provides a definition of disentanglement for pitch and timbre in music as: "a disentangled feature representation is defined as having disjoint subsets of feature dimensions that are only sensitive to changes in corresponding factors of variation from observed data". This definition is based on the notion that the data in a dataset is comprised of informational factors. Some factors will vary while other factors will be invariant. Once the varying factors are identified it would be possible to disentangle information [Bengio, 2013] using supervised or unsupervised techniques [Ridgeway, 2016]. In speech, factorization has served the purpose of removing irrelevant information from a representation such as a speaker embedding and then discarding what had been deemed irrelevant [Dehak et al., 2011]. Factorization has been used in speaker recognition to remove nuisance factors such as the variability caused by using different microphones in different recordings. After information has been removed, it could be argued that a representation is in some way more "pure". This type of factorization necessarily removes information since the technique is entirely based on applying linear transforms to a representation until factors of variation have been removed. Factorization in speech for speaker recognition resembles the principle of explicitness mentioned earlier from transformational disentanglement. The disentanglement sought after in this thesis aims to retain the information once it has been separated and this includes information that is variant as well as invariant.

1.2.3 Functional Disentanglement

According to Locatello et al. [2019], disentangled representations are defined by how they function. After disentanglement, the representations should separate distinct informational factors that have been identified in the data. Furthermore, the authors provide six principles of disentanglement. Disentangled representations should:

1. contain all of the information in a more compact manner
2. be interpretable
3. be independent from task at-hand
4. be useful for downstream tasks (including zero-shot learning)
5. integrate out nuisance factors
6. answer counter-factual questions

These six principles establish disentanglement in terms of functionality rather than by mathematical properties of the representations. The principles offer a more complete notion of disentanglement compared to other work. Functional disentanglement describes features of the representations as well as how to evaluate the representations. The principles are generalizeable and are not specific to image processing. Therefore a functional approach to defining disentanglement is adopted in this thesis and new principles are proposed in the next section.

Currently there are no single-best techniques to measure the intrinsic goodness of disentangled representations for speech apart from probing how well they perform in extrinsic tasks [Raj et al., 2019; Williams and King, 2019; Peri et al., 2020a; Chung et al., 2020]. Principles 3 and 4 are

important because the disentangled representations can be evaluated as to how much information they contain for prediction tasks that are separate from the disentanglement technique itself. Other forms of disentanglement evaluation in speech involve contrastive prediction tasks such as phone recognition and speaker recognition by observing that one disentangled representation "gains" information while another "loses" information [Ebbers et al., 2021]. Principle 6 is important because the speech representations can be used to answer counter-factual questions. For example, a disentangled phone representation should not perform well in a speaker recognition task.

1.3 Proposed Principles of Disentanglement

The following principles are proposed for the purpose investigating disentanglement of speech in this thesis:

1. The learned representations are sufficiently rich to model targeted speech features.
2. Representations that model targeted speech features do not also model non-target speech features (e.g., a representation of speaker identity does not also model speech content).
3. Representations have utility outside of the system that disentangles information.

1.4 Thesis Scope

The goal of this dissertation is to provide a machine learning technique that disentangles information in the speech signal into separate representations that can be re-used in various speech technology tasks. The learned representations can then be re-assembled in different ways depending on the downstream application. For example, in a voice conversion (VC) task, it is necessary to retain speech content and style but change only the speaker identity. Another goal of this thesis is to provide a set of intrinsic evaluation techniques to evaluate disentanglement completely independent of extrinsic tasks. There has been some recent work to separately factorize information from the speech signal. However, at the time of this writing, it remains an open research problem for how to best create multiple reusable representations of speech features using a single disentanglement technique.

1.4.1 Problem Statement

What is the best technique for disentangling multiple informational factors simultaneously from the speech signal and how can these disentangled representations be evaluated?

1.4.2 Approach

The approach to solve this problem involves exploring the utility of autoencoders and a rich latent space that can be learned in a self-supervised or semi-supervised manner. During the development of the disentanglement technique, different variations of the technique require different datasets due to the need to exploit metadata labels or data quality. Disentanglement will be considered in terms of three proposed principles that were inspired from work in the image domain. The adopted principles will serve to evaluate disentanglement in terms of the functional capabilities of representations.

1.4.3 Contributions

This dissertation contributes a novel and innovative lens to re-imagine speech processing through disentanglement of speech signal features using machine learning. The following contributions are made:

1. Learn to separate and retain informational factors from speech
2. Introduce three principles of disentanglement that are tailored to speech processing
3. Explore the strengths and weaknesses of several disentanglement techniques
4. Explore the concept of disentanglement versus the implementation of it
5. Consider the trade-offs between co-training representations versus learning them separately
6. Introduce a technique to probe speaker representations and disentangle style/emotion from speaker identity
7. Introduce and compare multiple variations of self-supervised speech re-synthesis with stacked encoders
8. Examine learned representations through intrinsic and extrinsic evaluation
9. Introduce a technique that allows for weighted voice mixing during a voice conversion task
10. Introduce a new method for speaker diarization
11. Introduce a new privacy task based on concealment of speech content
12. Introduce a novel analysis approach for disentanglement based on principles of natural language processing
13. Explore the limits of disentanglement between speaker identity and content
14. Compare and contrast evidence for and against disentanglement

1.5 Thesis Outline

Chapter 1: Introduction. This chapter introduces a plurality of disentanglement approaches from the image and speech domains. It presents open issues that are related to disentanglement for speech representations including definitions, assumptions, and evaluation. The problem statement and contributions of the thesis are discussed.
Chapter 2: Background. Relevant related work is presented and summarized.
Chapter 3: Methods to Estimate Speech Authenticity and Naturalness. Speech representations are introduced in the context of determining speech authenticity (i.e., spoofing detection). Building upon the representations for authenticity, new methods are presented for estimating synthetic speech quality and a new tool is developed for later use in Chapter 5.
Chapter 4: Methods to Disentangle Representations of Speaker Identity. Methods to disentangle information from existing speaker representations are introduced. It is shown that traditional speaker representations contain much more information than speaker identity. An autoencoder approach is used as a disentanglement technique. Several architectures are compared for disentangling style and emotion from speaker identity. The limitations of this approach are discussed.
Chapter 5: Methods for End-to-End Disentanglement. Following from the limitations that were uncovered from the experiments in Chapter 4, an end-to-end disentanglement technique is introduced. The underlying architecture is an autoencoder with vector quantization. Several versions of the architecture are compared for disentangling speaker identity, content, and F0 representations. Multilingual disentanglement is also explored.
Chapter 6: Intrinsic Analysis of Disentangled Representations The representations learned in Chapter 5 are evaluated using techniques from language modeling such as perplexity. The representations are visualized. The representations are probed for their informational content.
Chapter 7: Extrinsic Analysis of Disentangled Representations. The representations learned in Chapter 5 are evaluated using a variety of tasks from speech processing including speaker diarization, phone recognition, voice conversion, linguistic code-switching, and content privacy masking. Counter-factual tasks are presented to identify which representations perform well on tasks and which do not.
Chapter 8: Conclusion. Final thoughts and considerations are discussed.

Chapter 2

Background

2.1 Introduction

The experiments in this thesis require background knowledge in several key areas: speaker voice recognition, embeddings in speech synthesis, and speech re-synthesis. This chapter provides such a background for the reader and introduces important concepts that will be referenced in later chapters. The background is not meant to be exhaustive and the reader is encouraged to read each of the cited papers for a more thorough tutorial.

2.2 Speaker Recognition

Speaker recognition is the ability to determine the identity of a person based on the sound of their voice. Speakers can be recognized by other humans or automatically by a machine and this thesis is concerned primarily with the latter. In general, speaker recognition is a form of biometric authentication that uses features of the human voice as markers of uniqueness. Recognizing a speaker can be done using techniques that depend on the spoken content of an utterance and that is called text-dependent recognition [Heigold et al., 2016]. Text-dependent recognition requires the speaker to be identified correctly but the speaker must also be saying the correct content, as with a passphrase or password. Another form of recognition is completely independent of content and likewise called text-independent recognition [Kinnunen and Li, 2010]. In addition, speaker recognition can involve identifying a person out of a population (identification) or it can involve verifying that a person is who they claim to be (verification) [Reynolds, 1995].

Both types of content and both types of speaker recognition are relevant to the work that is presented throughout this thesis. For example the quality and authenticity experiments in Chapter 3 deal with text-independent speaker verification, and the disentanglement experiments in Chapter 5 explore the separation of speaker identity from content. This section will provide background about the types of variability that must be modeled in order to perform speaker recognition, how speaker recognition is evaluated, and an overview of anti-spoofing techniques that prevent speaker recognition failure or intrusion. While this thesis is not about Gaussian mixture models (GMMs) or factor analysis, those two topics introduce ideas and terminology that led to the creation of speaker representations from deep neural networks as well as the

techniques for generally scoring or evaluating the speaker representations that are used in this thesis. As Lee et al. [2020] describes, speaker recognition is an evaluation-driven research area and methods are traditionally developed with performance in mind. In general, speaker recognition systems are usually evaluated on standardized datasets from the U.S.-based National Institute for Standards and Technology (NIST)[1] through annual challenges and campaigns which commenced in the year 2000 [Doddington et al., 2000]. For an additional historical viewpoint of speaker recognition, the reader is encouraged to explore Campbell [1997].

2.2.1 Sources of Variability

As Hansen and Hasan [2015] explain, there are two main types of variability that must be considered when it comes to modeling individual speakers using voice as a biometric identifier. First, *intra*-speaker variability describes how speech from a single speaker may appear different depending on environmental factors or speaker affect. When dealing with intra-speaker variability, the goal is to reduce it as much as possible so that only the speaker is modeled and not the environment or affect. On the other hand, *inter*-speaker variability leads to the objective of maximizing differences among individuals as much as possible in order to observe individuals as unique speakers. Both types of variability are considered to be informational *factors*.

The human voice can exhibit a lot of variability for any given speaker. In some cases the variability is based on how a person sounds at different times of the day, the emotions that they are feeling, or health issues such as illness. Variability can also be related to the environment or technology that is used to capture and record the voice such as the microphone position and quality, and room acoustics. Intra-speaker variability can also be considered in terms of mismatch between model and test conditions, which are referred to as enrolment and test. An example is if a particular microphone is used when enrolling a speaker for model training, but then a different microphone is used when the speaker presents for recognition at test [Hansen and Hasan, 2015].

For speaker recognition, it is important to minimize or reduce all forms of intra-speaker variability as much as possible in order to achieve good recognition performance. This is a challenging task and it requires mathematical or machine learning techniques to achieve good results. The basis of minimizing intra-speaker variability is that sometimes speakers will have been recorded in different recording sessions and with varied conditions but the underlying speaker identity remains the same. In order to model a given speaker's identity, it is necessary to model information in a voice recording that remains unchanging from one session to the next. This is also challenging when the words being spoken also change from one session to the next. Dealing with variability between recording sessions is considered to be one of the most challenging issues in speaker recognition applications [Kenny, 2005; Vogt and Sridharan, 2008]. Intra-speaker variability is often referred to as *session* or *channel* variability. In this thesis the intra-speaker variability will be referred to as channel information. Methods for dealing with channel information will be described in 2.2.2.

[1] `https://sre.nist.gov/`

2.2.2 Representations for Modeling Variability

2.2.2.1 GMM Supervectors

The work of Reynolds and Rose [1995] and Reynolds [1995] established that Gaussian mixture models (GMMs) can effectively model speaker and session variability using low-level descriptors of the speech signal. The low-level descriptors include features such as cepstrum or mel-cepstrum features as well as the delta or delta-delta features that describe how the low-level descriptors change over time [Campbell, 1997]. GMMs are based on distributions of features represented by the means and variances of model parameters that are estimated from data for chosen low-level descriptors. GMMs explain observations from data. GMMs can be trained using the expectation-maximization algorithm [Reynolds and Rose, 1995] which provides an iterative way of efficiently calculating model parameters.

It is possible to formulate the problem of speaker recognition in a way that uses known data from known speakers in order to establish appropriate thresholds for deciding speaker identity. A single model can be created and used for all speakers in a task, and that model is referred to as the universal background model (UBM). In speaker recognition research, there is also an assumption that each utterance is spoken by only one speaker. Multiple speakers and multiple sessions per speaker are modeled by Gaussian distributions in the UBM. From the UBM, a GMM model can be trained that maximizes the likelihoods for speakers given the model parameters. A GMM for speaker recognition can therefore be trained from a very large UBM.

The utility and application of GMMs for speaker recognition depends on how many speakers were in the initial data as well as how similar or different those speakers are to each other. For example, a UBM with more variety of speakers and more intra-speaker variability may generalize better in unseen conditions. GMM supervectors can be created such that the values stored in a vector represent the GMM model parameters [Wan and Renals, 2003; Dehak and Chollet, 2006; Dehak et al., 2007; Dong et al., 2008]. GMMs can also be used to create *eigenvoice* representations which effectively reduces the number of parameters that must be estimated [Kuhn et al., 2000].

The GMM supervectors and resulting eigenvoice representations can be used for calculating a difference between two audio sources (e.g., two audio files containing speech), based on the features being modeled. The difference between two audio sources is used for confirming a hypothesis that the two audio sources come from the same or different speaker. Making a decision for the hypothesis requires a log-likelihood ratio. It is difficult to know or guess an appropriate log likelihood ratio threshold for confirming the hypothesis [Reynolds, 1995] so the thresholds are often determined experimentally.

2.2.2.2 Factor Analysis

One of the main techniques for dealing with variability is based on joint factor analysis (JFA) theory that was developed by Kenny [2005]. Factor analysis deals with the ability to model speakers and channel information that is found in a GMM. Factor analysis treats the GMM supervector s for an utterance as a linear combination of components as in Equation 2.1:

$$s = m + V_y + U_x + D_z \tag{2.1}$$

where the variable m represents a UBM that has been calculated from a database of speakers. The variable V_y is an eigenvoice matrix for the speaker factors y of a given speaker. The variable U_x is the eigenchannel matrix for channel factors x in a given utterance. The variable D_z is a residual matrix with residual factors z such that it contains any factors that are not speaker or channel factors. The term m is a speaker-independent representation (of all speakers in the UBM) while the term V_y is speaker-dependent for a given utterance. The term U_x is considered to represent channel information. As for the D_z term, although it is a residual, it is a speaker-dependent residual. All of these terms together define a GMM supervector according to factors of information that vary within a speaker as well as across speakers, for a given utterance. By formulating the variability in this manner, using eigenvoice and eigenchannel matrix representations, it is possible to use linear algebra to decompose sets of GMM supervectors into these factors and further reduce information related to intra-speaker variability via the channel factors while maximizing the inter-speaker variability via the speaker factors. A GMM supervector with high utility has minimized intra-speaker variability and maximized inter-speaker variability using the factor analysis techniques. For a derivation of the GMM supervector model and techniques for estimating the model parameters, see Section 1 and Section 2 of Kenny et al. [2008].

2.2.2.3 *i*-Vectors

One of the persisting challenges from GMM supervector modeling and factor analysis was that information about speaker was sometimes found in the speaker factor as well as the channel factor, as these two factors were difficult to separate completely. To address that problem, the original JFA technique [Kenny, 2005; Kenny et al., 2008] was further expanded by Dehak et al. [2009] such that the speaker factors and channel factors were treated as a single matrix representation. The single matrix was termed a *total variability* matrix and it was of lower rank than the original speaker and channel factor matrices from JFA. The expanded factor analysis from Dehak et al. [2009] is shown in Equation 2.2 defining M as a speaker- and channel-dependent GMM supervector:

$$M = m + Tw \tag{2.2}$$

where m represents the means of a UBM, the matrix T is a low-rank rectangular matrix representing variability, and w is a random vector of values following a standard normal distribution. The term w represents factors of total variability. Unlike the variability described by GMM supervector decomposition earlier in Equation 2.1, the total variability matrix M in Equation 2.2 assumes that all of the utterances from a given speaker (including multiple recording sessions) are from different speakers. The effect of treating the intra-speaker variability in this way is that factor analysis would behave as a method for feature extraction such that the term w contains a representation of a given speaker. For the full specification of the total variability model as well as the techniques for normalization please see Section 2 and Section 3 of Dehak et al. [2009]. The w representation was termed the i-vector as it represents the identity of a speaker as a vector [Dehak et al., 2011].

The i-vector model has been widely adopted for a variety of tasks. For example, it has been shown that i-vectors can be applied to language identification [Martinez et al., 2011; Li and Narayanan, 2014]. The i-vector has also been successfully adapted to model and predict channel factor characteristics such as microphone type or noise classes based on signal-to-noise ratio

(SNR) [Ferrer et al., 2012]. Later, Van Segbroeck et al. [2014] showed that i-vectors also encode useful intra-speaker characteristics such as changes in cognitive load that result from reading easy versus difficult sentences out loud. A summary of the variety of applications of i-vectors can be found in Verma and Das [2015]. i-vector representations are utilized in this thesis in Chapter 4.

2.2.2.4 x-Vectors

Neural speaker embeddings were introduced by Heigold et al. [2016] and Variani et al. [2014] to overcome issues of computational efficiency and model footprint that were problematic in the i-vector modeling approach. The benefits of developing speaker representations from neural network embeddings are that the neural embeddings scale well to larger datasets and they do not require as much heuristic parameter tuning compared to i-vectors. Heigold et al. [2016] showed that neural embeddings significantly outperformed i-vectors for speaker verification. The approach used a deep neural network that was trained to predict whether pairs of utterances were from the same or different speaker.

The original neural speaker embeddings from Heigold et al. [2016] were referred to as d-vectors but these were quickly superseded by x-vectors. The approach for d-vectors was text-dependent and therefore limiting for generalization to other phrases [Heigold et al., 2016]. The d-vectors were created in collaboration with Google for the purpose of performing speaker verification when the user interacts with their devices by saying: "Okay Google." To overcome the restrictions of text-dependent speaker representations, a text-independent approach was developed and presented in Snyder et al. [2016, 2018] and they called this text-independent approach x-vectors. The x-vector technique divided the speaker modeling task into two separate but related steps: creating neural embeddings and evaluation. This thesis deals with x-vectors in Chapter 3 and Chapter 4, and to a lesser extent they are used in Chapter 5 for evaluating similarity among speakers for synthetic speech.

In Snyder et al. [2017], a neural network architecture is shown that is used to create x-vector embeddings (re-created in this chapter as Figure 2.1). The input is a series of frames belonging to an utterance of length T frames $(x_1, x_2, ..x_T)$. The network is a time-delay neural network (TDNN) architecture because the frame-level layers utilize temporal context of $t-8$ to $t+8$ frames. The first five layers operate on the frames using a temporal context. The statistics pooling layer calculates the mean and standard deviation of activations for the final layer operating at the frame-level. After the statistics pooling, there are two fully-connected layers which operate at the segment level. The final layer of the network (during training) is the softmax layer and it is applied to estimate the probability of a speaker given the utterance frames. At inference, the softmax layer is discarded and the x-vector is represented by the activations in the network for one of the fully-connected layers. The actual x-vector embedding can be either embedding **a** or **b**. This thesis uses embedding **b** as it is shown in Figure 2.1. In this thesis, the same architecture that is described in Figure 2.1 is used for creating x-vectors that model environmental factors (rather than speakers) for evaluating speech synthesis quality and detecting speech spoofing attacks in Chapter 3.

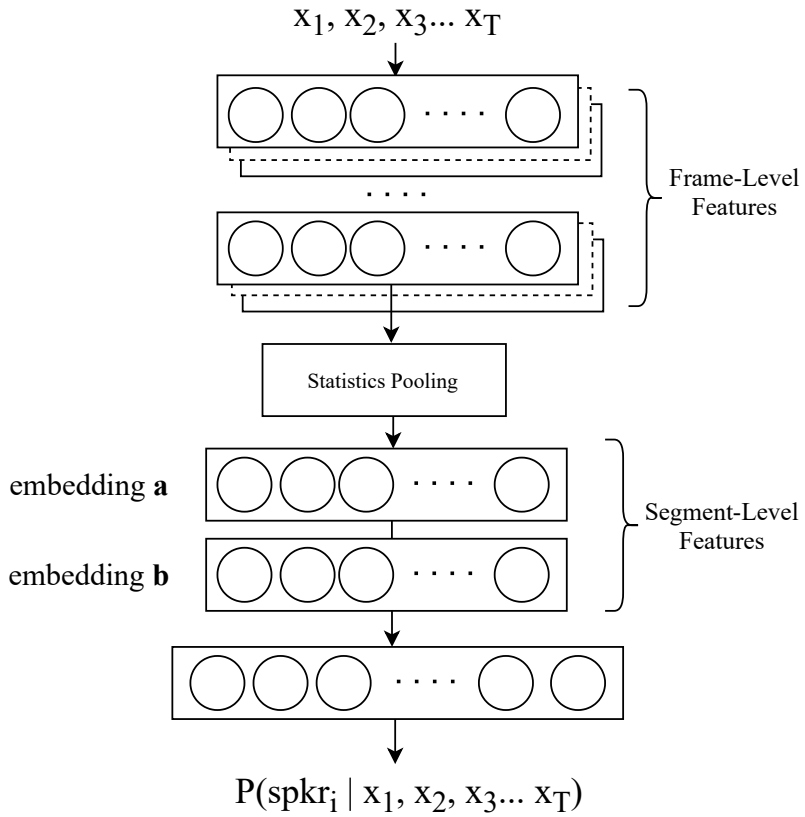


Figure 2.1: The time-delay neural network (TDNN) architecture from Snyder et al. [2017] for creating an x-vector embedding (either embedding **a** or **b**). The input is a series of frames belonging to an utterance of length T frames $(x_1, x_2, ..x_T)$. The output of the network (during training) is the softmax that estimates the probability of a speaker given the utterance. At inference, the softmax layer is discarded and the x-vector is represented by the activations of embedding **a** or **b**.

2.2.2.5 Encoded Information

It was previously mentioned in Section 2.2.2.3 that i-vectors can encode a variety of information from the speech signal such as cognitive load [Van Segbroeck et al., 2014]. Further studies have shown that both i-vectors and x-vectors are encoding a variety of information that is not specifically relevant to speaker recognition [Raj et al., 2019; Peri et al., 2020a; Moro-Velazquez et al., 2020]. The information encoded in speaker representations is further demonstrated by the recently organized VoicePrivacy Initiative which seeks to develop methods that remove identifying attributes of speaker identity from speech for privacy reasons [Tomashenko et al., 2020]. The basis for the work presented in this thesis is that multiple types of information can be encoded in i-vectors and x-vectors. The extra information is unwanted because it interferes with speaker recognition. However in this thesis, the experiments presented in Chapter 4 explore an alternative viewpoint that the information encoded in i-vectors and x-vectors could be separated or disentangled from other informational factors such as speaker factors, and then retained as an altogether new representation.

Work from Raj et al. [2019] showed that i-vectors and x-vectors also encode speaker gender, speaking rate, words, phonemes, length of utterance, and the type of noise used for data augmentation. In order to establish that the i-vectors and x-vectors contain such information, Raj et al. [2019] formulated a probing task. The i-vectors and x-vectors were treated as inputs to a machine learning classification task using a multi-layer perceptron (MLP). They report classification accuracy for each type of probing task as well as how the accuracy varies based on the size of the i-vector and x-vector dimensionality (128, 256, 512, 768). Their primary objective was to understand if data augmentation during the process of i-vector and x-vector training and extraction could improve speaker recognition performance. They conclude that using noise augmentation during the training and extraction process influences the i-vectors and x-vectors to encode more information relevant to speaker recognition and less of the irrelevant information. Another major finding from Raj et al. [2019] was that i-vectors encode more information about recording session, speaker gender, and words compared to x-vectors. In addition, the x-vectors performed better on their text-dependent speaker recognition task. Taken together, this evidence suggests that x-vectors are a better representation of speaker identity.

Shortly after the disentanglement experiments in Chapter 4 of this thesis commenced, Peri et al. [2020a] and Peri et al. [2020b], attempted to disentangle information encoded in x-vectors to arrive at better representations of speaker identity so that speaker recognition performance could be improved. The disentanglement method is described in Peri et al. [2020b] as an encoder-decoder system with two auxiliary prediction tasks that attempt to predict information from the learned latent space. The technique was evaluated using a probing classification task similar to the one described in Raj et al. [2019]. Disentanglement was evaluated based on a loss or gain of classification accuracy over channel factors (room size, microphone type, noise type), content factors (emotion, sentiment, lexical, and language), and speaker factors (identity label, gender). Their disentanglement method showed a small loss of accuracy for non-speaker factors and a small gain of accuracy for speaker factors. The work did not provide a definition of *disentanglement* and the probing classification task, and gains or losses of accuracy, were the only evaluation of the disentanglement claims. This thesis contributes new principles of disentanglement and assesses disentanglement through both intrinsic and extrinsic tasks in Chapter 6 and Chapter 7 respectively.

2.2.3 Speaker Verification Anti-Spoofing

Alongside the development of speaker recognition technology and improvements from GMM supervectors to i-vectors, there were some security vulnerabilities for speaker recognition that were presented in the work of Faundez-Zanuy [2004]. In their work, eight different types of biometric verification attacks (including those on voice) were outlined. The types of attack ranged from sensor-level where an attacker could record and replay the voice of an authorized user to the decision-level where the final authentication system can be overridden. All of the attack types were categorized into two main groups: communication channels and system modules. The attacks involving communication channels exploit the fact that data is in transit or being transmitted in physical space from one module to another. The attacks involving system modules involve manipulating components of the module and therefore require some insider knowledge of the system module configurations.

More recently, speaker verification anti-spoofing has become a very active research area [Wu et al., 2015b; Todisco et al., 2016; Sahidullah et al., 2015; Wu et al., 2017; Todisco et al., 2019] with biennial international challenges starting in 2015 and continuing to the present [Wu et al., 2015a]. The international challenge is called Automatic Speaker Verification Spoofing Challenge (ASVspoof)[2]. This challenge has been held in 2015, 2017, 2019, and 2021[3]. Only the 2019 challenge tasks and data are used in this thesis, but references for the other years challenges are provided here.

The first challenge in 2015 offered a dataset of spoofed and bonafide speech samples where the spoofed were generated using text-to-speech synthesis or voice conversion [Wu et al., 2015b]. The 2017 challenge [Delgado et al., 2018] introduced an additional type of attack called "replay" as well as a baseline system using a new feature called constant-Q cepstral coefficient (CQCC) with Gaussian mixture model [Todisco et al., 2016, 2017]. The 2019 challenge [Todisco et al., 2019] introduced a second baseline [Sahidullah et al., 2015], a new evaluation metric (discussed below in 2.2.3.3) [Kinnunen et al., 2018] and improved datasets [Wang et al., 2020b]. The 2021 challenge introduced a third task involving deepfakes [Nautsch et al., 2021].

This thesis explores representation learning for speaker verification anti-spoofing in Chapter 3.2 and utilizes similar representations for speech synthesis naturalness estimation in Chapter 3.3. The experiments in Chapter 3.2 deals with detecting physical access spoofing attacks in the 2019 challenge. The experiments in Chapter 3.3 utilize the dataset from 2019 logical access spoofing attacks.

The datasets for the 2019 physical access and logical access are described below, along with two of the evaluation metrics from the 2019 challenge. Two main types of attacks were considered in the challenge: Logical Access (LA) and Physical Access (PA). In both tasks, the datasets are derived from VCTK corpus[4], comprising 107 speakers (46 males, 61 females). The speakers were partitioned into three sets: training, development, and evaluation with non-overlapping speakers. Each partition contained 20, 10, and 48 unique speakers, respectively.

[2] `https://www.asvspoof.org/`

[3] At the time of this writing, the 2021 challenge and related workshop are underway but the evaluation datasets have not been publicly released.

[4] https://datashare.is.ed.ac.uk/handle/10283/2651

2.2.3.1 Physical Access Attacks

The physical access (PA) spoofing attacks that are part of the ASVspoof challenges correspond to attacks involving communication channels first described by Faundez-Zanuy [2004]. The nature of this attack is based on the idea that speech from a valid authentic speaker has been recorded and then replayed (perhaps out of context) to a speaker recognition system. The 2019 PA dataset is described in detail in [Wang et al., 2020b].

The speech samples in the 2019 PA dataset have simulated conditions to ensure precisely controlled experimental variables. Different environments (determined by room size, reverberation time, talker-to-ASV distance) and different attacks (determined by attacker-to-talker distance, replay device quality) are the variables that are considered in the simulation. The environment is defined using a triplet S, R, D_s where S is the room size, R is the reverberation time (T60), and D_s is the distance between the talker and the ASV system. Each information type in this triplet can take on one of three categorical values as shown in Table 2.1. The attack is characterized using a duple D_a, Q where D_a is the distance between the attacker and talker, and Q is the replay device quality. The variables for attack type are described in Table 2.2. Each datum in the PA challenge dataset consists of a sound file as well as these triple and duple labels indicating if the speech file represents an attack (spoofed) or not (bonafide).

	labels		
Environment Variable	a	b	c
S: Room size (m^2)	2-5	5-10	10-20
R: T60 Reverberation (ms)	50-200	200-600	600-1000
D_s: Talker-to-ASV distance (cm)	10-50	50-100	100-150

Table 2.1: Environment variable types and values in the ASVspoof 2019 Physical Access (PA) challenge dataset. The environment variables are represented by a triple (S, R, D_s) where each variable can take a categorical value a, b, c

	labels		
Attack Variable	A	B	C
D_a: Attacker-to-talker distance (cm)	10-50	50-100	>100
Q: Replay device quality	Perfect	High	Low

Table 2.2: Attack variable types and values in the ASVspoof 2019 Physical Access (PA) challenge dataset. The attack variables are represented by a duple (D_a, Q) where each variable can take a categorical value A, B, C

2.2.3.2 Logical Access Attacks

The logical access (LA) spoofing attacks are considered to be a type of *impersonation attack* because this type of attack involves text-to-speech (TTS) synthesis and voice conversion (VC) [ISO/IEC JTC1 SC37 Biometrics, 2017]. In this thesis, the 2019 LA dataset is used for evaluating speech synthesis naturalness rather than for anti-spoofing. The LA dataset is described in detail in [Wang et al., 2020b]. One of the benefits of this dataset is that it contains a mixture of

Table 2.3: Overview of TTS/VC systems and their identifiers in the ASVspoof 2019 Logical Access (LA) dataset. Each system is identified with relevant citation and the training or validation set where it was used.

System ID	TTS/VC	Partition	Reference(s)
A01	TTS	Train/Valid	Zen et al. [2013]; van den Oord et al. [2016]
A02	TTS	Train/Valid	Zen et al. [2013]; Morise et al. [2016]
A03	TTS	Train/Valid	Wu et al. [2016]; Morise et al. [2016]
A04	TTS	Train/Valid	Schröder et al. [2011]; Steiner and Le Maguer [2018]
A05	VC	Train/Valid	Hsu et al. [2016]; Huang et al. [2018]; Morise et al. [2016]
A06	VC	Train/Valid	Matrouf et al. [2006]
A07	TTS	Eval	Wu et al. [2016]; Tanaka et al. [2019]
A08	TTS	Eval	Wang et al. [2019a]
A09	TTS	Eval	Zen et al. [2016]
A10	TTS	Eval	Shen et al. [2018]; Jia et al. [2018]
			Wan et al. [2018]; Kalchbrenner et al. [2018]
A11	TTS	Eval	Shen et al. [2018]; Jia et al. [2018]
			Wan et al. [2018]; Griffin and Lim [1984]
A12	TTS	Eval	van den Oord et al. [2016]
A13	TTS+VC	Eval	Li et al. [2015]; Kobayashi et al. [2018]
A14	TTS+VC	Eval	Liu et al. [2018]; Kawahara et al. [1999]
A15	TTS+VC	Eval	Liu et al. [2018]
A16	TTS	Eval	Schröder et al. [2011]; Steiner and Le Maguer [2018]
A17	VC	Eval	Hsu et al. [2016]; Huang et al. [2018]
			Kobayashi et al. [2018]
A18	VC	Eval	Kinnunen et al. [2017]

different types of speech synthesis systems. The LA dataset is divided into 3 partitions consisting of training, validation, and held-out evaluation data. In the training and validation set, the same 4 TTS and 2 VC systems are used in these two partitions. In the held-out evaluation set, a different set of 7 TTS and 5 VC systems are used and these systems are unique to the evaluation set. Table 2.3 shows each of the system IDs for the entire LA dataset as well as which partition each system belongs to and the relevant citation for the system design including separate vocoder where appropriate.

2.2.3.3 Anti-Spoofing Evaluation

There are two metrics that are relevant to speaker recognition anti-spoofing in this thesis: equal error rate (EER) [Wu et al., 2015b] and tandem detection cost function (tDCF) [Kinnunen et al., 2018]. Both of these metrics are used in Chapter 3.2 to evaluate representation learning for speech replay spoofing detection. The EER metric provides a single value that describes the error rate at which the false alarm rate (FAR) and false reject rate (FRR) are both equal. The

FAR is a monotonically decreasing function and the FRR is a monotonically increasing function, both based on θ where θ is the threshold setting for deciding that an item is spoofed or bonafide. The threshold setting can be set to any value, as it provides a customized setting for real-world systems that may desire a higher FAR or higher FRR based on the specific application [Wu et al., 2015b]. The FAR is shown in Equation 2.3 and the FRR is shown in Equation 2.4. When using the EER to judge the quality of a spoofing countermeasure, lower EER values are better because this indicates that the countermeasure achieves the lowest possible number of misses and false alarms when the FAR and FRR are equal.

$$FAR(\theta) = \frac{\#\text{spoofed trials with score} > \theta}{\#\text{total spoofed trials}} \tag{2.3}$$

$$FRR(\theta) = \frac{\#\text{genuine trials with score} \leq \theta}{\#\text{total genuine trials}} \tag{2.4}$$

While the EER of an anti-spoofing countermeasure system can assess the countermeasure in a standalone fashion, it does not reflect the real-world use case which depends on an ASV system. The tDCF metric is an extension of the detection cost function (DCF) that was originally used for evaluating speaker recognition systems [Doddington et al., 2000]. The DCF provides a single value that evaluates the performance of automatic speaker verification without spoofing. By extending the DCF to tDCF, then it is possible to obtain a single value that evaluates the performance of spoofing countermeasures that operate in tandem with an automatic speaker verification system. The tandem performance of both the spoofing countermeasure and the speaker verification system reflect the real-world use case.

$$\begin{aligned} tDCF(s,t) = & C_{miss}^{asv} \cdot \pi_{tar} \cdot P_a(s,t) \\ & + C_{fa}^{asv} \cdot \pi_{non} \cdot P_b(s,t) \\ & + C_{fa}^{cm} \cdot \pi_{spoof} \cdot P_c(s,t) \\ & + C_{miss}^{cm} \cdot \pi_{tar} \cdot P_d(s) \end{aligned} \tag{2.5}$$

The tDCF metric is based on four main types of trials:

- **a**: target trial miss, CM system accepts, ASV system rejects
- **b**: non-target trial false accept, CM system accepts, ASV system accepts
- **c**: spoof trial false accept, CM system accepts, ASV system accepts
- **d**: target trial miss, CM system rejects, ASV system pass

The term *trials* is used here to describe positive and negative exemplars that are passed to the tandem ASV+CM system to explore how it handles various errors. Target and non-target trials refer to the expected decision by the ASV system whereas a spoof trial refers to the expected decision by the CM system. A target trial means that the speaker presented for verification in the ASV system is the correct speaker (and should be accepted) whereas a non-target speaker presented to the ASV system is an incorrect speaker (and should be rejected). Likewise a spoof trial should be rejected by the CM system. Some spoofing attacks, such as replay attacks, imply

that it is possible to present a spoofed trial of a target speaker but in this case the trial should be rejected by the CM system even though it is a target speaker (type **d**).

Equation 2.5 shows the tandem detection cost function considering an automatic speaker verification system (ASV) as well as a spoofing countermeasure system (CM). The details of this function can be found in Kinnunen et al. [2018]. The tDCF calculation that is shown in Equation 2.5 reflects a single value for the system s and a decision threshold t (previously referred to as θ in this thesis when describing EER). The terms C_{miss}^{asv}, C_{fa}^{asv}, C_{fa}^{cm}, C_{miss}^{cm} respectively represent the cost of an ASV system rejecting a target trial, the cost of an ASV system accepting a non-target trial, the cost of a CM system rejecting a bonafide trial, and the cost of a CM system accepting a spoofed trial. The term π represents the priors. For example, π_{tar}, π_{non} and π_{spoof} each represent the priors for trials that are target, non-target, and spoof. The terms for P represent each of the four types of trials (**a**, **b**, **c**, **d**). As with the EER, a lower tDCF score is preferred since a lower tDCF score indicates fewer errors between the ASV and CM systems.

2.3 Embeddings in Speech Synthesis

This section discusses some of the recent advances related to representation learning for speech synthesis, including voice conversion (VC) and text-to-speech (TTS) synthesis. When neural TTS synthesis was first introduced by Zen et al. [2013] and Lu et al. [2013], it was difficult to harness the power of multi-speaker datasets or to create speech output sounding like a specific speaker. In order to create speech for multiple speaker voices, it was necessary to train a TTS system on speaker-specific data (if available) or train an "average" voice on multiple speakers and then apply a separate post-processing step, such as voice conversion or speaker-adaptation [Yamagishi and Kobayashi, 2007]. Another technique was developed thereafter that involved using a speaker "code" during training to assist TTS training to learn that some of the training data belonged to different speakers such as in the work of Luong et al. [2017]. In their work, they experimented with one-hot, numeric and discriminant condition codes from the field of speaker recognition [Xue et al., 2014] for both multi-speaker TTS and speaker-adaptation. Luong et al. [2017] found that one-hot speaker encodings resulted in better naturalness but the discriminant condition codes resulted in higher speaker similarity. The discriminant condition codes were modeling speaker characteristics in a way that was not possible with one-hot vectors because the representations were richer and accounted for more nuanced variation.

Embeddings in speech synthesis are relevant to this thesis because they involve learning and using representations that model information such as speaker, gender, age [Luong et al., 2016] emotion [Akuzawa et al., 2018] or speaking style [Wu et al., 2018]. There are generally two ways that learned representations are treated in speech synthesis:

- representations are learned internally to a speech synthesis system and trained concurrently
- representations are learned externally and then incorporated into a speech synthesis system and used for training speech synthesis

The following sections (2.3.1 and 2.3.2) provide several specific examples of how representation learning has been used in speech synthesis. In this thesis, representations that are learned in Chapter 5 are later used in downstream speech synthesis tasks such as voice conversion and multilingual speech synthesis in Chapter 7.

2.3.1 Voice Conversion

Voice conversion is an area of research that involves converting speech from a source speaker into the voice of a target speaker, without modifying the content information. A full and recent survey on voice conversion can be found in Sisman et al. [2020]. The research area of voice conversion has recently been accelerated by the introduction of international challenges, starting in 2016 [Wester et al., 2016; Toda et al., 2016] and continuing every two years with challenges in 2018 [Lorenzo-Trueba et al., 2018] and 2020 [Zhao et al., 2020a]. These challenges provide an opportunity for researchers to build competing systems and contribute new techniques to the field. The effort is growing as the 2016 challenge had 17 participants, the 2018 challenge had 23 participants, and the 2020 challenge had 30 participants. Each participant can be a company or academic team or individual. Most of the system entries in the 2020 Voice Conversion Challenge were based on neural networks and four systems used an encoder-decoder model [Tomashenko et al., 2021]. The encoder-decoder models learn latent space representations and these representations can be passed to the decoders in order to control the speaker identity.

Separate from the biennial voice conversion challenges, there has been recent work on using speaker representations in neural vocoders for synthesising speech from either very few training examples (few-shot) [Liu et al., 2018], one training example (one-shot) [Wu and Lee, 2020] or no training examples (voice cloning) [Arık et al., 2018]. In all of these cases, representations that model speaker identity are shown to be helpful for speech synthesis and result in quality voice conversion with less training data requirements to generalize to unseen speakers.

Huang et al. [2020] suggests that voice conversion success comes from disentanglement between speaker identity and spoken content and that more disentanglement results in better voice conversion. They describe that it is problematic if there is residual speaker information from the source speaker being propagated through their system since it has the effect of mixing speaker information between the source and target speakers. They propose a VC system that is based on a variational autoencoder (VAE) where the VAE learns to generate speech parameters, such as those obtained or derived from features using the WORLD vocoder analysis (spectral envelope, mel-cepstrum coefficients, fundamental frequency and aperodicity) [Morise et al., 2016]. There is one adversarial discriminator for each type of speech parameter. A speaker classifier is applied to the latent space of the VAE and is used to encourage disentanglement of phonetic information and speaker information.

The work of Huang et al. [2020] uses a measure of disentanglement that was first introduced in Huang et al. [2019]. The disentanglement evaluation involves measuring a distance, such as cosine distance, between latent speaker codes from the trained VAE. A pair of parallel sentences is taken from the source and target speaker such that the phonetic content is the same. Then the distance between latent speaker codes for source and target speaker is calculated. If the disentanglement is very good then the source and target speaker codes will have a high similarity (or low distance). A similar approach to evaluating disentanglement is provided in this thesis in Chapter 6 using data probing techniques to assess if speakers saying the same content are similar or different. A cosine similarity measure is also used in Chapter 5 to assess whether a given speaker code is consistent for a given set of utterances.

2.3.2 Text-to-Speech Synthesis

This section discusses representation learning for two main issues in text-to-speech (TTS) synthesis: multi-speaker synthesis and expressive synthesis. Both of these areas of TTS research have extensive histories spanning several decades [Pitrelli et al., 2006; Yamagishi et al., 2008, 2010; Schröder, 2009; Govind and Prasanna, 2013; Liu and Mak, 2020]. This section highlights several examples of how representations can be used in TTS when those representations are either internally or externally trained.

2.3.2.1 Multi-Speaker Speech Synthesis

A well known multi-speaker TTS synthesis system is Deep Voice 2 [Gibiansky et al., 2017] which is built on top of the original Tacotron TTS system [Wang et al., 2017a] but includes an additional speaker modeling component. The speaker modeling is based on very low dimensional representations (16-*dim* or 32-*dim*) that model individual speakers in the training data. The representations are located in each module throughout the deep neural network (segmentation, duration, frequency, vocal and character-to-spectrogram modules). The representations are learned during system training simultaneously as with the TTS synthesis components. The Deep Voice 2 system was evaluated on two different multi-speaker datasets consisting of 108 speakers and 477 speakers each. In both sets of experiments, the speech output quality was shown to be comparable to a natural audio baseline and importantly the voices learned by the system were distinct and distinguishable in a separate held-out classification task.

The work of Jia et al. [2018] further expanded the idea of using speaker embeddings in TTS. They introduced a TTS system based on three separately trained modules: a speaker encoder network, a synthesis network, and a vocoder network. The purpose of the speaker encoder is to learn to encode speaker identity from waveform input into an embedding that can be used in the synthesis network (specifically the speaker embedding is concatenated to the synthesizer encoder output at each time step). The type of speaker embedding is based on d-vectors [Heigold et al., 2016; Variani et al., 2014] (mentioned earlier in Section 2.2.2.4 of this chapter). The TTS synthesis was evaluated on multi-speaker data from VCTK [Yamagishi et al., 2019] as well as LibriTTS [Zen et al., 2019]. Their experiments show that the speaker embeddings helped achieve high quality naturalness and speaker similarity on both datasets. In addition, they performed a cross-domain experiment by training on one dataset (such as VCTK) and testing on the other dataset (such as LibriTTS). The listening test results showed significant robustness for naturalness and speaker similarity for unseen speakers when trained and tested across mismatching datasets.

Another example of using externally-trained speaker embeddings for TTS synthesis comes from the work of Cooper et al. [2020]. They compared i-vectors, x-vectors and learnable dictionary encodings (LDE) [Cai et al., 2018] in a Tacotron TTS architecture [Wang et al., 2017a] with WaveNet vocoder [van den Oord et al., 2016]. From their experiments of generating speech for multiple speakers, they found that LDE speaker representations resulted in the best naturalness and speaker similarity judgements from human listeners. Furthermore, the LDE representations allowed for zero-shot speaker adaptation where the target speaker voice was not seen during training by components of the TTS system (in this case Tacotron and WaveNet).

The advances in multi-speaker TTS that have been described in this section highlight an interesting shift for TTS research. In particular, speaker representations allow for training a TTS

system on more data with more speakers, such as with the LibriTTS corpus [Zen et al., 2019] of more than 2,400 speakers. Rather than achieving TTS synthesis of an "average" voice [Yamagishi and Kobayashi, 2007], the TTS synthesis output can be made to sound like a specific speaker. As the work from Cooper et al. [2020] and Wang et al. [2020a] has shown, speaker representations can also enable a TTS synthesis system to generate speech in a speaker that was unseen during training or only seen few times. The LibriTTS corpus is used in this thesis for training a multi-speaker TTS system and evaluating synthetic speech naturalness in Chapter 3.

2.3.2.2 Expressive Speech Synthesis

Another area of speech synthesis where representation learning has been very helpful is expressive speech. In Wang et al. [2018], a type of representation was introduced called *global style tokens* (GSTs). The GSTs are embeddings that are learned jointly during training of a TTS system such as Tacotron. The GSTs do not require any labels of speaking style yet they learn to represent latent factors that correspond to speaking style. They can also be used for learning to represent speaker identity or noise depending on the characteristics of the training data. An expansion of GSTs was proposed by Kwon et al. [2019] using emotional speech. Other work, such as that in Stanton et al. [2018] showed that GSTs can be controlled in a manner that disentangles information such as style, content, speaker identity and noise - however they are not claiming to disentangle all of these factors at the same time concurrently. In each case, the speech synthesis quality and disentanglement was demonstrated as an improvement over a baseline such as Tacotron (without GSTs) and they did not propose an evaluation scheme for assessing disentanglement. Instead the term *disentanglement* is used in a way that could be approximated by saying "informational factors distinguished in a way that listeners could notice". In this thesis, experiments in Chapter 4 explore the separation of speaking style and emotion from representations of speaker identity.

2.4 Speech Re-Synthesis

In order to perform experiments about disentangled speech representations in this thesis, there must first be a method or paradigm for learning representations of speech. This section introduces the main speech re-synthesis architecture that will be used in this thesis: vector quantized variational autoencoder (VQ-VAE) [van den Oord et al., 2017]. Vector quantization (VQ) had its origins in the domain of signal compression as a technique to reduce the amount of bits required to represent information such as when transmitting a signal that contains voice data [Gersho and Gray, 1991]. VQ was also used in speaker recognition as precursory technique to GMMs [Burton, 1987; Soong et al., 1987]. In fact, a dual encoder VQ-VAE has been proposed recently for speech compression to overcome issues of intelligibility after speech has undergone high levels of compression and reconstruction [Gârbacea et al., 2019].

The original VQ-VAE system architecture from van den Oord et al. [2017] is shown at a high-level in Figure 2.2. It consists of three main components: an encoder module, a VQ dictionary (referred to as codebook in this thesis), and a decoder. The encoder will be discussed in this chapter in 2.4.1, the VQ codebook will be discussed in 2.4.2, and the decoder will be discussed in 2.4.3. The decoder takes global and/or local conditions. In van den Oord et al. [2017], the decoder was WaveRNN and the global conditions were a one-hot vector of speaker

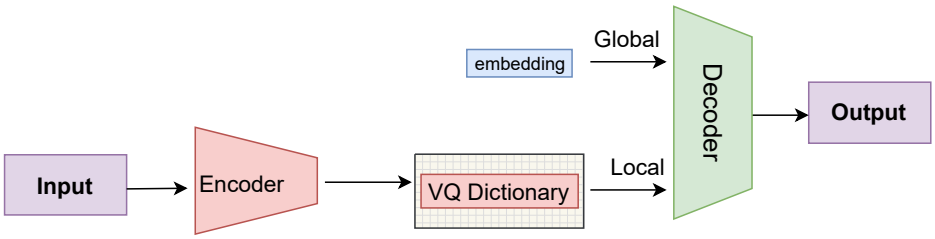


Figure 2.2: Original VQ-VAE system from van den Oord et al. [2017]: single phone encoder/VQ, and one-hot speaker vector for global conditioning with WaveRNN as the decoder. This system is used for experiments and analysis in Chapter 5, Chapter 6, and Chapter 7 where it is referred to as **VQ-VAE** and distinguished from variants that were developed in this thesis for experimentation.

labels (when multi-speaker training data was used). In the work of Zhao et al. [2020b], the decoder was also WaveRNN and global conditions were either one-hot vectors (for speaker labels, emotion, and gender) or LDE vectors (for speaker labels). Apart from any labels being passed to the decoder as global conditions, the VQ-VAE training procedure is fully self-supervised. The input is a waveform that is passed to the encoder and the output is a waveform that is generated by the decoder.

VQ-VAE was chosen for this thesis for several reasons. At the time that experimentation began for this thesis, the VQ-VAE paradigm was a promising but under-explored architecture for speech representation learning. It had been used by van den Oord et al. [2017] and Zhao et al. [2020b] and shown to produce high-quality re-synthesized speech using a neural vocoder for English, Chinese, and Japanese for both single-speaker as well as multi-speaker data. The original system from van den Oord et al. [2017] as well as the modifications from Zhao et al. [2020b] were fully self-supervised and did not require any labeled training data. Those characteristics made the VQ-VAE architecture interesting for this thesis because unsupervised and self-supervised learning was just starting to gain traction in the speech research community around the same time that experiments in this thesis commenced [Dhariwal et al., 2020; Locatello et al., 2019]. VQ was becoming popular for voice conversion [Wu et al., 2020; Wu and Lee, 2020] and WaveRNN was also being applied to voice conversion [Zhou et al., 2018]. In terms of speech representation learning, there was some work originating at the start of this thesis that explored various forms of "disentanglement" [Qian et al., 2020; Dhariwal et al., 2020; Hu et al., 2020; Jati and Georgiou, 2019; Luo et al., 2019] even if each of those works did not fully explore the meaning of the term *disentanglement*.

There was a robust and reliable code base freely available and conveniently implemented in PyTorch[5,6] along with the WaveRNN neural vocoder as the decoder. The implementation facilitated experiments that involved modifying the architecture for new capabilities, such as modeling speaker characteristics in Chapter 5 of this thesis. The existing implementations also provided system baselines for comparison and the ability to freeze particular parameters such as the size of the learned representations and features of the learned codebook dictionaries that will be described later in this section.

The work of Zhao et al. [2020b] showed that it was possible to extend the original model

[5] `https://github.com/mkotha/WaveRNN`
[6] `https://github.com/nii-yamagishilab/Extended_VQVAE`

from van den Oord et al. [2017] so that two types of representations are learned at the same time but in separate VQ codebooks. Those findings from previous work inspired the experimental approach in this thesis except the modifications to the architecture that will be described in Chapter 5 additionally explore global conditioning for WaveRNN based on learned utterance-level representations. The experiments that will be presented in Chapter 5 will identify the "best-performing" system variants for speech re-synthesis, and the learned representations from the system variants will later be analyzed and utilized in Chapter 6 and Chapter 7 respectively.

The learning process for VQ-VAE is governed by three terms in the loss function which balance the encoder, decoder, and VQ codebook (Equation 2.6).

$$L = L_R + \alpha L_{VQ} + \beta L_C \tag{2.6}$$

First, the L_R term is the reconstruction loss for the decoder, defined as $-\log p(x|z_q(x))$ which is the negative log-likelihood of decoder output x given the decoder input $z_q(x)$. The decoder input $z_q(x)$ consists of the encoder output $z_e(x)$ after quantization q. The second term L_{VQ} is the loss function for the VQ codebook, defined as $\|\text{sg}[z_e(x)] - e\|_2^2$ and it is an l_2 loss which guides VQ embedding vectors e (i.e., centroids) towards the encoder output $z_e(x)$. The sg term is a stop-gradient operator which effectively creates a non-updated constant. The purpose of the L_{VQ} term is to ensure that embeddings are also guided by reconstruction loss. Finally, the L_C term is a commitment loss defined as $\|z_e(x) - \text{sg}[e]\|_2^2$ to ensure that the encoder commits to a VQ embedding vector e and constrains how the VQ space is trained. Without any constraints on the VQ space during training, an arbitrarily large number of VQ embeddings may be active during training if the VQ embedding vectors e do not train as fast as the encoder. Therefore the commitment loss L_C term serves the purpose of encouraging meaningful VQ embeddings to be learned during training. The weights α and β can be tuned in order to balance the values of the loss function to ensure that no single component over-trains relative to the other components. The first term (L_R) is optimized by the decoder, the second and third terms (L_{VQ} and L_C) are optimized by the encoder, and the third term (L_{VQ}) optimizes the VQ embedding vectors [van den Oord et al., 2018].

2.4.1 Encoder Architecture

For the encoder architecture in VQ-VAE, convolutional layers are used to downsample the waveform input and compress it. There are 10 downsampling blocks where each block is a 1D convolution layer followed by another 1D convolution layer and a gated activation layer using tanh and sigmoid functions. The outputs of gated activation are further filtered using 1D convolution. The number of channels is set to 256 for the first convolution layer, and 128 for the second convolution layer. The final 1D convolution is also set to 128 channels. An additional residual connection is used for the block input and output. The compression ratio of the encoder is determined by the stride of the convolution layers and the number of layers. So if the stride is $s = 2$ and this stride is used in 6 of the 10 convolutional layers, then the compression ratio is $comp = 64$. This value for compression ratio refers to how the informational representation changes from being the quantity of audio samples for a file (e.g., 44800 samples at 16 kHz) to the quantity of VQ codes in a sequence that represent the same information (e.g., 700 VQ codes). Therefore an audio file that is 2.8 seconds of duration may be represented as

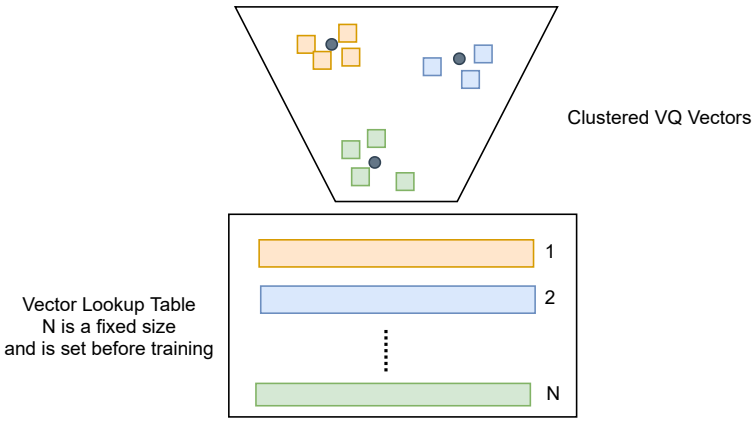


Figure 2.3: Embeddings inside of a VQ codebook correspond to the centroids of clusters that are learned during training. Each of the codebook entries is a centroid. The use of centroids in VQ-VAE adds additional structure to the latent space

44800 samples (uncompressed) or 700 VQ codes (compressed) from the downsampling rate of $N/T = 250$ codes/second.

2.4.2 Learnable Vector Quantized (VQ) Codebooks

This section describes the representations that are learned in the VQ codebook. Figure 2.3 shows how the vector lookup table corresponds to cluster centroids that were learned in training. The cluster centroids represent groupings of the VQ codebook atoms. In the VQ-VAE architecture used in this thesis, the VQ codebook atom sizes were fixed to 128-*dim* vectors for all VQ-VAE system variants and all codebooks. The atom size was set and kept constant to minimize the variable space and focus on exploring disentanglement effects.

During training, the L_C term from Equation 2.6 influences how the codebook space it utilized because it forces the output of the encoder to map to a VQ embedding vector. By adjusting this commitment loss, such as setting a low weight so that this loss is less influential during training, the amount of VQ embeddings utilized by the VQ-VAE system can be controlled. The analysis in Chapter 6 will discuss how codebooks were utilized in practice, given that they were set to a fixed size where the codebook size corresponds to the number of potential VQ embeddings that the encoder could commit to. If the VQ codebooks are not being fully utilized[7] then it is likely because the weighting of the L_C commitment loss is too high. Further details of the group latent embeddings can be found in Ding and Gutierrez-Osuna [2019]. They first proposed the idea of using groupings in VQ codebooks to add additional (unsupervised) structure to the latent space which is important especially for self-supervised learning.

Figure 2.4 shows the concept of the VQ codebook as a lookup table. Each VQ codebook corresponds to a lookup table where the key is an integer and the value is a vector of size 128-*dim*. Earlier, Figure 2.3 showed a codebook of size N, where each code in the index $[1, 2..N]$ has a corresponding vector. A single utterance of speech is represented by a sequence of phone codes (and by extension a sequence of vectors), as demonstrated in Figure 2.4. The length of the sequence of phone codes corresponds to the downsampling factor from the encoder as well as the

[7]For example, 256 VQ embeddings are allocated at the start of training but only 10% of them are ever active during training.

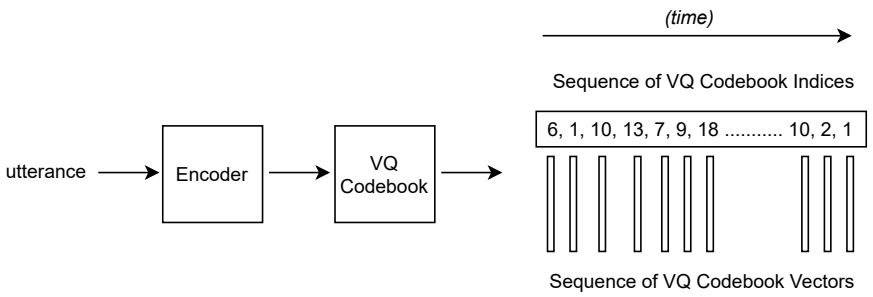


Figure 2.4: Concept of obtaining a sequence of codebook indices (*codes*) versus a sequence of codebook vectors (*vectors*) that can be passed further to the decoder as global or local conditions, or used for analysis in this chapter.

length of the original utterance. In this thesis, the codebook index keys are called *codes* and the codebook vectors are called *vectors*. The vectors are found using VQ clustering during system training, and the vectors are also provided to the decoder as either global or local conditions.

2.4.3 WaveRNN Decoder Architecture

The decoder that is used in the VQ-VAE architecture in this thesis is based on WaveRNN [Kalchbrenner et al., 2018]. WaveRNN is an auto-regressive (AR) model which means that it is a feed-forward generative sequence model that requires supervision to train. At each time step, WaveRNN utilizes input from the previous timestep as a history and it is used as input directly to the current timestep (rather than as a hidden state in RNNs). The supervision aspect of WaveRNN in VQ-VAE comes from a multi-class cross-entropy loss using a dual softmax layer that predicts a fine and coarse waveform representation. In the context of VQ-VAE, the fine and course waveform representations from WaveRNN are compared to the input speech and a negative log-likelihood (NLL) loss is calculated. The NLL loss corresponds to the term L_R in Equation 2.6. WaveRNN consists of three components: upsampling blocks, downsampling blocks, and the WaveRNN module [Kalchbrenner et al., 2018].

Upsampling Block: Time scales for the code vectors and waveform are different from each other because the encoder downsamples the code vectors. Therefore upsampling blocks are used in the decoder so that code vectors (local conditions) are at the same operating rate as the waveform points. Each upsampling block is a gated recurrent unit (GRU) layer followed by a transposed convolution network.

Downsampling Block: The downsampling block of WaveRNN uses the coarse output waveform (i.e., lower sampling rate) as an additional condition for predicting the next waveform point. To do this, the past coarse components are downsampled and combined with the next code vectors in a sequence. There are three downsampling blocks. The output of each one is connected to the corresponding upsampling block like a U-net [Ronneberger et al., 2015]. For the AR feedback, a teacher-forcing strategy is used for training, and waveforms are generated.

WaveRNN Module: This module uses dual softmax layers to separately predict coarse waveform samples and then uses the coarse waveform to predict a fine waveform. It consists of one shared GRU layer and two separated feed-forward layers (one for coarse and one for fine). The global conditions are provided as an input to all layers in the decoder via concatenation.

As described in 2.4.2, the VQ codebooks function as lookup tables so while an utterance can be represented as a sequence of VQ codes (integers) as in Figure 2.4, each VQ code corresponds to a vector in the codebook. The sequence of vectors can be utilized as input to WaveRNN. There are two types of inputs to WaveRNN: local conditions and global conditions. *Local conditions* are a sequence of features provided as input. For example, the input could consist of upsampled mel filterbank features, or in the case of VQ-VAE, the input may be upsampled sequences of VQ phone vectors. *Global conditions* are optional in WaveRNN. When global conditions are specified (such as a one-hot vector or other embedding) then the vector is provided as input to every layer in every module of the decoder.

The output of WaveRNN is a waveform at 16-bit depth. WaveRNN is considerably faster to train than other neural vocoders such as the convolutional WaveNet [van den Oord et al., 2016] particularly because WaveRNN has fewer parameters. While other systems like WaveNet may achieve higher-quality speech output, WaveRNN is more suitable for experiments in this thesis due to constrained resources. In principle, the VQ-VAE system may be trained with any neural decoder that can accept local and global conditions. While the choice of decoder may have some small impact on speech re-synthesis quality, the work in this thesis is centered around the technique for achieving and assessing disentanglement rather than finding the optimal neural network architecture and the ideal hyper-parameters.

2.4.4 Speech Analysis-Synthesis

One way to think about the utility of VQ-VAE and the learned representations is through analysis-synthesis. Figure 2.5 shows a workflow for analysis-synthesis after a VQ-VAE system has been fully trained. In one case, the system could be used as "analysis" to obtain VQ embeddings from input speech. Those embeddings could then be used in some external application such as a separate speaker recognition system. In another case, embeddings could be generated or modified and then passed to the decoder to generate synthetic speech output. An example of this comes from Hayashi and Watanabe [2020] using VQ-VAE representations to perform TTS. A mapping of text to VQ codes was learned using neural machine translation and then the resulting sequence of VQ codes was passed to the decoder as local conditions to generate speech. They found that the compression ratio of the encoder had some impact on the performance of TTS. If the compression ratio is too low then it is more difficult to learn the mapping between text and VQ codes because the VQ code sequences are significantly longer than grapheme/phoneme sequences. If the compression ratio is too high then the re-synthesized speech quality suffers because the compressed representation is too lossy. A similar approach for TTS synthesis using self-supervised discrete codes was also proposed by Tu et al. [2020], however their work did not use VQ-VAE for learning discrete symbols that map to graphemes/phonemes.

2.5 Conclusion

This background chapter has summarized relevant work that forms the foundation of knowledge required for the experiments that will be presented in subsequent chapters. While this chapter is intended to provide the reader with important related work, each later chapter in this thesis will contain a smaller and more focused related work section. This chapter has shown the reader how speech representations for speaker recognition have evolved into a larger context of representations

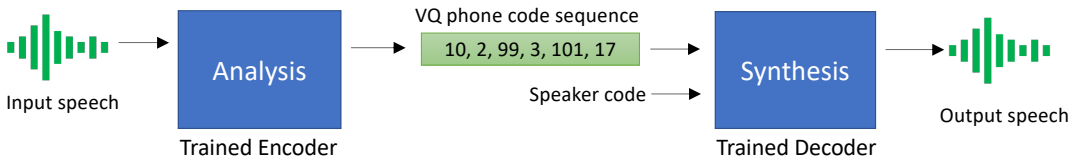


Figure 2.5: A workflow showing analysis-synthesis using trained VQ embeddings as local and/or global conditions for the decoder. The embeddings could be obtained by running analysis on the input, and then the embeddings could be used in an external application. Likewise, embeddings could be generated, modified or manipulated and then used for the decoder to generate speech output.

of other informational factors including emotion and style. Some representations can be learned separately from the intended use-case and then applied to a speech technology (such as TTS) to achieve better generalization or more expressivity of speech. A speech re-synthesis architecture was introduced and it will form the basis for experiments with disentanglement in Chapter 5. In addition, there was discussion of speaker spoofing in terms of the ability to detect replayed speech or synthetic speech in different types of speaker verification attacks. Next, Chapter 3 will further explore speech naturalness and authenticity using representation learning.

Chapter 3

Methods to Estimate Speech Authenticity and Quality

3.1 Introduction

Representation learning has many roles in speech processing, especially for building tools that automatically assess speech characteristics such as naturalness or authenticity. One of the reasons why representation learning is valuable for these tasks is because it is necessary to represent information in the speech signal abstractly at high levels, such as at the utterance level. While certain signal-level features have been shown to correlate with naturalness or speech authenticity, the signal-level features themselves can be used as input features into a deep neural network (DNN) which then learns abstract representations and makes decisions about speech naturalness.

As discussed in Chapter 2, there are two types of speech spoofing attacks: logical access (synthetic speech and voice conversion), and physical access (replayed speech). When an automatic speaker verification (ASV) system is presented with fake speech intending to impersonate a live human talker then this type of "spoofing" is commonly referred to as a *replay attack* or a *presentation attack* and the solution is sometimes referred to as *liveness detection* [Wu et al., 2015b; Todisco et al., 2016; Sahidullah et al., 2015; Wu et al., 2017; Todisco et al., 2019].

Physical access spoofing is particularly problematic for speaker verification because technically the speaker identity is correct, but it is an attack because a pre-recorded statement from a talker has been taken out of context. The aim of the experiments in this chapter are to detect whether or not a bonafide speech recording had been intercepted and subsequently replayed for speaker verification. There are numerous variables to be modeled including elements of how the attack was conducted as well as the type of ASV system being attacked. The approach presented in this chapter utilizes representation learning of the environment and attack type combined with signal-level features. The ability to automatically judge whether speech has been recorded and re-played out of context is an active research area gaining more attention lately due to predictive power of neural networks as well as advances in speech synthesis quality which can be used for various types of speaker identity attacks. An adversary might obtain a recorded snippet of authentic speech wherein the human target has used their voice to say a keyword or passphrase. It may be easy to record and replay speech to an ASV system. However,

automatically detecting replayed speech is a notoriously difficult task [Sahidullah et al., 2015].

A closely related problem is naturalness assessment which uses speech representations to predict or estimate the naturalness of synthetic speech in a way that aligns with human judgements. Automatic speech naturalness assessment can be used in a variety of speech tasks, from speech enhancement (SE) to voice conversion (VC) and text-to-speech (TTS) synthesis. Tools that can automatically assess synthetic speech would result in large gains in the field of speech processing such as faster and more efficient evaluation as well as savings from expensive listening tests. If there was an automatic speech naturalness prediction tool, it could be used for rapid assessment and comparison of models and it would facilitate design decisions such as the stopping criterion for TTS training. In addition, it may be possible to develop a tool that can someday be incorporated directly into the training process or used as part of a DNN loss function.

The purpose of this chapter is to present two related approaches to estimate speech authenticity and naturalness using representation learning. Section 3.2 will address the question: how should speech be represented in order to predict automatically whether the speech has been recorded once or re-recorded as a physical access spoofing attack when the speech is presented to an ASV system for speaker verification? An experimental technique will be proposed that combines signal-level features with learned representations. The performance of the system will also be described based on entry into the ASVspoof 2019 physical access (PA) challenge. Section 3.3 will address the question: how can synthetic speech be represented in order to automatically estimate human judgements of naturalness? The experiments will include utilizing the ASVspoof 2019 logical access (LA) challenge dataset consisting of a variety of speech synthesis and voice conversion systems with human naturalness ratings. A variety of different types of speech representations will be examined for how well they represent speech naturalness in a neural network prediction task. Finally, the most promising representation will be evaluated for generalization to a held-out text-to-speech system trained on different data (to simulate how a speech naturalness tool may be used in a real-world laboratory scenario).

3.2 Determining Speech Authenticity

This section describes joint work with Joanna Rownicka. Joanna's contribution was to create and analyze the environment and attack x-vector embeddings in Section 3.2.2.2 and Section 3.2.2.3. Joanna also was the primary author of those two sections though I helped edit the sections. Joanna created Figure 3.1 and Figure 3.2. The remaining portions of the work are my own including deciding which signal-level features to use, extracting the signal-level features, experimenting with combinations of the signal features and x-vector embeddings, implementing the CNN, training the CNN, and evaluating and submitting the system in the official competition. In the published paper at Interspeech 2019, all sections were co-written by both authors with myself taking the lead for approximately 85% of the writing and I was the corresponding author.

A fast-growing research area is to use automatic tools that determine speech authenticity [Wu et al., 2015b; Delgado et al., 2018; Todisco et al., 2019; Nautsch et al., 2021]. This is an important area of research that benefits society and consumers because speech is used more and

more in everyday devices. Voice biometrics are becoming more prominent as an alternative to fingerprint or face recognition because of the ease of use since voice is a natural human interface. While voice biometric systems become more prominent, one issue that continually arises is whether or not this type of biometric is secure enough for high-stakes access such as financial transactions. One of the security concerns involves the ability to spoof or fake speech in order to trick a biometric system such as automatic speaker verification (ASV). To mitigate this risk, researchers have been working on countermeasures that detect spoofing attacks. During recent years, there have been several international challenges where teams can compete for the best system configuration to produce the best countermeasure. While there are many different kinds of speech spoofing attack, including voice conversion and text-to-speech [Janicki et al., 2016; Shiota et al., 2016; Blue et al., 2018], this chapter focuses on replay attacks.

Recent work suggests that replayed speech contains artifacts that provide clues indicating that the speech has been recorded once, then replayed and recorded again. These clues may be found in either the time or frequency domains [Lai et al., 2019]. Other work has explored energy-based features [Kamble et al., 2018; Kamble and Patil, 2019], attention-based adaptive filters [Liu et al., 2019], and convolutional neural networks (CNNs) [Chettri et al., 2018a; Himawan et al., 2019]. It is particularly challenging to model different types of acoustic environments, playback devices, and recording devices [Chettri et al., 2018b]. Recent work has also shown that high-frequency sub-bands in the acoustic signal contain evidence of replay. For example, inverted mel frequency cepstral Coefficients (IMFCCs) consistently discriminate speech replay across several frequency sub-bands [Witkowski et al., 2017]. IMFCCs are derived by inverting the mel-scale filterbank in the frequency domain at approximately $f = 2$kHz to capture more detail from higher frequencies that would be missed by MFCCs [Chakroborty and Saha, 2009]. Other recent work has shown that Sub-band spectral centroid magnitude coefficients (SCMCs) are the best and most consistent signal feature [Font and Cano, 2017], while the constant-Q cepstral coefficient (CQCC) features are also promising [Nagarsheth et al., 2017]. SCMCs come from the spectral centroid magnitudes of the filter bank energy after a Fourier transform has been performed (i.e., in the frequency domain). After spectral centroid magnitudes are extracted from filter banks in an audio frame, a centroid is calculated for each sub-band in that frame. The logarithm of the centroids is taken and a discrete cosine transform is performed, resulting in SCMC features. The SCMCs are similar to MFCCs except that the SCMCs represent energy in a frequency sub-band more finely than MFCCs [Paseddula and Gangashetty, 2018].

The experiments that will be described in this section are related to the development of a spoofing countermeasure system that was submitted to the ASVspoof 2019 physical access challenge [Williams and Rownicka, 2019]. First, the specific signal-level features that were used for anti-spoofing experiments will be presented in 3.2.2.1. Second, a novel type of x-vector embedding will be introduced in 3.2.2.2 and these embeddings capture the recording conditions of an utterance using the labels that were provided by the PA dataset (attack and environment variables). Third, the x-vector embeddings will be analyzed in 3.2.2.3 to observe how they model factors of variation from different recording conditions. Finally, a trained countermeasure system will be presented in 3.2.3 to demonstrate that the combination of signal features and x-vector embeddings out-performs all baselines for both metrics on the ASVspoof 2019 development and evaluation datasets.

3.2.1 Dataset Description

The data used for experiments in this section came from the ASVspoof 2019 Physical Access (PA) challenge and is described in detail in [Wang et al., 2020b]. The PA track of the challenge addresses the development of countermeasures for replay spoofing attacks. This dataset was previously described in Section 2.2.3.1 and the reader is encouraged to review it before proceeding further to the experiments and system descriptions of this chapter.

3.2.2 Feature Development

3.2.2.1 Speech Signal Features

Following from the features analysis for replay attack detection that was presented in [Font and Cano, 2017] for the ASVspoof 2017 challenge, the following signal-level features were extracted: mel frequency cepstral coefficients (MFCCs), inverted mel frequency cepstral coefficients (IMFCCs), rectangular filter cepstral coefficients (RFCCs), linear frequency cepstral coefficients (LFCCs), sub-band spectral centroid magnitude coefficients (SCMCs), and constant-Q cepstral coefficients (CQCCs) [Todisco et al., 2017]. A description of the features is provided in Table 3.1. Static features were used because preliminary experiments indicated that the dynamic features were not as useful in the spoofing task, especially the second-order dynamic features.

Features	Coeff. num. (N)	$f_{min} - f_{max}$ (Hz)
MFCC	70	300-8000
IMFCC	60	200-8000
RFCC	30	200-8000
LFCC	70	100-7800
SCMC	40	100-8000
CQCC	50	15.62-8000

Table 3.1: Description of signal features extracted from the raw speech audio. Note CQCC used a specific f_{min} from Todisco et al. [2017]

The IDIAP Python *Bob.ap* signal processing library was used to extract this set of signal processing features from speech audio files [Anjos et al., 2012, 2017]. In the case of CQCCs, the code to extract this feature was provided by the challenge organizers and is further described in [Todisco et al., 2017]. For each audio file, the feature extractor output was an NxM-*dim* matrix with N as the number of coefficients and M as the duration of the file in frames.

Each audio file in the ASVspoof 2019 dataset was of a different length. To account for the variable length audio utterances, a further pre-processing step was performed which resulted in a set of same-sized feature vectors that could be used as input to a classifier later on. This pre-processing step consisted of a down-sampling technique in the feature space. This means that for a given coefficient in a given audio file, the number of frames was down-sampled to a constant value. The technique preserved the original per-coefficient distributions in a file while also setting the number of frames to a constant and also preserved more information than averaging across frames. To perform this re-sampling, the Fast Fourier Transform (FFT) was applied in such as way that the spacing between the original frames, $s = \delta x$, then became: $s = \delta x * M/M'$. This allowed to set a constant for the number of down-sampled frames M'

to $M' = 10$. Effectively this procedure shortened the audio file duration to 10 frames while preserving the mean and standard deviation of each coefficient[1].

After that pre-processing step, the signal processing features were therefore represented as a $Nx10$ dimensional matrix. This paragraph describes how that matrix was created. First, the number of frames selected was $M' = 10$ based on two motivations: 1) to reduce the overall size and overhead of the dataset for later processing, and 2) to allow the countermeasures to operate on very short audio examples. The coefficients were then stacked on a per-frame basis, creating a matrix, and this was done in an effort to preserve some temporal nature of the original signal. Finally, each of the signal feature values were re-scaled to be between -1 and $+1$ using per-feature max values from the training set, applied later to development and evaluation sets. The values were re-scaled in order to align with the selection of activation function for the regression task described in Section 3.2.3, which outputs a value between -1 (spoofed) and $+1$ (authentic) which was required for the ASVspoof 2019 Challenge submission.

3.2.2.2 *x*-vector Embedding Creation

In addition to the signal features, x-vectors [Snyder et al., 2018] were also used as auxiliary features for the CNN model described in Section 3.2.3. The goal of using x-vectors was to extract meaningful fixed-size utterance-level vectors representing the factors of variation, namely environment and attack conditions, in the spoofing task. The extracted representations were used to account for these factors in the final spoofing detection task. This effort was to improve the system robustness to unseen conditions by leveraging information about the environment and attack classes from the labels provided for each training example.

The Kaldi Toolkit [Povey et al., 2011] was used to extract x-vectors representing a joint environment+attack class (which is now referred to as *env+attack*). The input features for the x-vector extractor were 40-*dim* MFCCs, with 80 filters in a filter bank. The x-vector extractor was a time-delay neural network (TDNN) with the same architecture as in [Snyder et al., 2018] (i.e., 5 convolutional layers, 1 statistics pooling layer, 2 feed-forward layers for embeddings **a** and **b** and a final softmax layer). Batch normalisation was used as well as ReLU activations for the convolutional and feed-forward layers. This is the same TDNN architecture that was introduced in Chapter 2.2.2.4. The x-vectors were extracted from the seventh layer of the network (embedding **a**). Differently to the model in [Snyder et al., 2018] though, the one developed here was not trained to classify speakers. The extractor was trained to differentiate between classes jointly representing types of acoustic environments and types of attacks.

The joint *env+attack* classes were created by combining each category label of variation for the simulated acoustic environments and attack types (e.g., room size, T60 reverberation time, talker-to-ASV distance, attacker-to-talker distance, and replay device quality). From 10 attack type configurations, and 27 acoustic environment configurations (9 attacks plus authentic speech), 270 env+attack classes were created. Training an x-vector extractor to differentiate between *env+attack* classes provided the ability to learn fixed-size representations, capturing both the type of attack and the type of acoustic environment. The classification accuracy from the TDNN for the joint *env+attack* x-vectors was around 85% for 270 unique classes on a held-out validation set (10% of training), hence these representations were meaningful.

After the x-vector extraction, the dimensionality of the x-vectors was reduced from 512-*dim*

[1] `https://scipy.org/`

to 10-*dim* using Linear Discriminant Analysis (LDA). The LDA model also used *env+attack* classes for training. The idea for reducing dimensionality to 10-*dim* was based on assessing the equal error rate (EER) of the x-vectors on the development set. Multiple values of dimensionality reduction were tried (such as 400, 300, 200, 100, 50, and 10) and the EER was examined for each dimension size. The smallest dimension was selected (10-*dim*) because it gave the best EER. Finally the performance of the learned x-vectors with dimensionality reduction was examined on an evaluation set. For 59,400 evaluation trials with non-target proportion of 50%, the EER in the *env+attack* verification task was 23.96% with the LDA dimensionality reduction.

The LDA-reduced x-vectors (10-*dim*) were scaled with $c = 0.1$ constant. Empirically the scaling was found to have a good effect on the final system performance. Scaling x-vectors before concatenation is conceptually similar to applying a fixed LDA-like transform in Kaldi (usually used when i-vectors are concatenated to the input features for normalisation in ASR), which is scaling down the dimensions that are "non-informative". This has the effect of encouraging stochastic gradient descent (SGD) to ignore non-informative values. Altogether, six types of x-vectors were used for experimentation in Section 3.2.3: x-vectors that model only the attack classes (xA), x-vectors that model only the environment classes (xE), x-vectors that jointly model the environment and attack classes (xEA), and a version of each of these were also scaled (xAs, xEs, xEAs). The next section will describe some of the analysis of the x-vectors.

3.2.2.3 *x*-vector Embedding Analysis

In this section an analysis is provided to demonstrate how the x-vector embeddings could differentiate between different types of attacks and different types of environments. There were 27 environment classes and 10 attack classes. It is easier to show the analysis for environments and attacks separately, compared to modeling all 270 classes for the jointly trained *env+attack* embeddings (xEAs), which are the ones ultimately used in the challenge system submission.

Environment classification was an easier task (accuracy 86% on the validation set) than attack type classification (accuracy 60% on the validation set), even though the number of classes for classifying attacks was smaller than for classifying environment. It is possible that this may be caused partly by data imbalance in the case of attack recognition, compared to the evenly distributed examples for the environment classes.

To further investigate what the extracted x-vectors were capturing, the accuracy scores were analyzed from predicting attack and environment. Figure 3.1 shows a matrix of the classification accuracy scores for mean x-vectors per attack. First of all, it can be observed that the replay device quality is well captured by the x-vectors. Attack classes using poor replay device quality (indicated by the letter 'C' as the second letter in AC, BC, and CC) had a high classification accuracy. However, the attack-to-talker distance does not seem to be modeled well with the attack x-vectors. For example, scores for classes AC, BC, and CC are very close to each other, but different than for any other classes. The most evenly distributed scores can be observed for the medium-quality replay device classes (AB, BB, CB). In summary, the attack x-vector embeddings can detect when the replay device is very poor. However, if the replay device quality is near perfect, it is much more difficult to develop the spoofing countermeasures with the proposed x-vector embeddings.

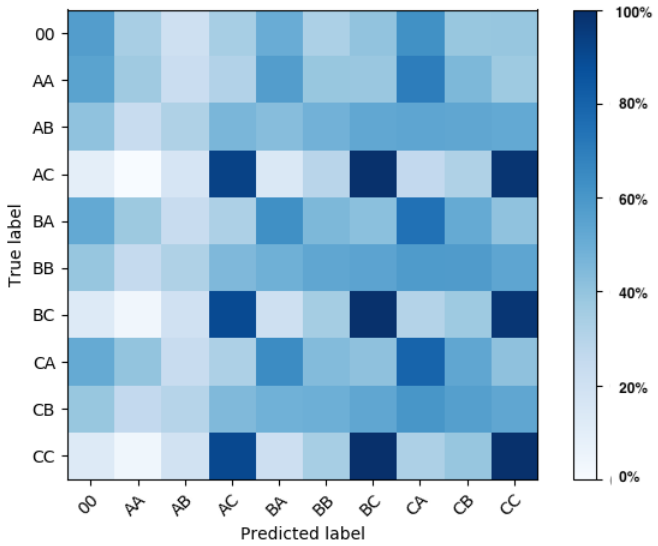


Figure 3.1: Matrix showing the classification accuracy for each attack class using the scaled attack+environment embeddings (xEAs). The scoring is based on the per-attack mean development x-vectors against per-attack mean training x-vectors. Class *00* is the bonafide class. For other labels, letters in the first position corresponds to the attack-to-talker distance (A - lowest, C - highest), letters in the second position corresponds to the replay device quality (A - the best, C - the worst).

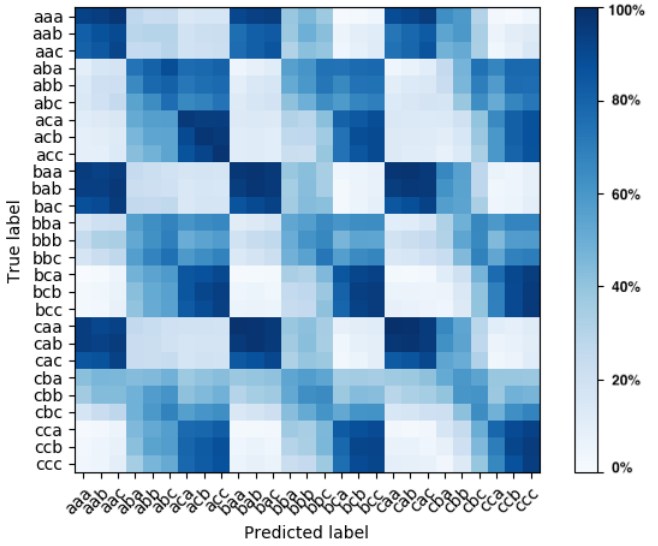


Figure 3.2: Matrix showing the classification accuracy for each environment using the scaled attack+environment embeddings (xEAs). The scoring is based on the per-environment mean development x-vectors against per-environment mean training x-vectors. Letters in the first position of the label ID correspond to the room size (a - smallest, c - biggest). Letters in the second position correspond to T60 reverberation time (a - shortest, c - longest). Letters in the third position correspond to the talker-to-ASV distance (a - shortest, c - longest).

Figure 3.2 shows a matrix of the classification accuracy where the scores represent how well the mean x-vectors discriminate environment classes. Recall that the environment variables are represented as a triple of discrete categories using letters: a, b, and c. Again, the talker-to-ASV distance and the room size do not seem to be very well captured (first and third letter position in labels). However, the reverberation time (especially short and long) discriminates classes well.

Both spoofed and bonafide recordings were simulated in a variety of environments. So the idea was to extract an embedding that would help to normalize out the effects of recording in different environmental conditions, to be able to generalize well to unseen conditions at test time. Even though the x-vectors do not differentiate very well between every attack and every environment class, they do differentiate between some of them. Furthermore, x-vector embeddings for similar acoustic conditions are close to each other in the x-vector space. Therefore, these can be useful representations complementing the other signal processing features that are used for the experiments in this chapter.

3.2.3 Experiment Design

The approach taken in this chapter for designing a spoofing countermeasure was to treat it as a regression problem using a convolutional neural network (CNN). Categorical target labels were converted to numerical values as follows: "spoof" became -1 and "bonafide" became $+1$. The activation function was set to the hyperbolic tangent function so the output of the system was a value between -1 and $+1$. The challenge evaluation plan called for negative values to correspond to the "spoof" class and positive values to correspond to the "bonafide" class. The system output values could therefore be interpreted to represent the degree of authenticity of the audio. Our system was later evaluated in tandem with an ASV system known only to the challenge organizers.

3.2.3.1 Feature Combination

Several combinations of features were explored while evaluating on the development set. For the first case, each signal processing feature was evaluated individually as the input feature for the CNN model. Next, each type of x-vector embedding was evaluated individually. Finally, the signal processing features were combined with x-vector embeddings. They were concatenated to the signal processing features at the input of the CNN model. This technique was intended to normalize out some of the factors of variation, and to enable the system to learn a spoofing countermeasure robustly.

The final system submission to the ASVspoof 2019 Challenge was based on two features: SCMC signal features concatenated with the $xEAs$ vectors (scaled and transformed as described above). For the submission, the dataset consisted of 54,000 training instances and 29,700 development instances. The training data was therefore of size $(54000, 410)$. These dimensions were based on 40 SCMC coefficients by 10 frames per coefficient, plus the additional 10-dim x-vectors. The two features were combined via concatenation. In each of the training and development sets, there were 5,400 instances labeled as bonafide while the remaining had been labeled as spoof, thus the dataset was extremely imbalanced for the two targets. This imbalance was manufactured by the challenge organizers.

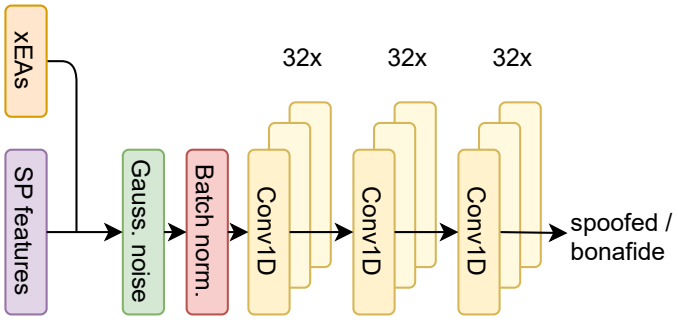


Figure 3.3: Overview of the CNN system architecture submitted to the ASVspoof 2019 physical access challenge. The system submission was based on scaled LDA x-vectors (xEAs) combined with SCMC signal features. The system consisted of a Gaussian noise layer, and 3 1D convolutional layers with max pooling. The final layers were one fully-connected layer followed by another layer with a single output using the hyperbolic tangent activation function (to obtain values between -1 and 1).

3.2.3.2 Convolutional Neural Network (CNN)

The system[2] that was developed for the challenge was implemented with the Keras library [3] with TensorFlow backend [Abadi et al., 2016]. It was designed to perform regression and output a continuous value to adhere to the expectations of the challenge. Figure 3.3 shows an overview of the system architecture. The first layer of the CNN was an additive Gaussian noise layer [Bishop, 1995; An, 1996; Dutta et al., 2018]. This noise layer was used to help the model generalize to unseen conditions, especially considering the strong imbalance of bonafide and spoofed targets. The placement of the Gaussian noise layer was determined experimentally and different values for the standard deviation of the noise distribution were also tried, finally deciding on $n_{std} = 0.001$. The next layer was a batch normalization layer. The CNN consisted of 3 conv1D layers. The kernel size was set to 3 with 32 convolutional filters. Each conv1D layer included an L2 regularizer [Ng, 2004]. Each conv1D layer was followed by a max pooling layer with pool size and stride set to 2. Finally, a fully connected layer followed with a single output using the hyperbolic tangent activation function ($tanh$) [LeCun et al., 2015]. The activation function had the effect of restricting the output between -1 and 1.

3.2.3.3 System Training

The CNN model was trained using the official challenge training set with 10% held out of the training set as a validation set. During training several different parameters were swept. First, the values for standard deviation in the Gaussian additive noise layer were explored in the range of: [0.000001, 0.00001, 0.00005, 0.0001, 0.001]. Experiments that omit the Gaussian noise layer altogether are indicated by -N for results reporting, meaning without noise. For L2 regularization the values were explored between [0.00001, 0.001]. For training, the loss was measured using mean-squared-error (MSE). Early-stopping [Prechelt, 1998] was used and the validation loss was monitored with $delta = 0$ and patience $p = 5$ epochs. The optimizer was Adam [Kingma and Ba, 2014] with learning rate $lr = 0.001$ and all other remaining parameters were default from the library.

[2] `https://github.com/rhoposit/ASVchallenge2019.git`
[3] `https://keras.io`

3.2.4 Results

The system was evaluated using two related metrics: equal-error rate and tandem detection cost function (described earlier in Chapter 2.2.3.3). The primary ASVspoof 2019 challenge metric was the tandem detection cost function (tDCF) computed in conjunction with an ASV system that was kept hidden from participants [Kinnunen et al., 2018] but provided by the challenge organizers. This allowed the organizers to vary the ASV system to evaluate robustness of all challenge submissions. The secondary metric was equal-error rate (EER) based on the quality of the countermeasure alone to predict bonafide versus spoofed. Our challenge system submission was selected using the best feature combination (SCMC+xEAs) based on performance with the tDCF metric on the development set.

Table 3.2 shows the system performance on the official challenge development set using the signal features, the x-vector features, and the best-preforming feature combination. Also reported are the official results for the challenge submission on the evaluation set. For reference, the evaluation performance for both baselines is also reported; it used a Gaussian Mixture Model (GMM) and two signal features: LFCC-GMM [Sahidullah et al., 2015] and CQCC-GMM [Todisco et al., 2017]. Our system submission performed better than both baselines for both metrics on development and evaluation sets (to 3-significant digits).

		Development		Evaluation	
		t-DCF	EER (%)	t-DCF	EER (%)
Signal Features	**MFCC**	**0.204**	**8.35**	-	-
	IMFCC	**0.199**	**7.98**	-	-
	RFCC	0.210	8.58	-	-
	SCMC	**0.209**	**8.47**	-	-
	LFCC	0.229	8.90	-	-
	CQCC	0.275	10.9	-	-
x-vectors	**xA**	**0.814**	**31.5**	-	-
	xE	0.971	41.5	-	-
	xEA	**0.820**	**31.6**	-	-
	xAs	**0.815**	**31.4**	-	-
	xEs	0.970	41.7	-	-
	xEAs	**0.820**	**31.9**	-	-
Combo	**SCMC+xEAs**	**0.194**	**7.74**	**0.235**	**9.15**
	SCMC+xEAs-N	0.225	9.16	-	-
	IMFCC+xEs	0.197	7.47	-	-
	MFCC+IMFCC	0.206	7.96	-	-
	LFCC-GMM	0.255	11.9	0.301	13.5
	CQCC-GMM	0.195	9.87	0.245	11.0

Table 3.2: Results for the architecture using different features. It was evaluated on the official development set and the official evaluation set (for the challenge submission). Two baselines from the organizers are included last, for reference.

3.2.5 Discussion

While the x-vectors alone did not distinguish well between spoofed and bonafide speech, there were performance improvements when the x-vectors were combined with signal features. Specifically, the best feature combination was found to be SCMC features with the scaled LDA x-vectors ($xEAs$) that jointly modeled environment and attack variations. The SCMC features capture the magnitude of energy in sub-bands, which can effectively distinguish two signals even if they share the same average energy. The SCMC feature was also one of the best-performing features from the analysis in [Font and Cano, 2017] for the ASVspoof 2017 challenge, though that challenge was based on different data. It has been recognized as a stable feature across experimental conditions. Experiments with the Gaussian noise layer indicated that using the noise layer always performed better than without it (reported as **SCMC+xEAs-N** in Table 3.2).

In earlier analysis (Section 3.2.2.3), it was found that differences in the replay device quality were captured well by the x-vectors if the device quality is poor. However, when the replay device quality is very good, or perfect, then it is much more difficult to detect spoofing. The official detailed performance results from the ASVspoof 2019 challenge organizers also indicate that certain attack types are much more difficult [Todisco et al., 2019], specifically with the high-quality replay device. That is an important finding for ASV research and is also supported by the experiments for our challenge submission described in this thesis.

In summary, a new type of x-vector has been introduced that models environment and attack characteristics as categorical values from the 2019 ASVspoof PA Challenge dataset. In the next section, experiments will be presented that build upon the idea of using representation learning to detect artifacts in speech for the purpose of evaluating synthetic speech quality. The special x-vectors will again be used but instead of detecting speech authenticity they will be used to predict the quality of speech produced by a text-to-speech (TTS) synthesis system.

3.3 Assessing Naturalness of TTS Output

This section describes joint work with Pilar Oplustil-Gallegos and Joanna Rownicka. Joanna's contribution was to create and analyze the CNN acoustic embeddings from the AMI corpus in Section 3.3.3.2 as well as create and analyze the x-vector embeddings presented in Section 3.3.3.3. Pilar's contribution was to provide the trained text-to-speech (TTS) synthesis system that was used as a held-out system for analysis in Section 3.3.7.1. The remaining portions of the work were my own. All sections were co-written and proofread between all authors, with myself taking the lead for 90% of the writing. Joanna contributed significantly to the writing of Section 3.3.3.2 and Pilar contributed significantly to the writing of Section 3.3.7.1.

It is widely accepted that data from certain speakers results in higher quality synthetic speech when building a text-to-speech (TTS) system and this is true across different types of TTS systems, from HMM-based [Cooper et al., 2016; Hinterleitner et al., 2014] and DNN-based statistical parametric speech synthesis (SPSS) [Cooper and Hirschberg, 2018], to encoder-decoder models [Oplustil-Gallegos et al., 2020; Lorincz et al., 2021]. It is becoming more commonplace to use large multi-speaker corpora for TTS training [Gibiansky et al., 2017]. Often these large

corpora of natural speech are of variable recording quality, such as LibriTTS [Zen et al., 2019]. While it was once preferable to train TTS systems on speech from studio-quality professional voice actors, there has been an increasing effort to incorporate more data that is 'found' on the internet and that consists of speakers 'in the wild'. This found data is often multi-speaker and could be very useful for multi-speaker TTS and deep learning if the data is well-managed. Multi-speaker TTS systems, such as [Gibiansky et al., 2017], learn simultaneously a common sound-to-phoneme mapping as well as the nuances of each speaker's individual voice.

Tools that can predict the quality or naturalness of synthetic speech could be very useful for research and development purposes. Such tools could speed up the development cycle or save costs from doing many listening tests. In fact, it can be difficult to compare across speech synthesis systems due to inconsistencies in how listening tests are conducted over the years. While authors report system performance at the time of publication, there is no guarantee that a listening test protocol is consistent from one year to the next or from one laboratory to another [Cooper and Yamagishi, 2021]. A standardized tool could help with making comparisons and allow researchers to better characterize gains in system performance.

One recent advancement in automatic speech naturalness assessment was MOSNet [Lo et al., 2019], which extended similar work from Quality-Net [Fu et al., 2018]. The MOSNet system was trained to predict mean opinion scores (MOS) as a measure of naturalness for voice conversion (VC), based on human judgments of MOS as the ground truth. The MOSNet system is free and open-source, and provides three different pre-trained models that were trained on MOS scores for voice conversion data. Although VC and TTS tasks are somewhat related, preliminary experiments in this chapter showed that the pre-trained VC models did not generalize well for synthetic speech from TTS.

Automatic naturalness assessment is inherently very difficult. It requires a large dataset that has multiple speakers, and multiple systems, and must be consistently labeled with naturalness scores. As a machine learning task, previous work has shown that regression models will tend to learn an average score and that it is more difficult for the model to predict high and low outliers [Lo et al., 2019]. A related problem comes from human judgments of naturalness. If human listeners are not exposed to a wide variety of poor and high naturalness speech during the listening test, there is a risk that many utterances will be marked "average" [Clark et al., 2005]. It is also widely accepted that MOS scores do not reflect the complex nature of a naturalness spectrum since the relationship between values and quality is not necessarily linear [Mayo et al., 2005] (i.e., the difference between MOS scores of 1 and 2 $\neq$ difference between 3 and 4).

There is no guarantee that a speech naturalness model trained for one TTS system would generalize to another TTS system. The approach described in this chapter is motivated by the desire to build a MOS estimation tool that could generalize to new systems. With that in mind, 13 different systems will be used to develop a new MOS naturalness estimation model (7 TTS, 3 TTS+VC, and 3 VC) that range in quality but that use the same speakers. The experiments that will be discussed in this chapter explore a variety of different types of speech representations (Section 3.3.3) including those presented in Section 3.2 for detecting speech authenticity. The speech representations will be used to make comparisons across multiple TTS and VC systems as well as multiple speakers. After identifying the best-performing speech representation and training a model, the model will be applied to a held-out multi-speaker TTS system trained on a different dataset (Section 3.3.7.1). The approach to speech naturalness estimation in this chapter has been inspired by previous work by Lo et al. [2019]. In this chapter, the following

contributions will be made:

1. Re-train the original MOSNet framework using a multi-system dataset (13 different TTS and VC systems) and explore frame-based weighting in the loss function
2. Train a new low-capacity CNN architecture on a multi-system dataset and compare 8 different utterance-level speech representation
3. Characterize model prediction performance based on speaker-level rankings
4. Assess how well the trained model generalizes to speech from an unseen TTS system

A multi-system database will be used for training with the expectation that training on multiple TTS and VC systems will result in better model generalization for unseen TTS systems. Another aim of the experiments is to try to identify speakers whose data results in the highest and lowest quality synthetic speech. A ranking-based evaluation will be used to assess the models and representations at the system-level as well as the speaker-level. The final output will be a publicly available tool based on the best-performing model configuration and representation that predicts MOS scores with the highest correlation to human judgments.

3.3.1 Related Work

Speech quality estimation is an important area of research, especially for synthetic speech, and has garnered an incredible amount of attention recently [Hinterleitner et al., 2013a, 2014; Hinterleitner, 2017; Leng et al., 2021; Huang et al., 2021; Cooper et al., 2021]. While automatic methods are well-motivated, this problem is inherently difficult due to variability in ground truth human judgments. MOS ratings are one of many de-contextualized TTS metrics. Wagner et al. [2019] calls for developing a set of best practices for TTS evaluation. Those best practices could include a critical analysis the effectiveness of automatic tools, such as AutoMOS [Patton et al., 2016], QualityNet [Fu et al., 2018], and MOSNet [Lo et al., 2019].

While there have been many previous works regarding the perception of synthetic speech [Hinterleitner et al., 2014, 2013a, 2011, 2013b], at the time that the experiments in this chapter were conceived there were no studies that explored MOS prediction using externally-trained speech representations. The result is a gap in understanding exactly which kinds of abstract speech representations are most useful for MOS naturalness prediction for TTS synthesis and the experiments in this chapter attempt to address that gap.

In the AutoMOS framework proposed by Patton et al. [2016], the speech representations that were experimented with were log-mel spectrograms and single-layer time-pooled convolution on the waveform (sampled at 16kHz). They trained a neural network with those two inputs and tried to predict MOS naturalness ratings using human judgements as ground truth targets. The TTS data came from a Google TTS engine that was compiled over multiple years for US English. Annotators rated the speech on a Likert scale of 1-5 (5 is natural). While AutoMOS experiments showed some correlation between predicted and ground truth MOS scores, the authors note that very low and very high scores were difficult to model. In QualityNet [Fu et al., 2018], the input was magnitude spectrogram and the prediction targets were PESQ scores [Rix et al., 2001]. One advantage of QualityNet is that it predicted a score frame-by-frame and that provided some interpretability of the model. Final utterance-level scores were based on

aggregated frame-level predictions. The PESQ scores are not intended for synthetic speech but rather for assessing quality of speech enhancement, codecs and telephony, so it is not directly applicable to the work of TTS naturalness estimation in this chapter. The MOSNet framework from Lo et al. [2019] is the previous work most relevant to the experiments that will be presented in this chapter (Section 3.3.4). MOSNet uses frame-level features from a magnitude spectrogram and predicts MOS naturalness at each frame, then aggregates the frame-level scores to create a final utterance-level score. Since the MOSNet system will be used for comparison experiments in this chapter, the architecture will be described in more details in Section 3.3.4.1.

CNN-based speech embeddings were proposed in [Rownicka et al., 2018] for the purpose of acoustic model adaptation. That work showed that a deep CNN model trained on filter-bank features could learn information about speaker, gender, and channel noise. While they demonstrated CNN representations worked well for the original case of acoustic modeling, they also show that these embeddings could be applied to differentiate between acoustic conditions. Similar acoustic representations will be used in this chapter for learning to estimate speech naturalness in Section 3.3.3.

In terms of image-based representations, the Deep Spectrum (DS) features are also generated from a deep CNN [Cummins et al., 2017]. For DS features, a pre-trained model (VGG16, VGG19, AlexNet, SqueezeNet, or GoogleNet) is used. The pre-trained models are based on image data (but not speech or spectrograms). To use the pre-trained model and extract DS features, a spectrogram is provided as input and an inference step is performed. Then the activations for a specific layer in the network (usually a fully-connected layer such as *fc2*) are output and saved. These activations from the network are the DS features and the dimensionality of the feature vector depends on the size of the layer, which is set to 4096-*dim* in this chapter. The DS features were shown to be useful for speech emotion recognition as well as the detection and analysis of snoring noises during sleep [Cummins et al., 2017; Amiriparian et al., 2017; Zhao et al., 2019]. DS features will be used in this chapter for speech naturalness estimation alongside other representations in Section 3.3.3.

3.3.2 Dataset Description

Two datasets will be used for system training in the experiments for this chapter. The ASVspoof 2019 Logical Access dataset will be used for training a MOS prediction network in Section 3.3.4. The LibriTTS Dataset will be used to train a held-out TTS system, and that system will be described in more details later in Section 3.3.7.1.

3.3.2.1 ASVspoof 2019 Logical Access (LA) Dataset

The ASVspoof 2019 LA dataset[4] was introduced in Chapter 2 and is also described in detail in Wang et al. [2020b]. When the dataset was created by the ASVspoof 2019 challenge organizers, the original intended use was to develop systems that learn to discriminate between genuine and spoofed speech samples. The work in this chapter uses a version of the LA dataset that had also been annotated with MOS scores by Wang et al. [2020b]. The MOS scores reflect human judgments of naturalness for synthetic speech. The reader is encouraged to consult the description of TTS and VC systems referenced in Table 2.3 that was presented in Chapter 2.

[4] `https://datashare.is.ed.ac.uk/handle/10283/3336`

The evaluation subset of the LA dataset was annotated with MOS naturalness scores and provided freely by Wang et al. [2020b]. Each utterance was labeled with a MOS naturalness score on a scale of 1 (definitely machine) to 10 (definitely human). The MOS scores were obtained through a human listening test conducted by Wang et al. [2020b]. Listeners were instructed with: "imagine you are working at a call center and must determine if speech from an incoming call is human or machine". Each utterance was rated with one MOS score, as the effort was to balance ratings between human and spoofed examples rather than achieving high inter-annotator agreement. For the experiments that will be described in this chapter, the annotated dataset was divided into training, validation, and test splits using similar proportions as the previous work of Lo et al. [2019]. Each split contained the same TTS/VC systems, but had different speakers. The test set had a total of 9 speakers with approximately 320 utterances per speaker evenly balanced across the 13 TTS/VC systems.

3.3.2.2 LibriTTS Dataset

The LibriTTS dataset is a large speech dataset of 2,456 speakers [Zen et al., 2019]. LibriTTS was designed and released by Google, aiming to be used for TTS applications, removing noisy data and normalizing text, among other improvements. It contains a wide range of speakers reading aloud books. As each speaker was able to recorded themselves anywhere and with any equipment and that resulted in diversity of recording quality, channel effects and room noise throughout the data. Despite Google's cleaning process, the level of quality throughout the "clean" version of the corpus varies but it is still used for TTS synthesis [Kim et al., 2020]. The LibriTTS data will be used in this chapter for training a held-out TTS system in Section 3.3.7.1 to observe how well the proposed MOS estimation tool can generalize to an unseen TTS system trained on an unseen dataset. Since all of the TTS/VC systems in the LA dataset were trained on multi-speaker data from VCTK, it will be helpful to train a held-out TTS system on LibriTTS in order to simulate how a MOS estimation tool would generalize in a real-world use case.

3.3.3 Speech Representations

The experiments in this chapter will utilize and compare speech representations alongside the frame-based features that were used to re-train the original MOSNet. The speech representations that will be used for experimentation are: Deep Spectrum features, x-vector embeddings, and acoustic model embeddings.

3.3.3.1 Deep Spectrum (DS) Features

Deep Spectrum (DS) features[5] will be used for the speech synthesis naturalness estimation experiments in Section 3.3.4. The motivation for exploring DS features is that they are a general-purpose image-based representation and they have been shown to be useful for detecting sleep apneas and snoring [Amiriparian et al., 2017], essentially modeling very fine-grained types of vibrations at a high resolution. The representations are based on modeling image features at a very high level of abstraction. Since they are image-based they will operate on the speech spectrogram rather than the waveform. DS features are created by a forward pass on a very deep pre-trained CNN model using an utterance-level magnitude spectrograms as

[5] `https://github.com/DeepSpectrum/DeepSpectrum`

input. The particular CNN that was used in this section was a pre-trained VGG19 network [Simonyan and Zisserman, 2015] that was originally trained for general image-recognition tasks. The activations from a particular layer of the deep CNN model for a given input are what constitute a DS representation. The VGG19 model and output layer *fc2* was used for all DS representations in this chapter. This particular layer was the default setting, and represents the second fully-connected layer (near the output layer) which comes after the deep convolutional layers. DS speech representations are 4096-*dim*.

3.3.3.2 Acoustic Model (AM) Embeddings

Another general-purpose embedding will be used for experiments in Section 3.3.4, however unlike the DS features, this general-purpose embedding is directly related to speech. An acoustic model (AM) is a model of speech that learns a mapping between acoustic features in the speech signal and phones, or more specifically sub-phone units such as senones [Hwang and Huang, 1992]. AM models are prominently used in automatic speech recognition (ASR). Recently Rownicka et al. [2018] and Rownicka et al. [2019] proposed to add utterance-level embeddings to neural network acoustic models, including DNNs and CNNs, for the purpose of learning abstract speaker-invariant representations to help AM models generalize better to unseen speakers for improved ASR. After training an AM, the embeddings can later be extracted from five selected layers $i = \{3, 6, 9, 12, 15\}$. For the experiments in this chapter, embeddings from a pre-trained deep CNN acoustic model from Rownicka et al. [2019] were used. The CNN acoustic model had been trained on MFCCs as input with cross-entropy loss for triphone states to perform speech recognition on the AMI multiparty meetings corpus [Renals et al., 2007]. As a first step for extracting AM embeddings, the pre-trained model parameters were fixed and a forward pass was performed for each utterance. To create utterance-level AM embeddings for use in the experiments of Section 3.3.4, the representations at different layers i were concatenated and reduced to 512-*dim* with a PCA transform. In this chapter, each utterance-level AM embedding will be 512-*dim*. The embeddings are interesting for speech naturalness estimation because they are extracted from a deep CNN model that is trained for speech (rather than images like the DS features). The AM embeddings could serve as potentially good generic utterance summaries, without imposing any specific characteristics or constraints.

3.3.3.3 *x*-vector Embeddings

So far, two generic types of embeddings have been proposed: DS features (image-based) and AM embedding (speech-based). In this section a less generic embedding is proposed that tries to model specific features of speech using categorical labels. In previous experiments from Section 3.2.3, new x-vector embeddings showed that it is possible to model a variety of variables from the ASVspoof 2019 PA dataset. In the previous experiments of Section 3.2.3, environment variables were modeled jointly with attack variables. However for the speech synthesis naturalness experiments that will be presented in Section 3.3.4, the embeddings will be created separately such that there will be one embedding for each variable. Since the PA dataset contained six different types of labeled variables, six different types of x-vector embeddings were created. To create the embeddings, the same TDNN architecture was used as in Section 3.2.3 (also from Snyder et al. [2018]). The six different types of embeddings can be distinguished as:

- speakers (**xvec1**)

- room size (**xvec2**)
- T60 reverberation time (**xvec3**)
- talker-to-ASV distance (**xvec4**)
- attacker-to-talker distance (**xvec5**)
- replay device quality (**xvec6**)

The purpose of training different x-vector embeddings is to try to capture the representations that encapsulate the defining categorical features. Since it is not known if speech representations should be specific or very general, these x-vector embeddings offer an attempt to model specific characteristics of speech. By comparing these different x-vector types as the utterance embeddings for the task of automatic naturalness estimation, this could lead to a better understanding of the desired aspects of the optimal embeddings for this task. Each utterance-level x-vector is 512-*dim*.

3.3.4 Experiment Design

This section presents two different architectures and features: one based on frame-level features and the MOSNet architecture of Lo et al. [2019], and the other based on abstract speech representations that were described in Section 3.3.3 and a CNN. Both systems will be described as part of the experimental design.

3.3.4.1 Original MOSNet

Architectures: The original MOSNet neural network architecture is described in detail in [Lo et al., 2019]. Three different architectures were used in their work and are available from Github[6]: BLSTM, CNN, and a CNN-BLSTM combination.
Pre-Trained Voice Conversion Model: The MOSNet architectures from Lo et al. [2019] were originally designed and evaluated for voice conversion speech. The Github repository from the authors includes three pre-trained models, one each for the architectures. Each of the pre-trained models had been developed for estimating MOS naturalness on a Likert scale of 1 to 5 (5 is natural) using data from the 2018 Voice Conversion Challenge [Lorenzo-Trueba et al., 2018]. For the experiments in this chapter, each of the pre-trained VC models will be applied to and evaluated on the LA dataset of synthetic speech for TTS/VC systems.
Re-Trained MOSNet: The re-trained MOSNet experiments will use each of the three architectures as-is, but they will be trained from scratch on the TTS/VC LA dataset. During this re-training, the frame weight parameter will also be explored. In the original MOSNet pre-trained models, the α value for frame weighting was set to $\alpha = 1.0$. This parameter is a weighting of the loss at the frame-level compared to the loss at the utterance level. A value of 1.0 suggests that all frames in an utterance are equal. For the experiments that will re-train the MOSNet on the LA dataset, several values for α will be explored in the range of: $[0.0, 0.5, 1.0]$.

[6] `https://github.com/lochenchou/MOSNet`

3.3.4.2 Convolutional Neural Network (CNN)

All of the other experiments will use a CNN as shown in Figure 3.4. Eight different CNNs will be trained corresponding to each of the 8 different speech representations described in Section 3.3.3. The CNN architecture was implemented with the Keras library [7] with TensorFlow backend [Abadi et al., 2016]. There are four conv1D layers with max-pooling and global average pooling. Each conv1D layer used a kernel size of 10, with L2 regularization [Ng, 2004] and ReLU activation [Hahnloser et al., 2000]. The Adam optimizer [Kingma and Ba, 2014] was used with learning rate $lr = 0.0001$, and early-stopping was governed by validation loss (mean square error). The parameters that were swept include: batch normalization (on/off), L2 value $[0.0001, 0.001, 0.01, 0.1]$, dropout rate $[0.1, 0.2, 0.3]$, number of filters $[16, 32, 64, 128]$, and batch size $[16, 64, 128]$.

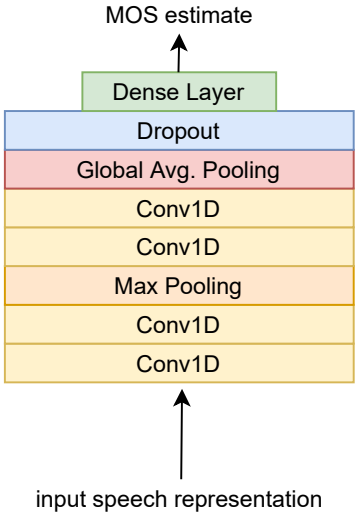


Figure 3.4: Overview of the CNN architecture that was used for experiments comparing 8 different types of utterance-level speech representations for MOS naturalness estimation.

3.3.5 Results

3.3.5.1 Evaluation

Four values were computed for evaluation and these are described in this section, below. Kendall tau correlation was included to assess the ordinal ranking of individual TTS/VC systems and individual speakers. All of the values except for Kendall tau correlation were also reported in the original MOSNet paper [Lo et al., 2019], so this will allow for a meaningful comparison especially because some of the experiments involved the MOSNet architecture.

Linear Correlation Coefficient (LCC) Also known as the Pearson correlation (r), this metric assumes that there is a linear relationship between variables, which may or may not be accurate. The values range between -1 and $+1$, where a score of $+1$ would indicate a perfect correlation. However, this metric alone can sometimes give a high score even if there is not actually a strong correlation because it is sensitive to outliers.

[7] `https://keras.io`

Spearman Rank Correlation Coefficient (SRCC) The Spearman rank correlation (ρ) metric can define correlation but does not restrict this to be at a constant rate between two variables, making Spearman potentially more useful than LCC because Spearman can capture non-linear relationships. Similar to LCC, the values range between -1 and $+1$, where a score of $+1$ would indicate a perfect correlation. It does not have a precise interpretation but it is non-parametric so could model different kinds of relationships. For example, if the distribution of MOS scores in poor-quality TTS is concentrated around particular values, but there is more diversity in high-quality TTS systems.

Mean Square Error (MSE) This metric simply measures the error between the mean predicted MOS score and the mean true MOS score. As a standalone metric, the MSE is not ideal for this task because it fails to capture information about the distribution of scores. For example, if true and predicted scores tend toward a central distribution, a good (low) MSE score may miss characterizing high and low outliers.

Kendall Tau Rank Correlation (KTAU) The Kendall tau (τ) was included for the purpose of evaluating rankings. Specifically, the tau-b was used [Knight, 1966]. The Kendall tau score tends to be more robust than Spearman ρ in terms of error sensitivity [Croux and Dehon, 2010]. It is known to be useful for ordinal rankings. To calculate the Kendall tau values, first the mean predicted score and mean true MOS score are calculated for each speaker or system (both are reported). The speaker or system is then sorted to obtain true rankings and predicted rankings. The Kendall tau is then computed based on these ordinal rankings.

3.3.5.2 Best Model Selection

Given that there are several metrics reported as well as different ways to aggregate results, some decisions were made about how to select the best performing model for each representation in the experiments. Of the metrics that were reported in previous work [Lo et al., 2019; Fu et al., 2018; Patton et al., 2016] the SRCC does best at capturing information about rank and distribution although it should be noted that no metric alone is perfect. When selecting the best model configuration for each representation in the experiments, this decision was based on SRCC aggregated at the speaker-level because the end goal is to understand how different speakers contribute to the overall quality of the multi-speaker TTS system, rather than compare systems. At the same time, metrics aggregated at the system-level are reported even though these did not influence the model selection. This allowed for a comparison to previous work, which only reported results using system-level aggregation.

3.3.5.3 System-Level vs. Speaker-Level Aggregation

One of the reasons for aggregating predictions at the system-level is to characterize an entire system, and compare systems with one another. By aggregating scores at this level, the low-quality systems (such as **A08** HMM-based TTS) will stand out from the high-quality TTS systems (such as **A10** WaveRNN TTS). Thus the correlation metrics are measuring if the trained model can characterize the average quality of a given system. A high correlation value at the system-level can be interpreted to mean that the trained model effectively separates low-quality and high-quality systems. High correlation scores also suggest that the CNN model may generalize well to new systems.

When the scores are aggregated at the speaker-level, this shows how particular speakers

perform in a particular system, or more generally across multiple systems. Some speakers always generate higher quality speech. Given this fact, some inferences can be made about how a particular speaker might contribute in a new TTS system. Similarly, some speakers may consistently produce low-quality speech in any TTS system. Knowing which speakers contribute low-quality speech could mean that these speakers are avoided in future TTS development.

3.3.5.4 Discussion

The results from the experiments are shown in Table 3.3. There is one pre-trained model from MOSNet that was trained on VC data and provided by the authors of that paper [Lo et al., 2019]. There is a second model from the MOSNet architecture that was re-trained in this chapter using the ASVspoof LA dataset. Then there are 8 different types of feature representations that were utilized in the low-capacity CNN architecture. For each feature representation, the best trained model and parameters were selected based on highest speaker-Level SRCC score (yellow). Also highlighted are the system-level SRCC scores (pink). The overall best representation was identified as **xvec6** (blue) because of the high SRCC scores at both system-level and speaker-level, as well as the very high Kendall tau score for ranking speakers.

	Metric	**MOSNet**		**CNN and Representations**							
		Pre-trained (VC-CNN)	Re-trained (LA-CNN)	xvec1	xvec2	xvec3	xvec4	xvec5	**xvec6**	DS	AM
System	**LCC**	0.122	0.717	0.537	0.751	0.495	0.532	0.820	**0.776**	0.918	0.788
	SRCC	**-0.027**	**0.868**	**0.659**	**0.725**	**0.368**	**0.412**	**0.868**	**0.709**	**0.885**	**0.808**
	MSE	1.513	1.277	1.437	1.003	1.208	1.453	0.968	**0.859**	0.388	0.799
	KTAU	-0.052	-0.026	-0.182	-0.026	0.130	0.000	0.000	**-0.156**	0.091	-0.208
Speaker	**LCC**	0.006	0.815	0.465	0.011	0.426	0.267	0.474	**0.694**	0.395	0.398
	SRCC	**-0.033**	**0.883**	**0.433**	**0.017**	**0.383**	**0.367**	**0.717**	**0.800**	**0.500**	**0.583**
	MSE	0.126	0.053	0.111	0.194	0.074	0.095	0.185	**0.091**	0.126	0.104
	KTAU	-0.057	0.114	-0.400	0.514	0.057	0.114	0.343	**0.771**	-0.057	-0.114

Table 3.3: System-level and speaker-level prediction results on the LA dataset, comparing original pre-trained MOSNet VC-CNN model, re-trained MOSNet LA-CNN model, and 8 new representations from best parameters on the CNN architecture. For each, the best model and parameters were selected based on highest speaker-level SRCC score (yellow).

Overall, many of the representations demonstrate similar performance for the chosen metrics. For example, at first glance there appears to be only minor differences between re-trained MOSNet LA-CNN and the DS representation. As a regression task, the DS representation yields a lower MSE which may suggest that the predicted MOS values are closer to target. The DS representation appears to perform well at the system-level, however it is not good at the speaker-level. Several representations stand out that are clearly not suited for distinguishing naturalness for either systems or speakers: pre-trained MOSNet VC-CNN, CNN+**xvec1**, CNN+**xvec3** and CNN+**xvec4**.

The CNN+**xvec6** representation (highlighted blue in Table 3.3) provided the highest Kendall tau score for speakers. Further examination was done on the ranking distributions for individual speakers per-system. From the ground-truth MOS scores, the following two systems were

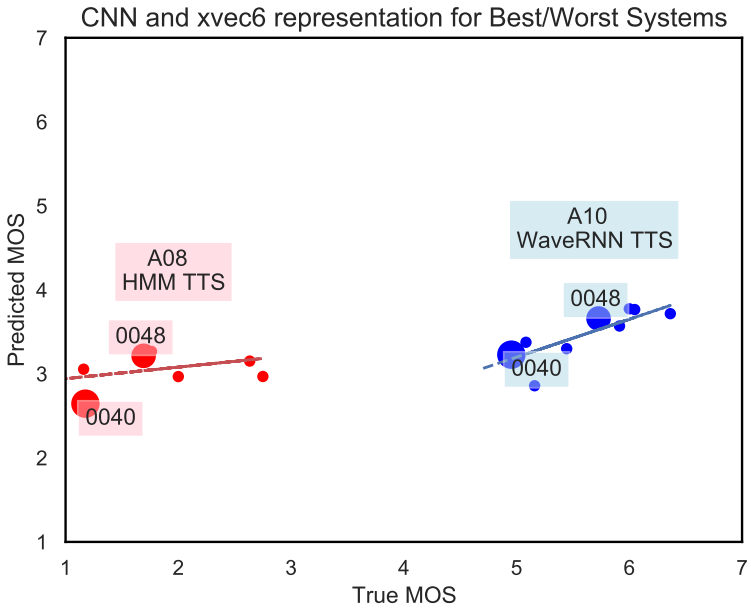


Figure 3.5: Comparison of true and predicted MOS scores using the CNN+**xvec6** representation and CNN model. The two systems shown are the best and worst systems in the LA dataset (based on ground truth MOS scores). The system **A10** (WaveRNN TTS) was rated by humans as the best system in the LA dataset, while **A08** (HMM-based TTS) was rated as the worst. The human MOS ratings were on a scale of 1-10, however this figure shows a scale of 1-7 because it reflects the maximum average scores (i.e., the best system was not rated above 7 on average).

explored because they were ranked by human judgements to be the best and worst in the ASVspoof 2019 LA dataset:

- **A08**: poorest quality system, HMM-based TTS ($MOS_{mean} = 1.79$)
- **A10**: highest quality system, WaveRNN TTS ($MOS_{mean} = 5.58$)

From Figure 3.5 the individual speakers can be seen for each of the best and worst LA systems. Speaker *0048* has the highest overall MOS ratings across all systems in the LA dataset, while speaker *0040* has the lowest. Figure 3.5 shows that the relative relationship between speaker qualities is preserved in both systems. Speaker *0048* is always predicted to perform better than speaker *0040* on average. Of course the absolute difference varies from system to system. The worst speakers in **A10** are as good as, or better than, the best speakers in **A08**. Since these relationships between speakers are relative to a given TTS system, it would not be possible to generalize across all speakers and all systems in the LA dataset. That explains the negative Kendall tau score of -0.156 for the system-level CNN+**xvec6** representation. The speech representation with the highest correlation to ground truth was **xvec6** and the low-capacity CNN used a batch size of 1, with 16 filters, dropout rate of 0.2 and L2 value of 0.0001. Likewise for the reported re-trained MOSNet LA-CNN model, the best-performing configuration used a CNN and $\alpha = 1.0$ for equal frame weighting (as reported from the original MOSNet paper).

The average MOS scores for speakers within a given system are not particularly spread out. Consider that the range of true MOS scores for **A10** is similar to the range of true MOS scores

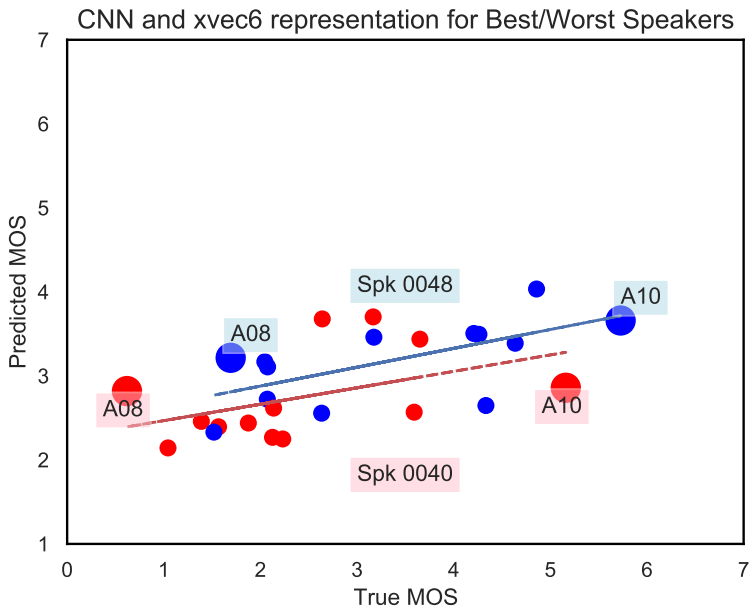


Figure 3.6: Comparison of true and predicted MOS scores using the CNN+**xvec6** representation and CNN model. The two speakers shown are the best and worst speakers in the LA dataset (based on ground truth MOS scores). The best speaker was *0048* and the worst speaker *0040* was. Note each speaker's performance on the best (**A10**) and worst (**A08**) TTS systems. The human MOS ratings were on a scale of 1-10, however this figure shows a scale of 1-7 because it reflects the maximum average scores (i.e., the best speaker was not rated above 7 on average).

for **A08**. While this is very helpful for characterizing systems, it does little for characterizing particular speakers. Therefore one recommendation is that comparison among speakers is done in the context of a particular TTS system. In the LA dataset, the average performing system based on median MOS score was **A11** with $MOS_{mean} = 2.41$ and it was an encoder-decoder sequence modeling TTS system with attention. The average performing system based on mean MOS score was **A07** with $MOS_{mean} = 3.65$ and it was the Merlin DNN statistical parametric speech synthesis (SPSS) with speaker codes and WORLD vocoder.

3.3.6 Speaker Analysis

The experiments with different representations have so far shown that a neural network can be used to predict MOS scores and can differentiate between TTS/VC system quality. In this section the analysis is further expanded to characterize speakers. The CNN using the **xvec6** representation correctly predicts that speakers *0046*, *0047*, and *0048* will tend to have higher MOS scores than other speakers, especially for neural TTS systems (**A12**, **A15**, **A10**). The MOS prediction system also correctly predicts that speakers *0040*, *0042*, and *0044* will tend to have lower MOS scores compared to the other speakers. This is consistently observed when the CNN is trained with the **xvec6** representation. The plot in Figure 3.6 shows that speaker *0040* true and predicted MOS scores are generally lower than those same scores for speaker *0048*. This finding could suggest that speaker *0048* is contributing to a performance increase for most systems since it is demonstrated with ground-truth MOS as well as predicted MOS.

	Metric	MOSNet		CNN and Representations							
		Pre-trained (VC-CNN)	Re-trained (LA-CNN)	xvec1	xvec2	xvec3	xvec4	xvec5	**xvec6**	DS	AM
Speaker	**LCC**	0.570	0.104	0.525	0.439	0.155	-0.021	-0.322	**0.096**	0.319	0.603
	SRCC	0.534	0.102	0.630	0.371	-0.042	-0.060	-0.392	**0.074**	0.347	0.632
	MSE	4.144	3.343	3.503	3.435	4.879	3.937	3.425	**4.320**	6.354	6.240
	KTAU	-0.138	-0.032	0.190	-0.063	-0.138	-0.106	-0.106	**0.296**	-0.159	0.032

Table 3.4: Speaker-level prediction results on Ophelia/LibriTTS synthetic speech. The performance shown here is based on models that were originally trained on the LA dataset (as reported in Table 3.3) before being used to evaluate the multi-speaker TTS from Ophelia/LibriTTS.

3.3.7 Model Generalization

3.3.7.1 Multi-Speaker Ophelia/DCTTS

A held-out TTS system was trained using the Deep Convolutional TTS (DCTTS) architecture [Tachibana et al., 2018], implemented in Ophelia[8]. This is a sequence-to-sequence model with attention. It was trained as a multi-speaker system on a subset of 37,000 utterances across 145 unique speakers from the LibriTTS corpus [Zen et al., 2019]. Ophelia/DCTTS corresponds to a state-of-the-art, fast trainable TTS system, and therefore was a good choice among current architectures at the time that experiments in this thesis were performed [Watts et al., 2019]. As mentioned in section 3.3.2.2, the diversity present in the LibriTTS dataset made this multi-speaker dataset a good candidate to test the automatic evaluation system. Recall that one of the goals of this chapter is to understand how a MOS estimation system may generalize to an unseen or held-out TTS system. The Ophelia/DCTTS system can generate synthetic speech for multiple speakers. However, it could become costly to evaluate many samples from many speakers in a listening test. An automatic MOS estimation tool can identify the best and worst speakers for the system.

3.3.7.2 MOS Naturalness Listening Test

One more listening test was conducted to generate ground truth MOS naturalness scores for the held-out system Ophelia/DCTTS that was trained on LibriTTS. It is worth noting that an alternative to using Ophelia/DCTTS and conducting an additional listening test would have been to use a held-out system from the LA dataset that was already annotated with MOS naturalness scores. However, the purpose of using a completely held-out system and conducting a listening test is to explore how a MOS estimation tool would perform if it was passed to another research lab, for example.

The listening test was designed to replicate aspects of the MOS tests that had been reported for the ASVspoof 2019 LA dataset. For example, the same Likert scale of 1-10 for naturalness was used. Listeners were advised that some speech samples could be from a machine or a human, however the listeners were only presented with TTS speech and no natural samples because the sentences that were synthesized were the Harvard Sentences [Rothauser et al., 1969] which did not have a natural audio reference. The instructions were given as: *"Rate the following*

[8] `https://github.com/oliverwatts/ophelia`

speech samples based on whether or not they sound more like a human (10) or a machine (1). Consider that some samples may sound like a human but have a poor quality of recording or other noise". These instructions were slightly different than the instructions that Wang et al. [2020b] used for rating the LA dataset, which used a "call center" scenario. The sentence content was, for example: "Rice is often served in round bowls.". The meaning of the sentences did not appropriately fit into a "call center" type of scenario.

To generate the synthetic speech utterances, first 20 TTS speakers were selected randomly from the 145 speakers that were used for training. For each speaker, the first 100 Harvard Sentences were synthesized. A total of 20 human listeners were recruited, aged 18 or over, from the University of Edinburgh community. All of the human listeners self-identified as a native speaker, or near-native speaker of English. Each human subject provided a MOS score (on a scale of 1-10), for each of 100 utterances. These were balanced across the 20 synthetic speakers, and randomized. Therefore, in line with the LA dataset ratings, each utterance was heard and marked by one listener but each listener heard utterances from all 20 of the TTS speakers. To elicit MOS scores from the human listeners, a simple web-based interface from Schoeffler et al. [2018] was utilized.

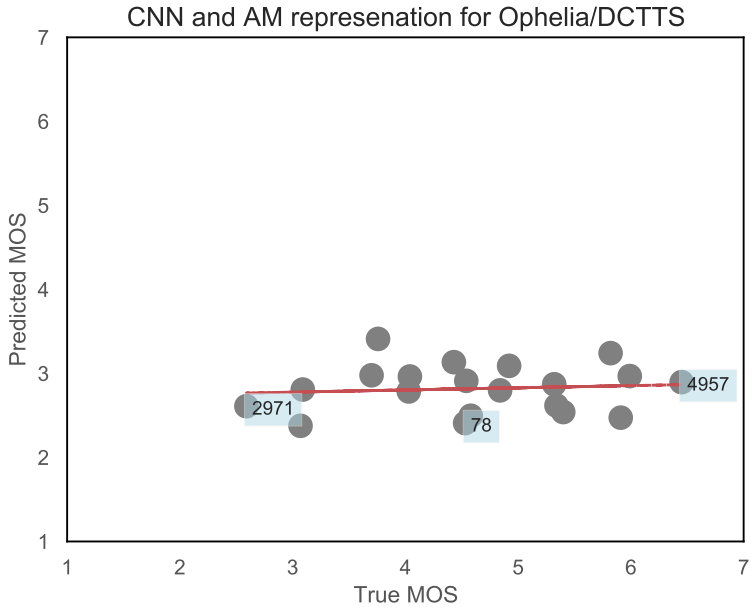


Figure 3.7: Predicted and ground truth MOS scores for 20 speakers from Ophelia/DCTTS. The speakers are marked on the plot for those MOS scores of best (speaker *4957*), worst (speaker *2971*), and mean (speaker *78*).

3.3.7.3 Speaker Analysis

Similar to the earlier analysis of speakers in Section 3.3.6, the predicted and ground truth MOS scores are visualized in Figure 3.7. In this figure, the best (*4957*), worst (*2971*) and average/mean (*78*) speakers are highlighted as determined from the human MOS judgments. From this figure, it can be seen that while the human judgments reflect a spread over a range of MOS scores, the best CNN model still only predicted MOS scores within a central distribution.

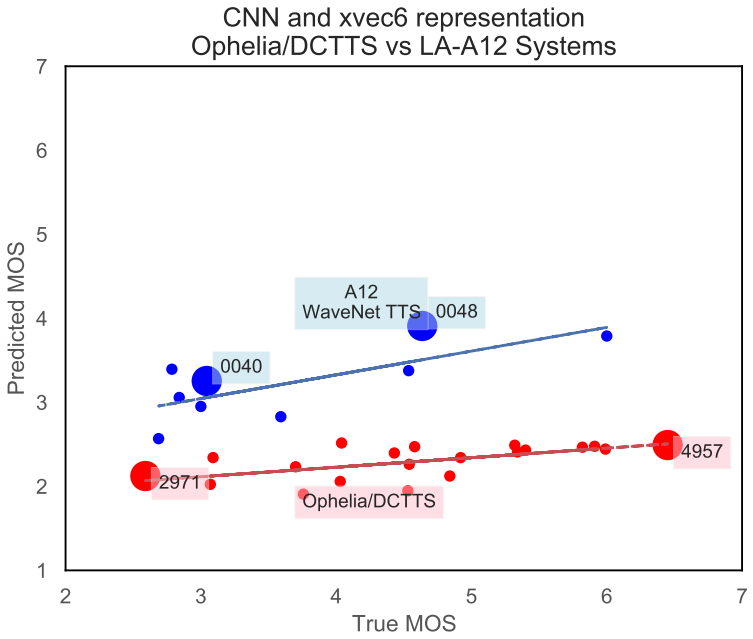


Figure 3.8: WaveNet and Ophelia/DCTTS side-by-side. The Ophelia/DCTTS system can produce some speakers with high quality comparable to WaveNet TTS. In both cases it is difficult to automatically rate the quality given this very small positive correlation. Selectively synthesizing particular speakers could artificially raise or lower the overall MOS score for either system.

This indicates that the approach for predicting MOS does not generalize well to a new TTS system. However the model seems to capture some aspects of the relative speaker quality (e.g., speaker *4957* is predicted to be higher quality than *2971*). But overall there is little correlation between predicted MOS scores and the ground truth MOS scores. The correlation that was reported for SRCC in Table 3.4 for the **xvec6** representation was very low ($SRCC = 0.074$).

Another finding from Table 3.4 is that the pre-trained MOSNet VC-CNN MOSNet model appears to perform better on the Ophelia TTS system, than the re-trained MOSNet LA-CNN based on the LCC and SRCC metrics. There were two TTS systems from the LA dataset for which the pre-trained MOSNet VC-CNN model performed best, and not surprisingly one of them is a similar architecture to the Ophelia/DCTTS (**A11**: encoder-decoder using sequence-to-sequence modeling with attention). System **A12** from the LA dataset had the most similar performance to the Ophelia/LibriTTS. With both of these TTS systems the CNN could rank best/worst speakers, though the ability to rank could be greatly improved with continuing research. This relationship is visualized in Figure 3.8. Of course, these two systems underwent different MOS tests at different times. The mismatch between MOS tests and TTS system types highlights the complexity of the problem for developing a single standalone tool for MOS ratings.

3.3.8 Discussion

The experiments in this section have laid the groundwork for developing a tool that assesses MOS naturalness at a large scale for a variety of VC and TTS systems using multi-speaker data.

Having such a capability would save resources, such as time and costs, and would otherwise facilitate rapid analysis and evaluation. It is important to note that, based on the analysis of re-training the MOSNet architecture on the LA dataset, this approach and technique overall is sound even though the models have some difficulty generalizing to held-out TTS/VC systems. From this outcome, it is suggested that multiple automatic MOS tools are utilized until more work can be done in this area to standardize the tool. One recommendation re-train a MOSNet system when switching from one dataset to the next or one TTS system to the next.

Further, the research direction of relative quality rankings could be useful. Speaker analysis has provided a window to the automatic MOS prediction task from a new angle. The ability to rank speakers is potentially very useful for numerous development and evaluation tasks. Instead of predicting a MOS score during a regression task, one could instead predict relative rankings akin to A/B testing. This would potentially require a different set of metrics and a different loss function as well. Another related problem that this could be used for is to pre-select speakers for TTS training based on the quality of their recordings. While the LibriTTS dataset underwent some cleaning [Zen et al., 2019], it is widely accepted that some speakers in the dataset are better-suited for TTS than others [Oplustil-Gallegos et al., 2020].

Finally, from the experiments on the LA dataset, it can be seen that some speech representations are better at characterizing speech naturalness very well for TTS/VC systems (**DS**) whereas others tend to perform better at characterizing speakers (**xvec6**). The results in this chapter have revealed some of the desired properties of such embeddings. For example, x-vectors that model replay device quality (**xvec6**) were superior to all types of other x-vectors tested.

3.4 Further Considerations

3.4.1 Speech Authenticity

There are several aspects of the work presented in this chapter that warrant further discussion. First, the design of signal-level features that were extracted for replay detection in Section 3.2 were adjusted in a manner to normalize for audio file length. The normalization was performed using a re-sampling method that shortened the length of the audio file to a set number of frames, and then the signal feature coefficient vectors were stacked. While this effectively made each audio file 10 frames long, there are some other ways to consider achieving a similar standard length in frames. It is possible that the re-sampling technique skewed information in the audio file inadvertently.

It may be possible to sample N frames from the audio file at random. The benefit of sampling randomly throughout a file is that a representative sample could be taken and the sample would reflect actual audio frames. A downside to using random samples of audio is that for spoofing, there may be audio artifacts correlated to the spoofing attack and these artifacts may occur at only one frame, multiple frames, randomly, or not at all. Randomly sampling the audio frames could miss the necessary frames that contain artifacts. Another technique to normalize for length is to average across all frames so that the feature vector is represented by M coefficients. For example, if an audio file has 100 frames and 80 MFCCs, then the resulting feature vector would have dimensionality of 1x80 after averaging over all of the audio frames. The potential downside to this method of averaging is that temporal information could be lost, or if there are particular audio artifacts these may be averaged out. A time-delay neural network (TDNN)

could overcome the issue of variable audio file length, however in the case of the ASVspoof 2019 challenge one objective was to explore novelty and the TDNN has been heavily explored in previous work.

3.4.2 Speech Naturalness

At the time of this writing, there is active and ongoing work on developing automatic tools for speech naturalness prediction [Cooper and Yamagishi, 2021]. The tool developed in this chapter has only addressed one aspect of synthetic speech evaluation (e.g., subjective naturalness ratings). There are other open problems related to naturalness, including simulating a MUSHRA test or estimating speaker similarity. Speaker similarity is another very important aspect and it is often overlooked as a research area. One reason why speaker similarity is important is because it relates to consistency for synthetic speech. This applies to TTS as well as VC. Speaker similarity was not investigated in Section 3.3 because the focus was to develop a naturalness tool and characterize which speakers contributed most to high or low MOS naturalness. Speech synthesis continues to mature as a research field so more additional tools will be needed such as a way to characterize prosodic variation or emotion expression.

3.5 Conclusion

This chapter has explored speech representations for automatically assessing the authenticity and naturalness of speech in different scenarios. In the case of speech authenticity detection, the special x-vectors performed better at detecting replayed speech than signal-level features alone. In the case of estimating MOS naturalness for synthetic speech, another type of special x-vector performed better at MOS prediction than frame-based features. Abstract representations of speech model general characteristics at a specified segment level, in this chapter that has been the utterance-level. These representations allow information to be distributed across multiple dimensions of a feature vector. Even when speech representations are rather high dimensional, as with the Deep Spectrum vectors which were 4096-*dim*, this dimensionality is still lower than speech in the time or frequency domain. When researching in new areas such as authenticity detection, it is not always possible to know which signal-level features are best-suited for a task. It may be possible to uncover new signal features which are specific to a task. It is not easy to discover new signal features for every possible research and development problem in speech processing. Machine learning affords an opportunity to exploit the nature of information in the speech signal using representation learning. As the experiments in this chapter have shown, there are many different types of information contained in an abstract representation. For example, a single x-vector can be trained to model device quality in the MOS estimation experiments, yet also be used to characterize the synthetic speech of individual speakers. In Chapter 4, this idea will be explored further through experiments that seek to separate types of information contained within a single abstract representations using disentanglement techniques. In Chapter 5, the MOS estimation tool developed here will be exploited to speed up the development process. The estimated MOS prediction scores will be examined in the context of another listening test. In later chapters, the idea of using speech representations for a variety of speech processing tasks will be examined and evaluated.

Chapter 4

Methods to Disentangle Representations of Speaker Identity

4.1 Introduction

Advances in neural machine learning have made their way from other data domains (such as the image domain) into speech processing. One thing that makes deep neural networks (DNNs) attractive for speech processing is that they can be trained to perform well for complex classification even when the input features are very high-dimensional or vary over time. Speaker recognition is one such task in speech processing. The field of speaker recognition has benefited greatly from DNNs [Snyder et al., 2016, 2018, 2017; Heigold et al., 2016]. In addition to the ability to learn to classify speakers, the internal layers of trained DNNs are also very important for speaker recognition. In fact, the activation values for certain layers of trained networks such as time-delay neural networks (TDNNs) or bottleneck features, can function as learned representations of speaker identity. The representations learned from TDNNs are often referred to as x-vectors [Snyder et al., 2018], whereas the ones learned from bottleneck features are termed i-vectors [Dehak et al., 2011]. These representations of speaker identity that are learned from DNNs can also be utilized for other speech tasks including (but not limited to) speaker diarization [Sell et al., 2018], speaker age classification [Grzybowska and Kacprzak, 2016], and assessing Alzheimer's disease [López et al., 2019].

Traditional speaker embeddings, such as i-vectors and x-vectors (introduced in Chapter 2.2.2) are optimized to model speaker identity for tasks such as speaker recognition, speaker verification, and speaker diarization. While they have been designed for performance in speaker recognition tasks they also contain extra information that is unrelated to speaker identity such as style, content, speaking rate, and gender [Peri et al., 2020a; Raj et al., 2019]. These representations aim to maximize the variance between speakers, while minimizing within-speaker differences (recording device, mood, age, etc) [Kinnunen and Li, 2010]. Experiments in this chapter will demonstrate that style and emotion information can be found within utterance-level speaker embeddings. The objective of these experiments is to develop a framework that can separate style and speaker information. Successful factorization of speaker embeddings in this way could later be exploited for various speech applications from controlling the speaker voice in text-to-speech synthesis (TTS) to converting speech into a target speaker in voice conversion (VC).

In the process of training high-quality speaker embeddings for speaker recognition, one necessary step is to separate speaker-invariant characteristics from residual channel information [Chen and Lin, 2020]. The channel information contains factors related to the recording device and session noise. Historically, large datasets for speaker recognition were developed with mixed recording session variables or mismatched recording conditions to the actual application. For example, data for a given speaker may have been recorded with a variety of different quality microphones. Sometimes the quality of speech that was used for training algorithms did not match the quality of speech used in an application (e.g., telephony speech). From this mismatch came the need to neutralize characteristics that vary across a given speaker's recordings. This means removing factors of variation for a given speaker, such as mixed microphones or mixed quality recordings. Speaker-variant characteristics have been semantically grouped together and referred to simply as the "channel". When speaker embeddings are created, the final embeddings for a given speaker discard channel information, leaving only speaker-invariant characteristics. In both i-vectors and x-vectors, the channel factorization is performed using Probabilistic Linear Discriminant Analysis (PLDA). Therefore the experiments in this chapter utilize the utterance-level representations obtained before PLDA.

The experiments presented in this chapter investigate four categories of speaking style: spontaneous conversation, goal-directed interaction, retold speech, and read speech. As the term *speaking style* does not have a single definition, a working definition is adopted to be: *how speakers adapt their speaking manner according to the speaking context.* Alongside the experiments which explore speaking style, this chapter presents another separate set of experiments that explore emotion. The experiments involving emotion explore four basic categories: angry, happy, sad, and neutral. The datasets used for speaking style and emotion experiments are different datasets, and will be described in Section 4.3.

There is some evidence that utterance-level i-vector representations (before applying PLDA) can discriminate different emotions in speech [Jauk, 2017] when utilized as feature vectors for training a classifier. This is despite the fact that i-vectors are designed for speaker recognition. It is hypothesized that the emotion information can be found within the "channel factor", as it is traditionally referred to in the field of speaker recognition. If information about speaker emotion does reside within the channel factor, then both i-vectors and x-vectors will also contain this information in utterance-level representations. In that case, the channel factor would more appropriately be referred to as a *style factor* or *emotion factor.* With this in mind, the methods developed in this chapter take an alternative approach that is based on deep autoencoders. Autoencoders offer an attractive solution because, if the reconstruction is very good, then it becomes possible to learn very rich latent representations. This chapter explores whether or not autoencoders can learn to separate different kinds of information (such as speaking style and speaker identity) into separate representations. This chapter makes three main contributions as follows:

1. Show that utterance-level i-vectors and x-vectors contain information about speaking style and emotion

2. Compare several disentanglement methods

3. Evaluate disentanglement using accuracy for classification tasks

4.2 Related Work

4.2.1 Speaking Style and Emotion

There are systematic and measurable differences when a speaker expresses a particular emotion or speaks in a particular context. Two well-studied contexts are *read* and *spontaneous* speech. For example, differences were found between news broadcast speech and freestyle conversations and these differences include intonation patterns ($F0$) and speaking rate [Hirschberg, 2000]. Previous work has shown that content words are more likely to carry marked pitch accents in spontaneous speech than in read speech [Yuan et al., 2005]. On the other hand, there are fewer pitch accents overall in spontaneous speech does not seem to influence variation as much as the speaking style [Mixdorff et al., 2005].

Speaking style has been investigated in the context of utterance-level speaker vectors, before DNNs were part of the process of creating speaker representations. Work from Asami et al. [2014] showed that GMM supervectors [Reynolds, 1995] can be used to distinguish between spontaneous or read speech at the utterance level with $> 95\%$ accuracy, independently of any knowledge about speaker identity. Their use of GMM supervectors aligns to the experiments in this chapter since GMM supervectors form the basis for i-vectors (described earlier in Chapter 2). All of this prior work suggests that it is possible to arrive at a representation of speaking style and emotion.

4.2.2 Expressive Speech Synthesis

Expressive speech synthesis is one application that benefits from robust representations of style and emotion. Recently Wang et al. [2018] introduced a method to construct and apply *style tokens* in an effort to model style and speaking rate. These tokens, which are essentially learned embeddings, represent a set of discovered latent representations of style in speech data rather than human-labeled categories. The final style embeddings are not convincingly categorical to listeners. Further, as in Wang et al. [2017b], there is an underlying assumption that *style tokens* can be learned directly from features that have been extracted from the speech signal, such as $F0$, and this is somewhat contradictory to other work that calls for higher-level abstract representations.

Others have explored abstract representations for expressive speech synthesis, including Jauk [2017], who showed that i-vectors can form unsupervised clusters corresponding to emotion categories. They showed that i-vectors can be utilized for expressive synthesis. However, this approach did not remove speaker-invariant characteristics, making it difficult to control speaker identity and emotion independently. Recent work has explored variational autoencoders (VAEs) for expressive speech synthesis with VoiceLoop, but they were unable to model global characteristics of style, possibly because the approach was fully unsupervised [Akuzawa et al., 2018].

4.3 Dataset Description

Two different datasets are used in this chapter: one that is labeled for speaking styles and another that is labeled for speech emotion. The primary reason for exploring both emotion

and style was due to a lack of existing baselines for classifying a variety of speaking styles from speech in this manner. Recall that earlier work had explored read versus spontaneous speech. The dataset used in this chapter includes more than these two speaking styles. At the same time, there is existing work on classifying emotional categories in speech using the same emotion dataset that is used in this chapter. By evaluating the same disentanglement methodology applied to both style and emotion, it is possible to understand the overall performance of the method.

4.3.1 The IViE Corpus for Speaking Style

The Intonational Variations in English (IViE) corpus[1] was originally collected for the purpose of exploring 9 regional/dialect variations throughout the United Kingdom. The creators of this dataset had elicited four different speaking styles from participants: spontaneous conversational speech (**Conv.**), goal-directed interaction (**Directed**), retold spoken passages (**Retold**), and read spoken passages (**Read**). The spontaneous speech conversation category used same-gender pairs who discussed the topic of cigarette smoking. The goal-directed interaction involved a version of the conventional map-task (i.e., providing directions while reading a map). The retold and read passages involved an excerpt from the story *Cinderella*[2]. The main difference between retold and read speech is that for retold speech, the participant had read the passage and then was recorded reciting it from memory. The read speech was recorded as the participant read the text of the speech aloud.

Each of the speakers is approximately 16 years old, and the collection was split evenly between males and females. For the purposes of experiments in this chapter, gender and dialect labels were disregarded such that the experiments focus only on these four style categories. For the spontaneous conversation and goal-directed interaction categories the audio had not been diarized by the dataset creators, and was not diarized for the experiments in this chapter. However, the experiments in this chapter required a speaker label for each utterance so the speaker labels were combined together and the speaker label of an utterance was set to be the label of the first (primary) speaker. A description of the data split, number of unique speakers, and average utterance duration (in seconds) is provided in Table 4.1.

4.3.2 The IEMOCAP Corpus for Speech Emotion

The Interactive Emotional Dyadic Motion Capture (IEMOCAP) [Busso et al., 2008] dataset was used because it contains labeled emotion categories and this was helpful for making a comparison to previous work for the experiments that were conducted in this chapter. Ten professional actors were prompted to enact hypothetical situations or to read directly from a script while performing emotions. Although this dataset is multimodal (speech, video, head movements, transcription, etc), only the audio was used for experiments in this chapter. When the dataset was created, the curators had employed human annotators to label utterances with categories of emotion. Their annotation procedure allowed utterances to have multiple labels. To overcome this issue, for the experiments in this chapter a subset of four emotions was selected wherein each of the utterances had only one emotion label (**Angry**, **Sad**, **Happy**, **Neutral**). More

[1] `http://www.phon.ox.ac.uk/ivie`
[2] `http://www.phon.ox.ac.uk/files/apps/IViE/stimuli.php`

		Train	Valid	Test	S	$dur(s)$
IEMOCAP	**Angry**	318	56	125	10	4.8
	Sad	269	48	106	10	5.6
	Happy	60	10	23	10	5.0
	Neutral	247	44	97	10	4.4
	Totals	894	158	351	10	4.9
IViE	**Conv.**	210	37	82	108	50
	Directed	426	75	168	112	31
	Retold	558	99	219	110	27
	Read	431	76	169	110	44
	Totals	1625	287	638	112	38

Table 4.1: Number of utterances for IEMOCAP and IViE datasets, where S is the number of unique speakers in the category, and *dur* is the average duration of audio segments in seconds.

detail about the data split is provided in Table 4.1. While this significantly reduced the overall size of the data, it circumvented issues of label reliability from the annotators.

4.4 Methodology

The experiments in this chapter explore several configurations of a DNN autoencoder architecture to separate different types of information contained within x-vectors and i-vectors. The autoencoder will attempt to move style/emotion information into one latent space, and move speaker information into another latent space. This is similar to the role that PLDA serves, except that information is moved into another representation rather than removed and discarded. Another difference between the proposed architectures and PLDA is that PLDA mathematically distinguishes between factors that vary for a given speaker from factors that are consistent for a given speaker. In the experiments for this chapter, the style/emotion labels would be treated like factors that vary, and therefore are taking the functional role of "channel".

Utterance-level embeddings are projected into lower dimensions using a vanilla autoencoder (**AEV**) or principle components analysis (**PCA**). The reason for using lower dimensions is to observe any effects of compression and to understand if there are any limits to compression which render the embeddings meaningless, as measured by classification accuracy described below. Additionally, the embeddings are projected into lower dimensions as well as into a pair of latent spaces using a several different techniques that involve autoencoders. One latent space is intended to contain *only* style (or emotion) information, and the other to contain *no* style (or emotion) information. To quantify how disentangled the latent spaces were, each of the latent spaces were utilized to train a separate classifier. This was done to probe whether or not the learned latent spaces contained any information related to style (or emotion) by examining whether or not the resulting classification accuracy was high or low. The act of probing speaker representations using classification is an established technique [Raj et al., 2019]. Another way to demonstrate that learned latent spaces of the autoencoder contained complementary information was to degrade one of the latent representations and examine the autoencoder

ability to reconstruct the input. This degradation was performed on the learned latent space that was meant to contain *no* style (or emotion) information. It was reasoned that if one latent space has been degraded and the autoencoder is unable to reconstruct the input, then both of the learned latent spaces contain complementary information. Explained differently, if the autoencoder cannot reconstruct the input when one latent space is degraded this indicates that both latent spaces are necessary. On the other hand, if the autoencoder can reconstruct the input when one latent space is degraded this indicates that the decoder has learned to ignore one latent space, effectively rendering that latent representation meaningless. The latter case is often referred to as *posterior collapse* and is a problem that can sometimes arise when working with autoencoders.

4.4.1 Utterance-Level Embeddings

The Kaldi Toolkit [Povey et al., 2011] was used to extract utterance-level i-vectors and x-vectors. As described in Chapter 2, the creation of i-vectors and x-vectors includes training a DNN model along with several additional components. The pre-trained models[3] described in [Snyder et al., 2018] were used as they also provided PLDA, mean vectors, and transform vectors which had been trained and evaluated on the VoxCeleb corpus. That corpus contains approximately 2,000 unique celebrity speakers [Nagrani et al., 2017; Chung et al., 2018] and was augmented with noise during training. The VoxCeleb data is considered spontaneous and 'in the wild' and it is also known to exhibit natural emotion [Albanie et al., 2018]. The front-end configuration was provided fully-specified with the following settings. The audio signal sampling rate was 16 kHz and the frame length was 25 ms. For feature-level vocal-tract length normalization (VTLN), the low-frequency cutoff was 20 Hz and the high-frequency cutoff was 7600 Hz. The features were 24 MFCCs for i-vectors and 30 MFCCs for x-vectors. The number of mel-cepstrum filterbank bins was 30. The features were mean-normalized over a sliding window up to 3 seconds.

Utterance-level embeddings in this chapter come from deep neural networks. For x-vectors the neural network is a TDNN (Chapter 2.2.2.4) and the embedding is the first fully-connected layer after frame-level statistics pooling. Each embedding is 512-*dim*. The i-vectors come from phonetic bottleneck features of a DNN-based universal background model (UBM) where each utterance-level embedding is 400-*dim* [Snyder et al., 2017].

4.4.2 Dimensionality Reduction

One aspect of the experiments was to find out to what extent the i-vector and x-vector utterance embeddings could be compressed while retaining style (or emotion) information. The embeddings were projected into lower-dimensional latent spaces using either PCA or the vanilla autoencoder of Figure 4.1, with varied dimensions in the range of $dims = [512, 400, 300, 200, 100, 50, 20, 10]$ (omitting 512-*dim* when using i-vectors). All such autoencoders consisted of 13 fully-connected dense layers for the encoder and 13 fully-connected dense layers for the decoder with the same training parameters as our DNN classifier: ReLU activation [Hahnloser et al., 2000], $L2$ regularization [Ng, 2004], Adam optimizer [Kingma and Ba, 2014] with learning rate $lr = 0.0002$, and early-stopping monitored by validation loss. The input embeddings were mean normalized[4].

[3] `http://kaldi-asr.org/models/m7`

[4] This process is often referred to as z-score normalization. It scales the features by subtracting the mean and dividing by the standard deviation. The mean and standard deviation were calculated over the training data.

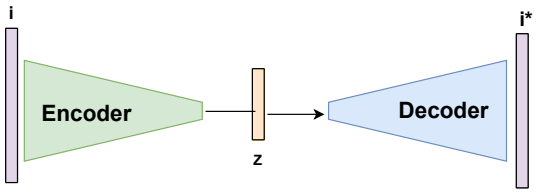


Figure 4.1: Vanilla autoencoder (**AEV**) showing a single latent space z. The input is compressed into a latent representation. The decoder learns to reconstruct the input from the latent representation.

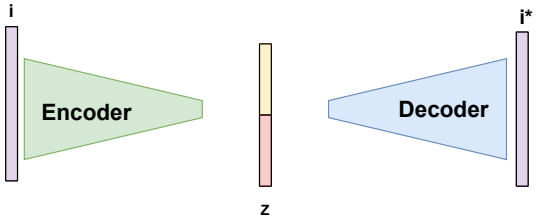


Figure 4.2: Vanilla autoencoder showing a split latent space $z1$ and $z2$. While there are two latent spaces, the architecture does not have incentive to learn which type of information belongs in $z1$ versus $z2$.

4.4.3 Disentanglement Solutions

The reasoning behind splitting the latent space from one single latent z (shown in Figure 4.1) into two latent spaces (shown in Figure 4.2), was to move information of one type into one latent space, and information of another type into another latent space. If successful, this would approximate a solution for disentanglement. While the architecture in Figure 4.2 does contain two separated latent spaces, $z1$ and $z2$, there is no incentive for the architecture to learn how to perform this kind of information separation. To overcome this challenge, auxiliary classifiers were attached to the latent spaces $z1$ and $z2$, in a multitask paradigm. This was first attempted with a single encoder and later with two encoders. It was found that the single encoder approach did not perform well and became unstable during training. The single-encoder approach was therefore abandoned for a dual-encoder approach.

The proposed autoencoder for disentanglement has two encoders and two auxiliary classifiers, shown in Figure 4.3. The $z1$ and $z2$ latent spaces each have a separate auxiliary classifier. To cause $z1$ to encode style, its auxiliary style classifier is trained to minimize a cross-entropy loss. To cause $z2$ to encode anything *except* style, various approaches were explored.

First the baseline autoencoder (**AE1**) was established using two latent spaces and two auxiliary classifiers, and is shown in Figure 4.3. However, the **AE1** architecture did not sufficiently separate information between the latent spaces. To cause the residual latent space $z2$ to contain as little style information as possible, the **AE2** architecture variant was tried where the $z2$ space was reset to the batch mean before being passed to the decoder. In another variant called **AE3**, the auxiliary classifier was set to maximize cross-entropy loss rather than minimize it. Another variant called **AEC** degraded the $z2$ space using noise (described below) while training the decoder, and the auxiliary classifier was also configured to maximize the cross-entropy loss. The degradation of $z2$ means that the values in the $z2$ representation were

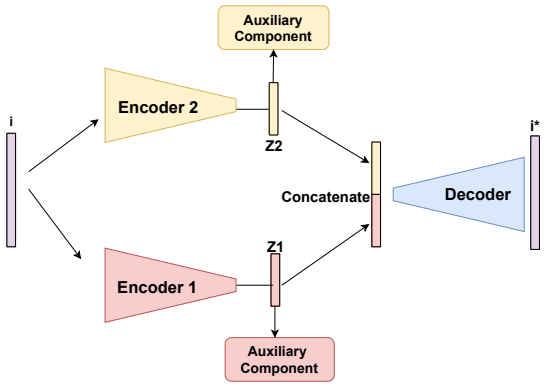


Figure 4.3: Autoencoder with dual-encoders and single decoder. Each latent variable is provided as input to an auxiliary component that attempts to classify speaker or style/emotion.

made less reliable due to the addition of noise.

The Gaussian noise layer in **AEC** was only active during training. The noise layer is an effective tool for forcing corruption as it is used here, however this layer is most often used as a tool to avoid over-fitting during training. This noise is "additive" in the sense that it is in addition to any existing noise inherent in the z-representation. The noise is "white" because it has a uniform power across frequencies. The noise is also Gaussian because it follows a normal distribution and has zero mean. The tunable setting for this noise was the standard deviation. In these experiments it was set to $std = 0.5$. This standard deviation was chosen arbitrarily. Generally speaking, a high value results in more noise and a low value results in less noise. When the Gaussian noise is used in machine learning to avoid over-fitting, a reasonable value is often $std = 0.1$ or some magnitude smaller $std = 0.01$.

The loss function for these multi-task autoencoders (**AE1, AE2, AE3, AEC**) is described by Equation 5.1. The loss of the multitask system L_{AE} is represented by three terms. Here, the L_R term is the autoencoder reconstruction loss as the mean square error (MSE), and was used in all versions of the autoencoders that were experimented with. The L_{Aux1} is the loss of the auxiliary classifier on the $z1$ space, and it measured categorical cross-entropy. Finally L_{Aux2} is the loss of the auxiliary classifier on the $z2$ space, and it also measured categorical cross-entropy. When the maximized cross-entropy loss was used on the residual $z2$ auxiliary classifier, a weight (γ) was introduced to balance the losses between the autoencoder reconstruction and the auxiliary classifier.

$$L_{AE} = L_R + L_{Aux1} + \gamma L_{Aux2} \tag{4.1}$$

The loss weight for the auxiliary classifier on $z2$ (L_{Aux2}) was set to $\gamma = 0.05$ to help balance the magnitude of the losses. If the weights are not balanced, at least in the same orders of magnitude, then the algorithm may disproportionately over-train one portion of the architecture while leaving other portions under-trained. This value was chosen arbitrarily and was based on the order of magnitude of the losses as they were observed in trial runs. Setting optimal loss weights is itself an open problem in multi-task learning and was not fully explored in these experiments. A description of the set of autoencoder configurations is presented in Table 4.2 showing which ones maximize cross-entropy loss, or corrupt the $z2$ representation.

Technique	Num Encoders	$z2$ Corruption	Max Loss
PCA	–	–	no
AEV	1	–	no
AE1	2	–	no
AE2	2	μ batch	no
AE3	2	–	yes
AEC	2	full	yes

Table 4.2: Description of disentanglement methods. PCA and AEV do not attempt disentanglement and are used for examining compression before any disentanglement.

4.4.4 Style / Emotion Classification Experiments

After each variation of the autoencoders were trained, $z1$ and $z2$ representations were obtained on a held-out test set. These $z1$ and $z2$ representations were then used to train a completely separate classifier for style or emotion. These classification tasks were designed to probe how much style or emotion information was contained within each type of representation, from each autoencoder configuration. These classification tasks were completely separate from the auxiliary classifiers that were used for multi-task autoencoder training. Here, new DNN classifiers were trained using the learned latent representations as feature vectors for input. The style classification was done separately from emotion classification. The same train/test split was used as described in Table 4.1. Each DNN consisted of three fully-connected dense layers with ReLU activation ($alpha = 0.2$) and $L2$ regularization ($L2 = 0.0001$). The optimizer was Adam with learning rate set to $lr = 0.0002$ and the remaining parameters were kept as default. The loss function was cross-entropy and early stopping was also used while monitoring validation loss. The input was mean normalized. The first classifier was trained on raw i-vectors or x-vectors which had not undergone any disentanglement. Subsequent classifiers were trained on the compressed representations from either PCA or the vanilla autoencoder. Finally, classifiers was trained using the disentangled latent spaces of $z1$ or $z2$ from trained autoencoders AE1, AE2, AE3 and AEC.

4.5 Results

4.5.1 Autoencoder Reconstruction

One demonstration of disentanglement is to examine the ability to reconstruct the original input i-vector or x-vector from the latent space. As a reconstruction baseline, an average i-vector or x-vector was calculated over the training data and compared to each training example to calculate the upper bound mean absolute error (MAE). Any MAE values above this bound would indicate that the autoencoder reconstruction was very poor. The upper-bound baseline MAE for i-vectors was 0.75 for both IEMOCAP and IViE datasets. For x-vectors it was 0.62 for IEMOCAP and 0.63 for IViE. The MAE upper bound was exceeded with x-vectors when using the **AEC** model on the IViE data (MAE $>$ 0.64). The MAE upper bound was approached and exceeded for the **AEC** model on IEMOCAP data for both types of input vectors.

High reconstruction error from the **AEC** model (when $z2$ is corrupted) indicates that *both* $z1$ and $z2$ components are needed for reconstruction. That is, they contain complementary – or disentangled – information. In other models, the MAE upper bound was not exceeded and this suggests that the decoder for those models was utilizing both $z1$ and $z2$ latent spaces.

4.5.2 Style / Emotion Classification Results

Another demonstration of disentanglement was to use each of $z1$ and $z2$ in turn as the input to style (or emotion) classifiers described in Section 4.4.4. The classification results are reported on held-out data in Figure 4.4 and Figure 4.5. The plots (a, b) in Figure 4.4 show classification accuracy for methods that do not uniquely disentangle style in the $z1$ and $z2$ latent spaces (**PCA, AEV, AE1, AE2**). The set of plots (c, d) Figure 4.4 show classification accuracy for methods that were successful in isolating factors of style in $z1$ from residual in $z2$ (**AE3, AEC**). Since the $z2$ latent space in **AEC** was corrupted artificially using Gaussian noise, it is expected that the $z2$ space would lose information, as reflected in plots c and d of Figure 4.4. Therefore the autoencoder configuration which has been successful at disentanglement (as defined in this chapter) is the **AE3** system. At the same time, notice in Figure 4.4 that there is a non-zero accuracy for the $z2$ space which makes it difficult to show an absence of information. Therefore the loss of accuracy should be interpreted as a relative loss of accuracy. This same reasoning also applies to the upper and lower plots in Figure 4.5 which are for the emotion dataset.

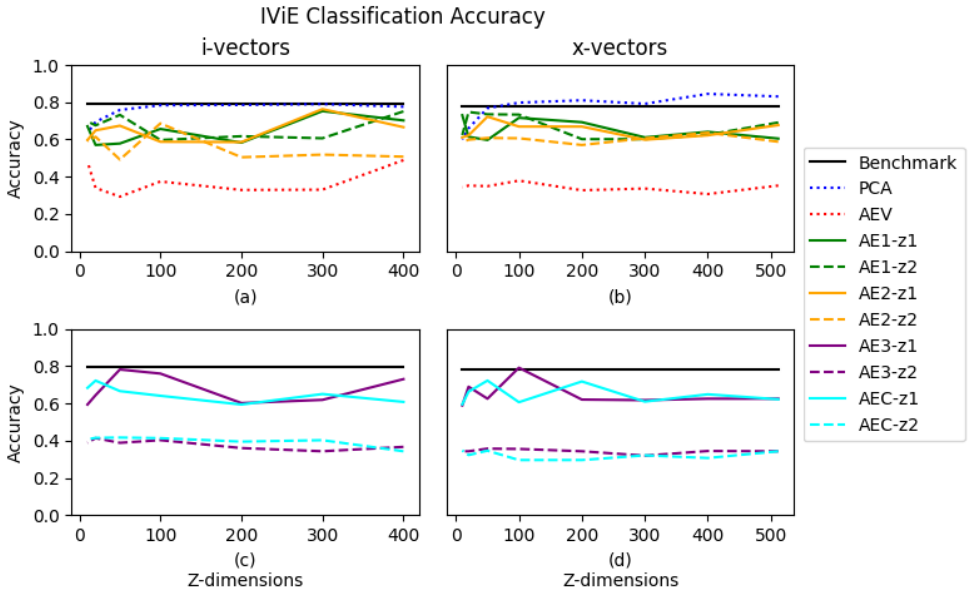


Figure 4.4: Style classification accuracy results before disentanglement (top) and after (bottom), with benchmarks constant for comparison. The benchmarks use raw i-vectors or x-vectors respectively as input and are shown in the plots as a constant horizontal line indicating classification accuracy without any compression or disentanglement. On IViE: 79% and 78%. Z-dimensions reflects the size of $z1$ and $z2$ as dimensionality reduction was explored.

Overall, the $z2$ space has lost information about style and emotion in **AE3, AEC**. On the other hand, the $z1$ space shows that style and emotion information was preserved even with dimensionality reduction for latent dimensions. In Figure 4.4 (plots c/d) and Figure 4.5 (plots

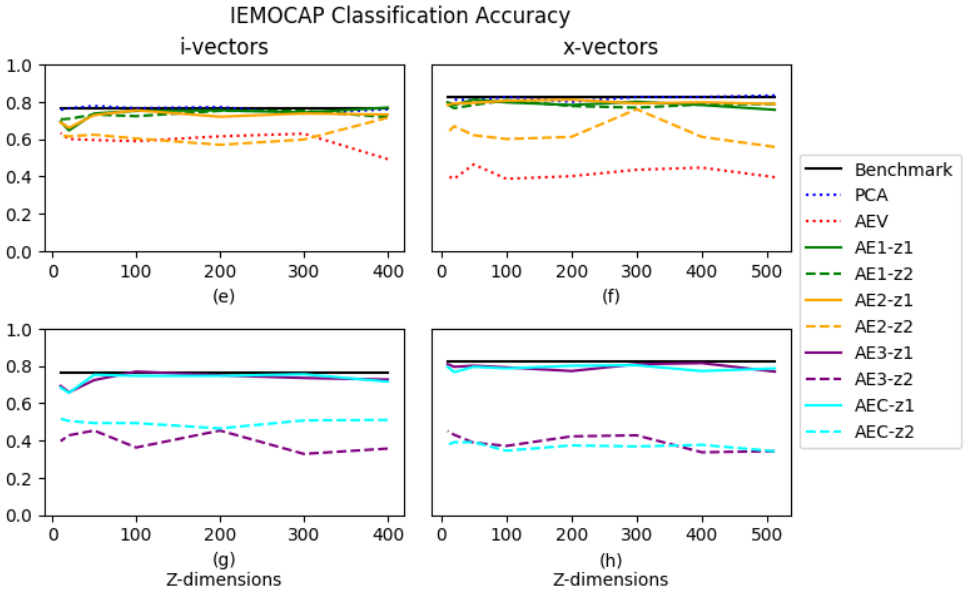


Figure 4.5: Emotion classification accuracy results before disentanglement (top) and after (bottom), with benchmarks constant for comparison. The benchmarks use raw i-vectors or x-vectors respectively as input and are shown in the plots as a constant horizontal line indicating classification accuracy without any compression or disentanglement. On IEMOCAP: 76% and 82%. Z-dimensions reflects the size of $z1$ and $z2$ as dimensionality reduction was explored.

g/h), the accuracy with dimensionality reduction for $z1$ performed close to benchmark, even when the dimensions were reduced by more than half. This suggests that the style/emotion information in the $z1$ latent was robust. The overall best classification accuracy for style and emotion came from **AE3**. Recent work from [Satt et al., 2017] on 4-class emotion classification on IEMOCAP achieved 68.8% overall accuracy. The classification done in [Satt et al., 2017] was based on features that were taken directly from the speech spectrograms. The exact train/test splits from their work were not available, so a direct comparison cannot be made between their results and the results reported in Figure 4.5. However, their work does provide a sense of how easy or difficult the task is. Variation between their reported accuracy and the accuracy reported in Figure 4.5 could be due to the method as well as differences from the training sets and labels. For example, Figure 4.5 plot g indicates that the $z1$ latent for **AE3** had 4-class accuracy of approximately 80% across various reduced dimensionality, but that accuracy fell slightly to approximately 70% when the dimensions approached 10-*dim*.

Figure 4.6 and Figure 4.7 show the classification accuracy for style and emotion classes, for each dataset. These results are for the best-performing disentangled $z1$ encodings, which came from **AE3** for both style and emotion. The best dimensionality for IViE i-vectors was 50-*dim*, and for x-vectors was 100-*dim*. In the IViE style prediction tasks, spontaneous conversation was often mistaken as retold speech. This may be a consequence of using non-diarized conversational speech although it did not seem to affect the goal-directed speech style which was also not diarized. The best dimensionality for IEMOCAP i-vectors was 300-*dim*, and x-vectors was 400-*dim*. The poor performance on 'happy' is likely related to class imbalance, similar to previous work [Hodari et al., 2018] and this class was often misidentified as 'sad' or 'neutral'.

4.5.3 Speaker Recognition Results

A final evaluation was conducted to compare how well the $z1$ (*style factors*) and $z2$ (*residual*) retained *speaker* identifying information. This was measured using traditional PLDA-based speaker classification. The original extracted i-vectors and x-vectors for IViE data discriminated speakers with below 10% equal-error rate (EER)[5] while for IEMOCAP the EER was above 30% EER. For all of the $z1$ and $z2$ representations in both datasets, the EER was always greater than 30%. This suggests speaker information was lost. At the same time, the style information was preserved as demonstrated by the classification results discussed earlier. The IEMOCAP EER was consistently so poor, which is likely due to mismatches between the IEMOCAP data and the PLDA model (the PLDA component was pre-trained and provided alongside the x-vector model).

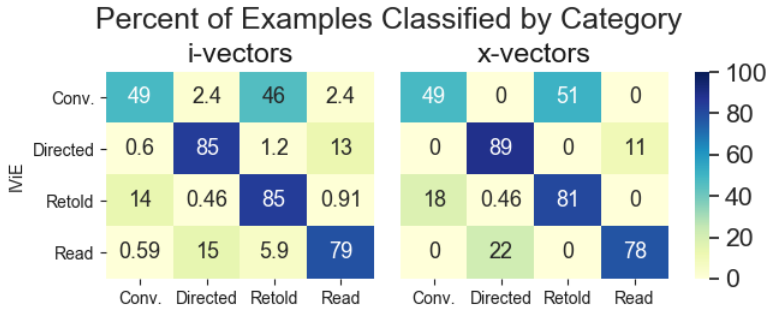


Figure 4.6: Classification accuracy for style categories in the IViE dataset for x-vectors and i-vectors.

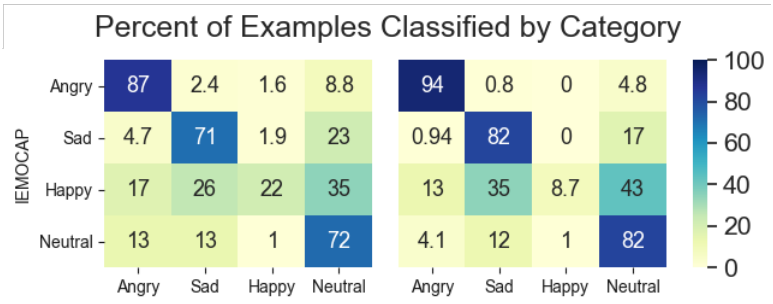


Figure 4.7: Classification accuracy for emotion categories in the IEMOCAP dataset for x-vectors and i-vectors.

[5]Equal-error rate (EER) is a standard metric for speaker recognition (described in Chapter 2.2.3.3). It is a value wherein the false accept rate is equal to the false reject rate. Lower values are better. In the current state-of-the-art for speaker recognition, performance is less than 3.0% EER.

4.6 Further Considerations

Most speech processing technologies are limited by the quality and type of data that is available. This is especially true when it comes to neural networks and representation learning. In this chapter, two different datasets were utilized that reflected different speech phenomena: speaking style and expression of emotions. While both datasets were carefully curated for style and emotion, they were not designed for speaker recognition. In fact, both datasets are very poorly suited for speaker recognition tasks. In the IEMOCAP data, there are often two speakers heard in the recording with one being a primary and the other more distant in the background. There is no diarization information provided with the IEMOCAP dataset, and in fact the background speaker is often unintelligible. However, this does present some issues when using IEMOCAP for any aspect of speaker recognition. The lack of diarized audio was possibly a major contributing factor to the poor EER described in Section 4.5.3. Likewise the IViE dataset contained audio that was not properly diarized and which suffered similar issues as the IEMOCAP data in terms of a primary and background speaker. In both audio datasets, there was high reverberation. It is possible that the recording conditions have also interfered with speaker recognition performance. For these reasons, the IEMOCAP and IViE datasets are determined to be unsuitable for further experimentation. Continuing with these datasets in further experiments in later chapters presents a risk of modeling noise or non-diarized audio during disentanglement experiments rather than the intended features of style/emotion and speaker information. A possible solution to the noise issue would be to use off-the-shelf speech enhancement tools, however such tools may be unreliable or inadvertently propagate additional noisy artifacts. The development of tailored speech enhancement strictly for the purpose of disentanglement experiments would draw efforts away from principled disentanglement research. Instead future experiments (such as those presented in Chapter 5) would benefit from a studio-quality multi-speaker dataset such as the voice cloning toolkit (VCTK) [Yamagishi et al., 2019] for the purpose of establishing and evaluating disentanglement techniques.

4.7 Conclusion

In this chapter, experiments have explored disentanglement using traditional speaker representations: i-vectors and x-vectors. The disentanglement methods that were proposed operate on these existing representations as input. While the representations are often used in speaker recognition, experiments show that both types of representation contain information that is predictive of style and emotion. Further, an autoencoder is able to simultaneously reduce the dimensionality of the representations while separating style/emotion factors from other factors. The ability to separate style factors would be useful in many speech applications including speech-to-speech translation, speech synthesis, and speaker identification. For example, removing the style/emotion information from traditional speaker representations could result in more "pure" speaker representations. If speaker representations were more "pure" then they would exhibit a lower EER value compared to "less pure" representations. During the evaluation it was found that neither the $z1$ or $z2$ representations improved EER over the original x-vectors and i-vectors. While the experiments in this chapter did not produce a more "pure" speaker representations, it is left to future work to explore the idea of purification. Very recent attempts have been made, but this is an open research area [Peri et al., 2020a; Raj et al., 2019].

It is difficult to find data that has appropriate style/emotion labels alongside speaker labels, with an adequate quality that could be used for this type of modeling. While the data forms a limitation, another main limitation is that the autoencoder method does not result in two distinct disentangled representations. It creates a representation of 'style' and 'not style'. It is not clear how a representation of 'not style' could be useful for speech applications. Furthermore, as the method is based on externally-formed i-vectors and x-vectors, the final disentanglement is always limited to the initial conditions which created these speaker-based representations. If, for example, the i-vectors and x-vectors were improperly trained or there is a data mismatch, then any attempts to disentangle the information may be pointless.

Therefore a different approach to disentanglement will be proposed in Chapter 5 that instead operates on the speech signal directly. Such an approach would be considered end-to-end. This would overcome the issue of how i-vectors and x-vectors are extracted. It may also allow other aspects of disentanglement to be explored such as disentangling multiple types of information at the same time. An end-to-end approach may utilize autoencoders once again, though it is recommended to use an architecture that provides a much richer latent space, such as feature clustering, compared to the one proposed in this chapter. Additional motivations for exploring an end-to-end approach include at the time of experimentation in this thesis, such an approach has not yet been explored. The speech signal also contains more information than intermediary embeddings, which may be noisy or have other quality issues. Working directly with the speech signal may lead to higher quality learned representations.

Chapter 5

Methods for End-to-End Speech Disentanglement

5.1 Introduction

Many open questions remain regarding which types of informational factors from the speech signal should be represented abstractly, as well as how this is best accomplished. For example, prosody can be represented as a low-level acoustic descriptor called fundamental frequency (F0), and it can be extracted directly from the speech signal waveform using autocorrelation [Rabiner, 1977]. Prosody can also be represented abstractly in the form of "style tokens" [Wang et al., 2018] that are embeddings learned from a deep neural network (DNN). Furthermore, prosody can vary from speaker to speaker and from utterance to utterance. When speaker representations are created for speaker recognition tasks, the prosody is not accounted for. In fact, prosody is ignored entirely.

Rich representations of speaker identity, such as x-vectors [Snyder et al., 2018], i-vectors [Dehak et al., 2011], and learnable dictionary encodings (LDE) [Cai et al., 2018], are commonplace in many speech technologies ranging from speaker and language recognition, to text-to-speech (TTS) synthesis [Cooper et al., 2020], and voice conversion (VC) [Ding and Gutierrez-Osuna, 2019]. Representations of speaker identity are also known to contain unwanted information not necessarily related to speaker identity because they often encode extra information such as recording environment, speaker emotion, speaking style, and lexical content [Raj et al., 2019].

Machine learning methods can remove these unwanted informational factors. There have been recent attempts to "purify" speaker representations by removing extra information [Peri et al., 2020a]. However, the techniques discard unwanted information rather than extracting it into separate representations. Even recent attempts to explicitly create separate representations [Williams and King, 2019] have fallen short. This is due to data quality and also the limitations of using intermediate speaker representations as input rather than the raw speech signal. So far, disentanglement methods have not been able to demonstrate that any of the learned representations meet the following criteria:

1. The learned representations are sufficiently rich to model targeted speech features.
2. Representations that model targeted speech features do not also model non-target speech features (e.g., a representation of speaker identity does not also model speech content).
3. Representations have utility outside of the system that disentangles information.

Therefore, an end-to-end solution is proposed in this chapter. It aims to simultaneously disentangle phone content, prosody, and speaker identity. In the proposed approach, traditional "channel" information is not explicitly modeled. The term channel refers to elements of recording and session variability. This is an important issue when there is potentially a large mismatch between training data and real-life application [Vair et al., 2006]. In this chapter, the dataset that is used in experiments does not exhibit as much channel variability as other types of datasets traditionally used in speaker recognition [Curelaru, 2018].

To achieve disentanglement between content, prosody and speaker identity, an architecture is created that builds upon the original vector-quantized variational autoencoder (VQ-VAE) from voice conversion [van den Oord et al., 2017] that is shown in Figure 5.1. While the original, unmodified VQ-VAE can learn a type of phone representation, the system does not generalize well to unseen speakers or unseen content primarily because the speaker representations are one-hot vectors. Further, it does not learn representations of speaker characteristics or prosodic information. The method presented here learns three different types of VQ codebook at the same time (phones, speaker, and F0) while producing synthetic speech that is highly intelligible. It is also able to generalise to unseen conditions. Learning multiple VQ codebooks at the same time forms the basis of the proposed solution for achieving disentangled representations.

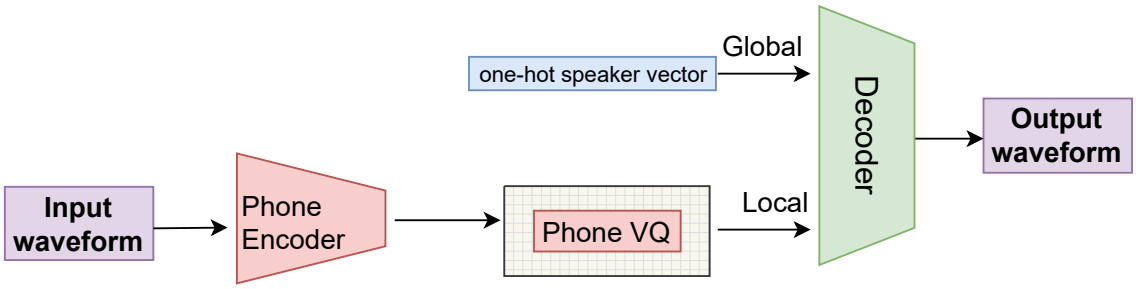


Figure 5.1: Original System [van den Oord et al., 2017]: single phone encoder/VQ, and one-hot speaker vector for global conditioning with a WaveRNN decoder. This system is referred to as **VQ-VAE**.

5.2 Related Work

In the VQ-VAE paradigm, learned VQ codebooks result in rich continuous-valued embeddings as well as their corresponding discrete codes. As embeddings, the learned representations can be used to condition a decoder, such as WaveRNN, locally or globally. It was previously shown that a large VQ space can learn to represent phones and sub-phones when trained with speech [van den Oord et al., 2017]. In the original VQ-VAE model, it was found that the number of learned VQ codebooks exceeded the unique phones in the corresponding phone set for the content of the audio. This suggests that the VQ-VAE phone codebook learned a unit smaller

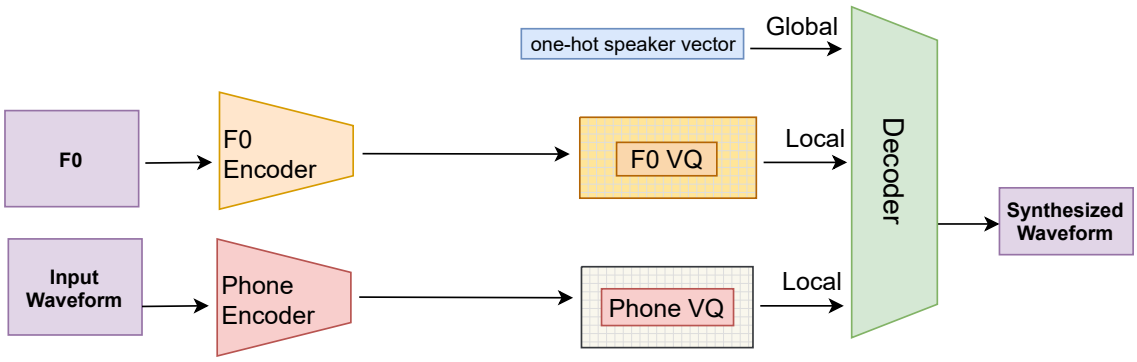


Figure 5.2: Dual-Encoder system [Zhao et al., 2020b]: phone encoder/VQ, F0 encoder/VQ as local conditions, with one-hot speaker vector for global conditions and a WaveRNN decoder. The one-hot speaker vector could be replaced with any other type of speaker vector obtained externally such as LDE or x-vectors. This system is referred to as $+$**F0**.

than a phone, and that is why they used the term "sub-phone". In this thesis, for clarity the VQ codebook that models speech content is referred to as a "phone VQ codebook" or "phone codebook", however the contents of the learned codebook is modeling a unit smaller than phones. The exact nature of that unit is beyond the scope of this chapter.

More recently, VQ-VAE has been successfully modified to learn two separate VQ spaces at the same time, F0 and phones [Zhao et al., 2020b], and can be referred to as a *dual-encoder* model since it consists of two separate encoders as shown in Figure 5.2. Incorporating an additional encoder for F0 greatly improved speech synthesis quality for Japanese and Mandarin. However, the speaker representation was a one-hot encoded speaker vector or a separately obtained LDE vector. The experiments presented in this chapter will build on this effort by learning the speaker VQ space during training and using it for global conditioning in the WaveRNN decoder.

In speech synthesis, a well-known representation is the "global style token" (GST) first proposed by [Wang et al., 2018]. GSTs are embeddings learned during TTS training and subsequently applied during inference to control speaking style output. GSTs do not require prosodic labels, and the style representations are separate from phone (or grapheme) content. However, the representations may be highly specific to TTS applications. It is not known if the representations can be useful outside of the TTS system, such as within another different TTS system or for voice conversion. Some style tokens actually capture different types of *noise* rather than speaking style because the method is unsupervised and data-driven.

Recent work from [Ebbers et al., 2020] aims to disentangle content from speaker identity for the purpose of voice conversion. Their technique might be considered unsupervised (e.g., it does not rely on explicit speaker labels), except that it does utilize some speaker-specific information. Rather than speaker labels, it uses speaker-specific vocal tract length perturbation (VTLP) in order to assist the algorithm with disentanglement. The VTLP functions as a regularization for the adversarial component. While this approach was demonstrated to be useful specifically for voice conversion, it was not shown to meet all of the criteria for disentanglement that was proposed earlier in Section 5.1. Their work only evaluated the disentanglement success in terms of a post-training classification task. The classification task showed that representations of content had lost some ability to classify speakers, and representations of speaker had lost some ability to classify content. The loss of content or speaker information was measured

by classification accuracy, and compared before and after disentanglement. Furthermore, the approach did not show that learned representations were re-usable outside of the system they were trained on.

5.3 Architecture Models

This section will provide an overview of the VQ-VAE models that will be experimented with and developed in this chapter. A more detailed description of the original VQ-VAE was provided in Chapter 2 (including descriptions of the encoder and decoder). The following sections will describe the proposed VQ-VAE variants. The main differences between each proposed system is the number of stacked encoders (and VQ codebooks) as well as the amount of supervision from the use of auxiliary classifiers. Variations of VQ-VAE are summarized in the list below, along with citations (where appropriate, if the system configuration was taken from previous work), and pointers to diagram figures. For system variants that are fully self-supervised but contain a global "speaker" component, these are referred to as *global conditions* because the self-supervised version does not use explicit speaker labels so there is no guarantee that the global conditions monitor speaker identities. Once the speaker labels are introduced for auxiliary classifiers, then the architecture can be referred to as semi-supervised. All versions of all architectures were implemented in PyTorch and were based upon the original VQ-VAE [van den Oord et al., 2017] as well as the dual-encoder model by Zhao et al. [2020b]. The bold indicates how each system variant will be referred to in the remainder of this thesis.

- Original Self-Supervised **VQ-VAE**, from van den Oord et al. [2017], Figure 5.1
- Self-Supervised VQ-VAE +**F0**, from Zhao et al. [2020b], Figure 5.2
- Self-Supervised VQ-VAE +**Global**, Figure 5.3
- Semi-Supervised VQ-VAE +**Speaker**, Figure 5.4
- Adversarial Semi-Supervised VQ-VAE +**Adversarial**, Figure 5.5
- Self-Supervised VQ-VAE +**Global**/**F0**, Figure 5.6
- Semi-Supervised VQ-VAE +**Speaker**/**F0**, Figure 5.7
- Adversarial Semi-Supervised VQ-VAE +**Adversarial**/**F0**, Figure 5.8

5.3.1 Original Self-Supervised VQ-VAE

As previously described in Chapter 2, the original **VQ-VAE** system (Figure 5.1) is fully self-supervised during training as it does not require any labeled data. The learning process is governed by three terms in the loss function which balance the encoder, decoder, and VQ codebook (Equation 5.1).

$$L = L_R + \alpha L_{VQ} + \beta L_C \tag{5.1}$$

First, the L_R term is the reconstruction loss, defined as $-\log p(x|z_q(x))$ which is the negative log likelihood of decoder output x given the output of the encoder $z(x)$ after quantization q. The second term L_{VQ} is the VQ objective, defined as $\|\text{sg}[z_e(x)] - e\|_2^2$ and it is an l_2 loss which guides

VQ embedding vectors e towards encoder output $z_e(x)$. The sg term is a stop-gradient operator which effectively creates a non-updated constant. The purpose of the VQ term is to ensure that embeddings are also guided by reconstruction loss. Finally, the L_C term is a commitment loss defined as $\|z_e(x) - \text{sg}[e]\|_2^2$ to ensure that the encoder commits to a VQ embedding vector e and constrains how the VQ space is utilized.

The size of the phone VQ codebook was set to 512 and each VQ embedding vector was set to 128-*dims*. This model is referred to now as **VQ-VAE** for the purposes of referring to it in results tables or to discuss comparisons to other variants of the architecture. The WaveRNN decoder was implemented as described in [Zhao et al., 2020b], and is capable of performing local conditioning (e.g., phone VQ embeddings) as well as global conditioning (e.g., speaker one-hot vectors). While the implementation used in this thesis was WaveRNN, other neural decoders would be sufficient such as WaveNet [van den Oord et al., 2017].

5.3.2 Self-Supervised VQ-VAE + F0 Codebook

The first dual-encoder model was developed by [Zhao et al., 2020b] and is shown in Figure 5.2. In this fully self-supervised model, an additional encoder for F0 was added to the original encoder for phones. The purpose of introducing the F0 component to VQ-VAE was because preliminary experiments showed that the original architecture did not always produce appropriate prosody in languages such as Chinese or Japanese. It was hypothesized that a successful VQ-VAE architecture would need to simultaneously extract representations corresponding to the segmental features (such as phones), as well as representations corresponding to the supra-segmental features (such as F0).

In this new architecture there were two inputs: waveforms and F0. The input waveform representation was linear PCM. The output waveform was parameterized for coarse and fine parts separately using scripts provided by [Zhao et al., 2020b]. The input to the F0 encoder was the F0 as extracted from the speech signal using the software REAPER[1], though any decent F0 tracking software would be sufficient. The original VQ-VAE models used WaveRNN for the decoder, subsequent implementations utilized WaveRNN[2] [Zhao et al., 2020b], and all of the implementations in this chapter also utilized WaveRNN for the decoder. The F0 and phone VQ vectors were utilized as local conditions to the decoder. These two inputs had differing sample rates. To deal with the different sampling rates, and to align the tensors for local conditioning, the F0 tensor was upsampled to match the phone tensor. Speaker identity was controlled with a one-hot speaker vector. The loss function is very similar to that of the original system, however additional loss components were added which correspond to the F0 encoder and F0 VQ clustering (Equation 5.2).

$$L = L_R + \alpha(L_{VQp} + L_{VQf}) + \beta(L_{Cp} + L_{Cf}) \tag{5.2}$$

This model is referred to now as **+F0**. As with the original **VQ-VAE** system, the terms of the loss function are the same except the phone and F0 components are represented by separate terms, since they are learned separately. Therefore, the L_{VQp} and L_{VQf} terms represent two separate VQ objectives, as previously defined. Likewise, the L_{Cp} and L_{Cf} terms represent

[1] `https://github.com/google/REAPER`
[2] `https://github.com/mkotha/WaveRNN`

separate encoder commitment losses, as previously defined. The additional losses were balanced using weighting. The components learning phone features (L_{VQp}, L_{Cp}) and the components learning F0 features (L_{VQf}, L_{Cf}) were weighted (α and β, respectively), while the reconstruction loss remained unweighted. While these weights are highly tunable, they were utilized for the purpose of ensuring that each of the losses were approximately of the same magnitudes [Zhao et al., 2020b]. This balancing of the weights helped to ensure that no particular component was over-trained. In this system the additional components was a F0 VQ codebook of size 10, with each VQ embedding size set to 128-*dims*.

5.3.3 Self-Supervised VQ-VAE + Global Conditioning

Building upon the open-source implementation provided by [Zhao et al., 2020b], a different kind of encoder and VQ codebook component was added to the original VQ-VAE architecture (Figure 5.3). This addition was a speaker-based component to model global conditions, and it replaced all of the F0 components that were being used for local conditioning. To transform the encoder output into global conditions, a temporal average pooling layer (TAP) and two feed-forward layers (FF) were added along with two feed-forward layers between the encoder and VQ codebook. The dual-encoder VQ-VAE is also self-supervised, and the loss function was adjusted to account for the additional global terms (Equation 5.3).

$$L = L_R + \alpha(L_{VQp} + L_{VQs}) + \beta(L_{Cp} + L_{Cs}) \tag{5.3}$$

This model is referred to now as +**Global**. The L_{VQp} and L_{VQs} terms again represent two separate VQ objectives as previously defined (one for the phone VQ and another for the speaker VQ). Likewise, the L_{Cp} and L_{Cs} terms represent separate encoder commitment losses, as previously defined. Similar to the dual model defined earlier, this model also used a weighted linear combination of losses. The local features (L_{VQp}, L_{Cp}) corresponding to the phones and the global features (L_{VQs}, L_{Cs}) corresponding to the speaker were weighted (α and β, respectively), while the reconstruction loss remained unweighted. In this system the additional components were a global VQ codebook of size 256, with each VQ embedding size set to 128-*dims*.

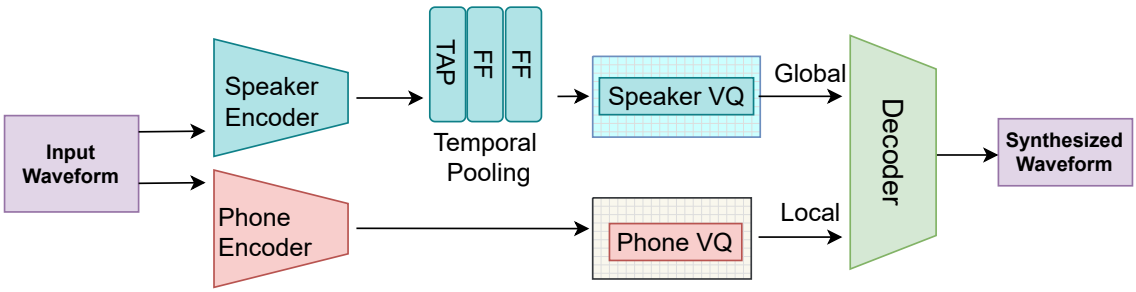


Figure 5.3: Self-Supervised dual-encoder system (proposed): speaker encoder with phone encoder. The global conditions are produced with a temporal average pooling layer (TAP) with two feed-forward layers (FF). This system is referred to as +**Global**.

5.3.4 Semi-Supervised VQ-VAE + Speaker Codebook

The semi-supervised variation described here was created because it was found that the fully self-supervised version was difficult to train, often became unstable, failed to output intelligible speech, and was unable to utilize the VQ codebook space in a meaningful way. For the VQ codebook usage, upon examining changes in the VQ codebook before and after training, it was found that only one single codebook entry was being used for all of the speakers, instead of multiple different codebooks. This means that the speaker VQ codebook was not learning information about speakers for the self-supervised model. Therefore this semi-supervised architecture was developed (Figure 5.4), with slight differences from the self-supervised one described in 5.3.3. Here, speaker labels were provided explicitly as additional information in order to *lightly* supervise the speaker encoder. There were two semi-supervised variants of the dual-encoder model. First, an auxiliary speaker classifier was added to the global encoder, using speaker labels as ground truth. Two different loss functions were explored and compared for the speaker classifier, referred to now as **+Speaker, Softmax** for the softmax loss, and **+Speaker, A-Softmax** for the angular softmax loss. Previous work has shown that angular softmax (first proposed for face recognition [Liu et al., 2017]) can achieve significant improvements in speaker recognition modeling because it places constraints on within-speaker variance for learned embeddings. This means it can potentially cause learned speaker embeddings to be more robust to variations, such as recording session. In each case, the classifier loss (Aux_s) was added, and this is expressed in Equation 5.4.

$$L = L_R + \alpha(L_{VQp} + L_{VQs}) + \beta(L_{Cp} + L_{Cs}) + \eta Aux_s \tag{5.4}$$

The L_{VQp} and L_{VQs} terms again represent two separate VQ objectives as previously defined (one for the phone VQ and another for the speaker VQ). Likewise, the L_{Cp} and L_{Cs} terms represent separate encoder commitment losses, as previously defined. The Aux_s term is the loss of the speaker classifier, defined as the softmax or angular-softmax for speaker labels.

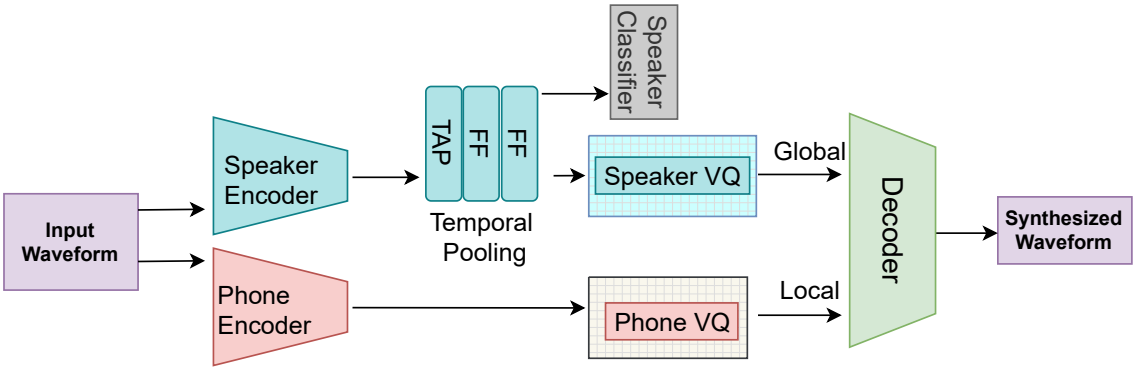


Figure 5.4: Semi-Supervised dual-encoder system (proposed): speaker encoder with phone encoder. The global conditions are produced with a temporal average pooling layer (TAP), two feed-forward layers (FF), alongside an auxiliary speaker classifier. This system is referred to as **+Speaker**.

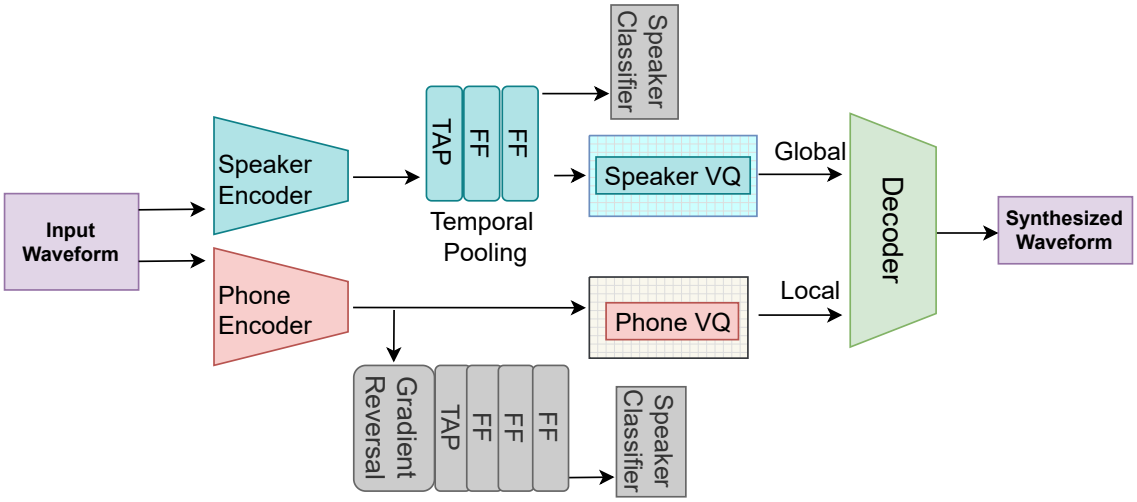


Figure 5.5: Adversarial Semi-Supervised dual-encoder system (proposed): speaker encoder with phone encoder. The global conditions are produced with a temporal average pooling layer (TAP), two feed-forward layers (FF), alongside an auxiliary speaker classifier. An additional adversarial classifier is included. This system is referred to as +**Adversarial**.

An additional version was developed, which included an additional auxiliary adversarial classifier added to the phone encoder (Figure 5.5). The adversarial component consisted of a gradient reversal layer [Ganin et al., 2016], feed-forward layers, and a speaker classifier. The purpose of this adversarial component was to encourage disentanglement of phone and speaker information by nudging the phone encoder to ignore speaker-related global information. The adversarial model therefore includes two auxiliary losses (Aux_s and Aux_{adv}) as described by Equation 5.5.

$$L = L_R + \alpha(L_{VQp} + L_{VQs}) + \beta(L_{Cp} + L_{Cs}) + \eta Aux_s + \theta Aux_{adv} \tag{5.5}$$

The L_{VQp} and L_{VQs} terms again represent two separate VQ objectives as previously defined (one for the phone VQ and another for the speaker VQ). Likewise, the L_{Cp} and L_{Cs} terms represent separate encoder commitment losses, as previously defined. The Aux_s term is the loss of the speaker classifier, defined as the softmax or angular-softmax for speaker labels. The Aux_{adv} term represents the softmax loss for the adversarial classifier. The speaker classifier for the adversarial component used softmax loss. However, as before two different types of losses were explored for the non-adversarial speaker classifier: the softmax loss (+**Adversarial, Softmax**) and the angular softmax loss (+**Adversarial, A-Softmax**). The terms of Equation 5.4 and Equation 5.5 both include weights on the losses (α, β, η, and θ) that balance the magnitudes of the loss functions to help ensure that no component is over-contributing to the training of the system as a whole.

5.3.5 Self-Supervised VQ-VAE + Global Conditions + F0 Codebook

So far, two types of models have been described: the original **VQ-VAE** that had one single encoder and VQ codebook, and a set of dual-encoder variations that introduced an additional encoder and VQ codebook. In this section, the triple-encoder variation is introduced. The idea behind using a triple-encoder version of VQ-VAE was to exploit the generalization capabilities that a learnable speaker codebook provides, along with better prosody from using an explicit F0 representation. Therefore the triple-encoder variation of VQ-VAE simultaneously learns representations for phones, speaker, and F0 (Figure 5.6). The input to this system variant remained similar to the other variants. For example, the F0 was extracted using REAPER as before. In this case, both F0 and phone representations were treated as local conditions for WaveRNN. For this model, there were only one auxiliary classifier, which sought to classify the F0. No other auxiliary classifiers were utilized. The overall system loss is expressed in Equation 5.6.

$$L = L_R + \alpha(L_{VQp} + L_{VQs} + L_{VQf}) + \beta(L_{Cp} + L_{Cs} + L_{Cf}) \tag{5.6}$$

It incorporates losses for the phone encoder (L_{Cp}) and phone VQ (L_{VQp}), speaker encoder (L_{Cs}) and speaker VQ (L_{VQs}), and the F0 encoder (L_{Cf}) and F0 VQ (L_{VQf}). As with system variants described earlier, the weights (α, β) were used to balance the order of magnitude of the losses. This system is known as +**Global/F0**.

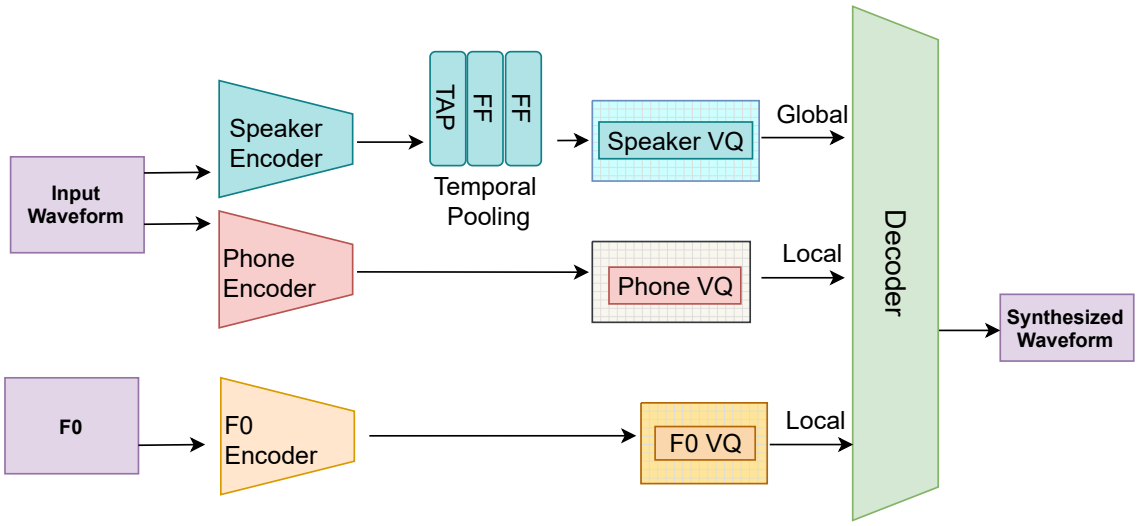


Figure 5.6: Self-Supervised triple-encoder system (proposed): speaker encoder, phone encoder, and F0 encoder. The global conditions are produced with a temporal average pooling layer (TAP) and two feed-forward layers (FF). This system is referred to as +**Global/F0**.

5.3.6 Semi-Supervised VQ-VAE + Speaker Codebook + F0 Codebook

The final system variant proposed in this chapter combines semi-supervised learning from auxiliary classifiers with the triple-encoder approach. It is shown in Figure 5.7. In this case, the loss function is a linear combination of weighted losses for the phone encoder and VQ ($L_{VQp}+L_{Cp}$), the speaker encoder and VQ ($L_{VQs}+L_{Cs}$), the F0 encoder and VQ ($L_{VQf}+L_{Cf}$), and auxiliary losses. This system is described by Equation 5.7 and Equation 5.8, and is known as +**Speaker/F0**.

$$L = L_R + \alpha(L_{VQp} + L_{VQs} + L_{VQf}) + \beta(L_{Cp} + L_{Cs} + L_{Cf}) + Aux_w \tag{5.7}$$

$$Aux_w = \eta Aux_s + \mu Aux_f \tag{5.8}$$

The auxiliary loss for the speaker (Aux_s) is the same as previously described in other systems. A new auxiliary F0 classifier (Aux_f) was added. This classifier is based on quantized F0, where the range of the F0 has been pre-processed into one of 10 quantized frequency regions, as a form of stylization, similar to work that utilized F0 in a VQ-VAE context [Wang et al., 2019b]. This allowed for the creation of an F0 classifier which aimed to classify the F0 encoder output into one of the 10 quantized regions. As before, the weights (α, β, η, μ) were used to balance the order of magnitude of the losses.

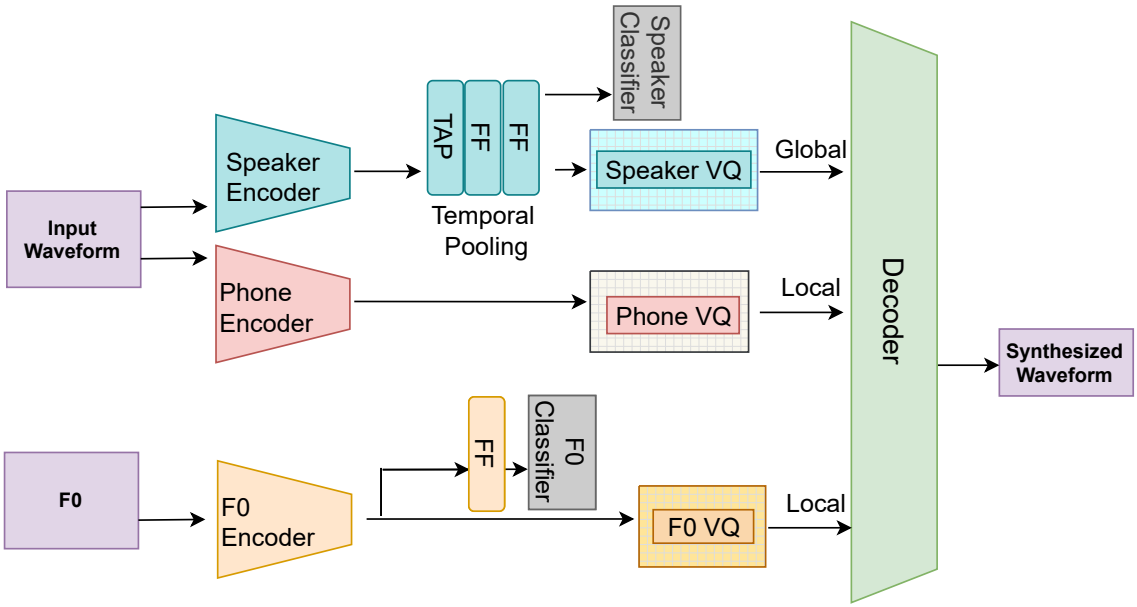


Figure 5.7: Semi-Supervised triple-encoder system (proposed): speaker encoder, phone encoder, and F0 encoder. The global conditions are produced with a temporal average pooling layer (TAP). An auxiliary speaker classifier and auxiliary F0 classifier are included. This system is referred to as +**Speaker/F0**.

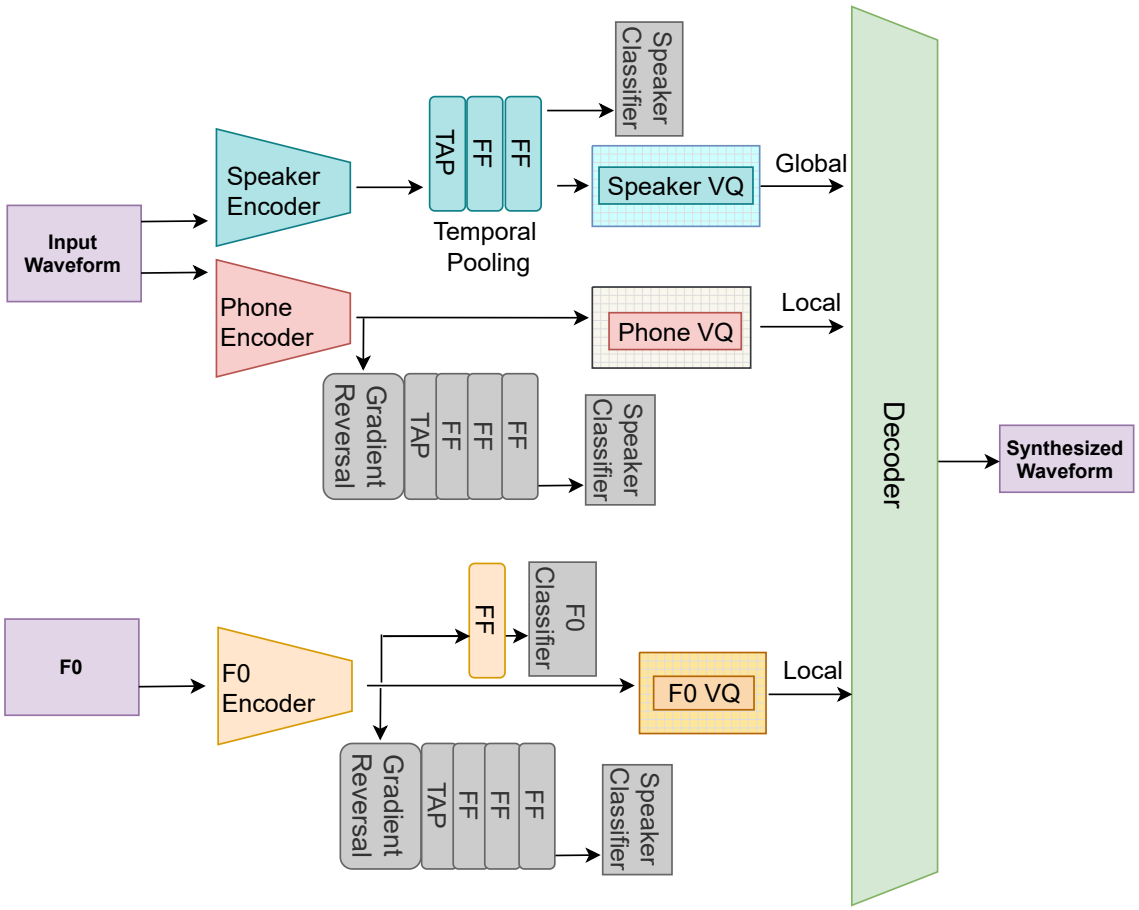


Figure 5.8: Adversarial semi-Supervised triple-encoder system (proposed): speaker encoder, phone encoder, and F0 encoder. The global conditions are produced with a temporal average pooling layer (TAP). An auxiliary speaker classifier and auxiliary F0 classifier are included. Two additional adversarial classifiers are included. This system is referred to as +**Adversarial/F0**.

Another variant of this system (Figure 5.8) included two additional adversarial losses: one monitoring output from the phone encoder (Aux_{advP}) and the other monitoring output from the F0 encoder (Aux_{advF}). These two adversarial classifiers are included to assist the two encoders to ignore speaker-related information and to attempt to de-correlate encoder output and speaker labels. This system is known as +**Adversarial/F0**. It is described by Equation 5.9 and Equation 5.10

$$L = L_R + \alpha(L_{VQp} + L_{VQs} + L_{VQf}) + \beta(L_{Cp} + L_{Cs} + L_{Cf}) + Aux_w + Aux_{adv} \tag{5.9}$$

$$Aux_{adv} = \theta Aux_{advP} + \lambda Aux_{advF} \tag{5.10}$$

As before, the speaker classifier could use either softmax or angular-softmax, and these were again compared as additional variants. Similar to the earlier descriptions, the weights on the losses for this system (α, β, η, μ, θ, λ) were used to balance the magnitudes of the losses.

5.4 Dataset Description

The dataset consisted of English audio of 110 unique speakers from the Voice Cloning Toolkit (VCTK) corpus [Yamagishi et al., 2019]. The data was prepared following the steps from scripts provided by [Zhao et al., 2020b], including quantization and normalization. All of the audio was downsampled to 16 kHz and normalized using ITU-T Rec. G.191 sv56[3]. The leading and trailing silence was trimmed using voice activity detection (VAD) scripts from the DIHARD II 2019 challenge[4]. F0 was extracted using REAPER with a frame-shift of 5ms.

The training set consisted of 100 speakers, and a held-out test set consisted of 10 speakers. Further, four testing conditions were compiled, each with 10 speakers and 8 utterances per speaker:

- C1 seen speakers / seen content
- C2 seen speakers / unseen content
- C3 unseen speakers / seen content
- C4 unseen speakers / unseen content

These four testing conditions vary in whether or not the content or the speaker has been seen during training. The C1 condition should result in the best quality of speech because it was seen during training. The remaining conditions help to gauge how well the models generalize to unseen conditions. The C4 condition should be more challenging because it represents full held-out conditions in terms of the content as well as the speaker.

5.5 Training Strategy

All of the VQ-VAE variant systems were trained using the initialization and configuration from [Zhao et al., 2020b]. However, for the dual-encoder and triple-encoder systems, a warm-up model was used to reduce overall training time and to avoid over-training certain components while inadvertently leaving other components under-trained. The warm-up model for dual-encoder systems consisted of a fully-trained original **VQ-VAE** model trained to 800k steps. This provided a trained phone encoder and phone VQ, and allowed the subsequent training to focus on learning F0 or global speaker representations. For triple-encoder systems, the warm-up model consisted of a fully trained version of **+Adversarial, softmax** that was trained to 550k steps. That particular warm-up model was selected as the best-performing dual-encoder model. Layers and weights that were not part of a triple-encoder design (such as adversarial auxiliary classifiers) were discarded. Therefore it can be seen that development of each variation of the VQ-VAE models was somewhat dependent on training simpler models, before training more complex models such as the triple-encoder variations. All dual-encoder and triple-encoder systems were trained for an additional 800k steps, and the best model was selected based on lowest overall validation loss.

[3] `https://github.com/foss-for-synopsys-dwc-arc-processors/G722`
[4] `https://github.com/iiscleap/DIHARD-2019-baseline`

5.6 Estimated Quality of Synthetic Speech

This section reports a suite of automatic methods that were used to estimate the quality of the synthetic speech that was generated by each of the system variants. The purpose of using automatic methods to assess synthetic speech is because to speed up the development process for scientists and well as help to save costs from listening tests. Automatic methods to assess synthetic speech are rapidly growing in popularity [Williams et al., 2020; Cooper and Yamagishi, 2021]. While it is true there is no replacement for subjective naturalness, as judged by humans, certain automatic evaluation methods can estimate synthetic speech quality. This is especially true for identifying larger trends during the development process. None of these metrics alone is perfect and none of them should be used to replace human judgements. This section presents and defines 6 such metrics, which can help provide a general sense of the model performance especially when all of the metrics are considered together. Results from these metrics presented in Table 5.1 and further described in this section. The values in Table 5.1 are averaged across all four testing conditions to gauge larger-scale trends such as best-performing models overall. However, it is also important to evaluate how well these models can generalize to unseen conditions. For finer granularity of results across seen and unseen conditions, bar plots are provided alongside the detailed description of each metric in 5.6.1 – 5.6.5.

Table 5.1: Speech synthesis quality estimation for four testing conditions on VCTK data. (S: softmax, AS: angular-softmax). Reported values are averaged across all four testing conditions. Two systems were taken from previous work: original **VQ-VAE** [van den Oord et al., 2017] and +**F0** [Zhao et al., 2020b]. All other systems were newly proposed in this chapter. MOS1, MOS2, and P.563 estimate speech naturalness. MOS1 and MOS2 systems were developed earlier in Chapter 3. Speaker similarity compares synthetic to natural speech on a per-speaker basis. WER is calculated using ASR. F0 root mean square error (RMSE) measures the difference of the log(F0) between synthetic and natural speech.

	Method		Est. MOS1	Est. MOS2	P.563	Speaker Similarity	WER (%)	log(F0) RMSE
	Natural Speech	–	3.6	2.8	2.7	1.0	9.1	0.0
	VQ-VAE	–	3.3	2.7	2.4	0.7	65.7	0.5
Dual-Enc.	+ Global	–	2.2	2.9	2.3	0.5	82.1	0.8
	+ Speaker	S	3.8	2.9	2.5	0.9	31.1	0.3
		AS	3.7	2.9	2.6	0.9	33.1	0.2
	+ Adversarial	S	3.9	3.0	2.4	0.9	27.6	0.2
		AS	3.8	2.9	2.5	0.8	33.8	0.3
	+ F0	–	3.5	2.9	2.4	0.5	39.3	0.4
Triple-Enc.	+ Global/F0	–	3.9	3.0	2.4	0.8	38.4	0.3
	+ Speaker/F0	S	3.8	2.9	2.6	0.8	33.0	0.3
		AS	3.1	2.8	2.6	0.8	45.1	0.4
	+ Adversarial/F0	S	3.7	2.8	2.6	0.8	44.6	0.3
		AS	3.4	2.9	2.4	0.7	46.2	0.4

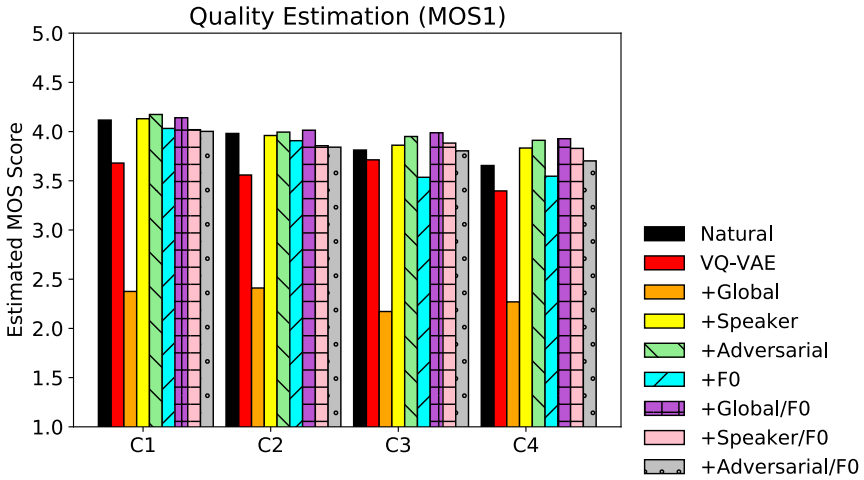


Figure 5.9: MOS1 values for selected system variants. Four testing conditions were examined based on which content or speakers were seen during training: C1 is seen speaker/seen content, C2 is seen speakers / unseen content, C3 is unseen speakers / seen content, and C4 is unseen speaker / unseen content.

5.6.1 Estimated Naturalness (MOS)

Recent efforts to learn to predict mean-opinion scores for speech have resulted in new publicly available tools[5]. In this section, the MOS estimation tool developed earlier in Chapter 3.3 was used. Two different versions of MOS are reported in Table 5.1. Each version is based on a different MOS estimation model trained on different speech representations. The first is MOS1 uses frame-level representations and was trained on TTS and VC speech from the 2019 Logical Access (LA) dataset from the ASVspoof Challenge. The second is MOS2 which extracts a special x-vector embedding that models device quality and was developed with labeled data from the ASVspoof 2019 Physical Access (PA) challenge. Both of these pre-trained models are described and compared in Williams et al. [2020].

The MOS results are shown in Table 5.1. One of the shortcomings of this tool is that there is no way to anchor to natural speech to ensure that natural speech achieves the highest MOS score. It is therefore possible for synthetic speech to be ranked higher than natural speech. This can be seen in the results where natural speech is rated as $MOS1 = 3.6$ and $MOS2 = 2.8$, while synthetic speech from the proposed systems "outperforms" natural speech on both metrics. In addition, this tool has been reported as sometimes having too little variability among scores - meaning that scores tend to cluster in a narrow middle range. The slightly lower MOS scores for natural speech highlights a shortcoming of using fully-automated methods for speech quality evaluation. Natural speech and some proposed methods appear to be very close in quality. The semi-supervised systems achieved higher estimated MOS scores compared with the original **VQ-VAE**. In general, the dual-encoder systems perform comparably to triple-encoder systems for MOS1 and MOS2. Even with the shortcomings of the MOS1 scoring, this metric seems to pick out **+Global** as being particularly lower quality than other systems.

For additional analysis across the four testing conditions, the MOS1 is shown for selected

[5] `https://github.com/rhoposit/MOS_Estimation2`

systems in Figure 5.9, reflecting a 5-point Likert scale. Angular-softmax did not provide significant gains, so these variants were omitted from the figure. The dual-encoder and triple-encoder systems are comparable across conditions, however the dual-encoder models tend to perform better. The fully self-supervised +**Global** system performs poorly across all conditions. In general, the unseen conditions are more difficult. There is a slight downward trend.

5.6.2 Estimated Naturalness (P.563)

Another tool used to automatically estimate speech naturalness is based on a publicly available standard tool called ITU-P.563 [6]. This tool estimates speech naturalness and it was designed for narrowband telephony speech. Although narrowband telephony is a mismatch to the synthetic speech that was generated in these experiments, this metric was reported in related work, so it is included here in an effort to be complete and make a comparison [Zhao et al., 2020b]. The results in Table 5.1 indicate that the range of scores are so close that this metric does not distinguish between systems very well. For the purposes of selecting a best-performing system, this metric could be ignored.

5.6.3 Speaker Similarity

As the model variants are all based on multi-speaker data, it is important that synthetic speech reflects the true and intended speaker identity. This is important especially because the proposed models construct a latent representation of speaker. Speaker similarity could be approximated in human listening tests using an A/B/x paradigm where listeners are asked to select between samples a and b which match a "reference speaker" x. In this chapter, such a large listening test may become very costly because there of the quantity of proposed systems to compare. Therefore automated estimates of speaker similarity were used to get a general sense of which systems perform poorly for speaker similarity on average.

Speaker similarity was estimated by calculating the cosine distance between x-vectors taken from natural speech (as a reference) and synthetic speech. First, utterance-level x-vectors were extracted using a pre-trained x-vector model, which had been trained on VoxCeleb data [7]. Natural and synthetic speech were compared by calculating the cosine similarity on a per-utterance basis, and averaging across utterances. The cosine similarity is useful in this context because it is a value in the range of $sim = [0, 1]$ where a value of $sim = 1.0$ designates that there is 100% similarity and a value of $sim = 0.0$ indicates that there is no similarity. This metric indicates that several systems indicate had poor speaker similarity on average, specifically the fully unsupervised +**Global** and +**F0**.

[6] `https://github.com/qin/p.563`
[7] `https://kaldi-asr.org/models/m7`

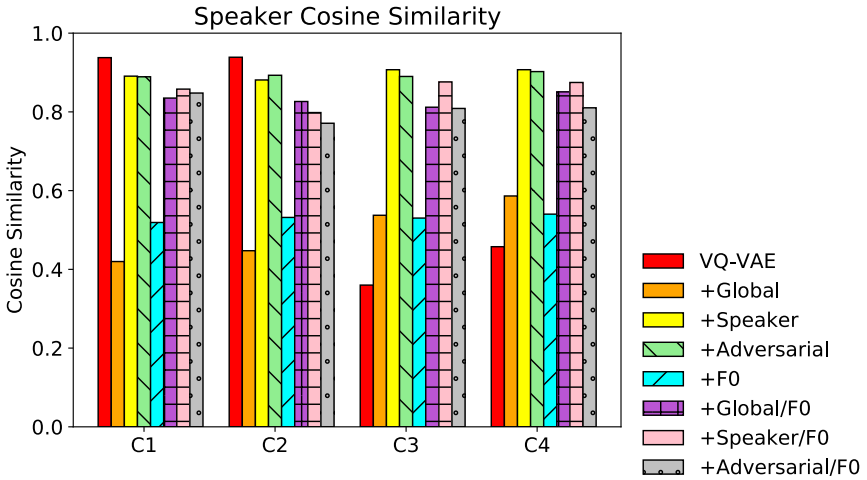


Figure 5.10: Speaker similarity for selected system variants, showing performance across four different testing conditions. Higher similarity values indicate better performance. Four testing conditions were examined based on which content or speakers were seen during training: C1 is seen speaker/seen content, C2 is seen speakers / unseen content, C3 is unseen speakers / seen content, and C4 is unseen speaker / unseen content. Note: natural speech was omitted because the similarity compared speakers between synthetic and natural speech.

Unlike the naturalness estimations in Figure 5.9, the speaker similarity in Figure 5.10 has more variation across the different seen and unseen testing conditions. First, the original **VQ-VAE** does not generalize well to the C3 and C4 unseen conditions, likely due to the limitations of using a one-hot speaker encoding instead of a more robust embedding. Similarly, the self-supervised +**Global** and +**F0** systems did not generalize well to unseen conditions. Generally, the semi-supervised methods for dual-encoders and triple-encoders performed well across all four conditions. The semi-supervised dual-encoder +**Speaker** had the most consistent and highest similarity across conditions. The triple-encoder systems are more complex than the dual-encoder systems in terms of the number of combined loss functions as well as the number of VQ spaces being learned simultaneously. Given that the triple-encoder systems are more complex than the dual-encoder systems, it is interesting to see that the speaker similarity measures for these two categories of system is very close.

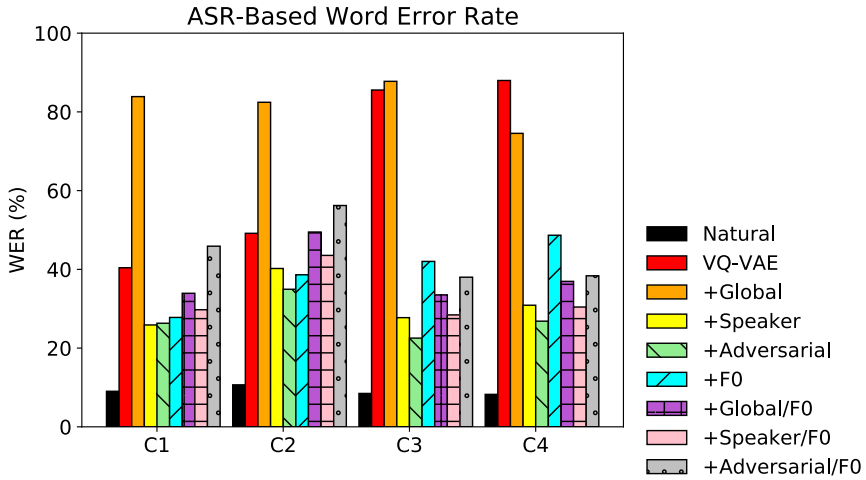


Figure 5.11: ASR-based word error rate (WER) for selected system variants. Four testing conditions were examined based on which content or speakers were seen during training: C1 is seen speaker/seen content, C2 is seen speakers / unseen content, C3 is unseen speakers / seen content, and C4 is unseen speaker / unseen content.

5.6.4 ASR-Based Word Error Rate (WER)

Intelligibility is a useful metric because it is possible for speech to have features of naturalness but if the content is not intelligible then the speech is of limited use. Human intelligibility tests are extremely expensive and can be difficult to implement due to factors such as spelling variation or typos. Therefore automatic methods to estimate intelligibility are attractive because they can save time and costs.

In this section, the intelligibility of the synthetic and natural speech was estimated by measuring word error rate (WER) from transcripts obtained through ASR [Morris et al., 2004]. As the WER is measuring an error rate, lower scores indicate better intelligibility. The Watson Speech Recognition API[8] was used for this purpose. Natural speech achieved the best overall WER and this is expected because the VCTK natural speech was recorded at studio quality.

The self-supervised system (+**Global**) was unable to produce intelligible output on average, despite having been rated similarity to other systems for naturalness and speaker similarity. The semi-supervised dual-encoder systems maintained the lowest WER. The WER reported for triple-encoder systems indicated some loss of intelligibility except for +**Speaker/F0, Softmax**, which tends to also have good performance across other reported metrics.

The intelligibility across four testing conditions is reported in Figure 5.11. As this is an error-based metric, lower values indicate better performance. The output of the original **VQ-VAE** system becomes very unintelligible for unseen conditions, whereas all of the proposed systems are more stable. All three of the proposed triple-encoder systems exhibit an improvement for intelligibility in unseen conditions.

[8] `https://www.ibm.com/cloud/watson-speech-to-text`

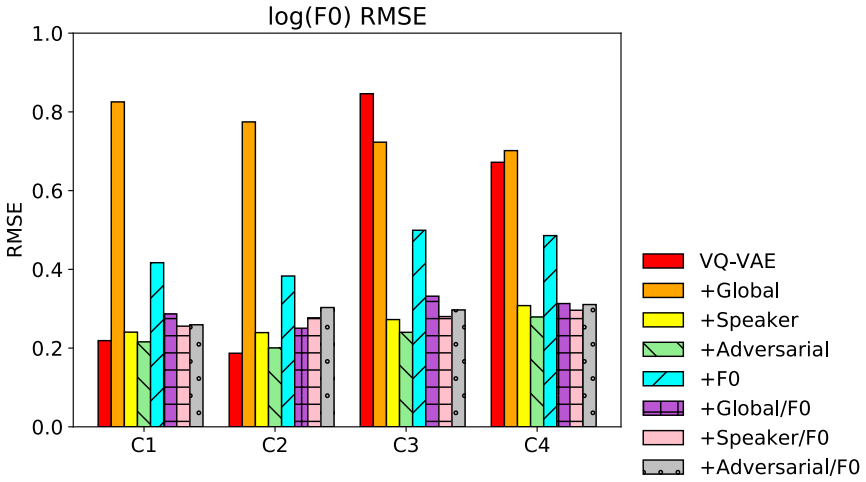


Figure 5.12: log(F0) root mean square error (RMSE) for selected system variants. Four testing conditions were examined based on which content or speakers were seen during training: C1 is seen speaker/seen content, C2 is seen speakers / unseen content, C3 is unseen speakers / seen content, and C4 is unseen speaker / unseen content.

5.6.5 F0 Error

Previous work [Zhao et al., 2020b] had found that the original **VQ-VAE** system did not always produce appropriate prosody in Chinese and Japanese speech. In fact, that was the original motivation for creating an F0 encoder and VQ component. To assess the F0 in synthetic speech, first the F0 was extracted from the synthetic and natural speech using REAPER, with the same settings as earlier described in the data preparation. The difference in F0 was calculated between natural and synthetic speech for on a per-utterance basis. The error was calculated as root mean square error (RMSE) where lower error indicates better performance. The results in Table 5.1 reflect that the fully self-supervised dual-encoder system +**Global** consistently achieved the worst performance. In the +**F0** system, it was found that performance was mostly comparable to the original **VQ-VAE** system. While this was unexpected, it could be due to the fact that English is not a tonal language, so the additional F0 modeling components did not significantly help the F0 reconstruction for English. However, the triple-encoder models with softmax were able to improve F0 reconstruction across both metrics, compared to the original **VQ-VAE**.

The F0 error is reported across testing conditions in Figure 5.12. Slightly different to the results for Chinese and Japanese reported in Zhao et al. [2020b], the addition of an F0 encoder and VQ space did not improve F0 compared to the original **VQ-VAE** system, except for unseen conditions (C3 and C4). This finding was not revealed earlier in Table 5.1 because the values in the table were averaged across all four conditions. While the triple-encoder systems explicitly model F0 as well as speaker in the learned VQ spaces, the F0 error is better in the dual-encoder systems, which do not attempt to model F0. This could be due to the fact that triple-encoders are more challenging to train, and the training strategy used in this chapter was potentially not optimal.

5.6.6 Discussion

The purpose of using automatic methods to estimate quality was ultimately to speed up the development cycle as well as save costs from human listening tests - leaving only the most interesting or important system variants for further testing and experimentation. While results from human listening tests would be ideal, the tools that were used to estimate speech quality in this chapter have saved costs as well as identified some of the best overall systems. Overall, significant improvements were not found from using angular-softmax. Across most metrics, the dual-encoder **Adversarial** system performed best, as well as demonstrated better ability to generalize to unseen conditions. The overall best-performing triple-encoder system was the +**Speaker**/**F0**, although it did not always perform as well as the **Adversarial** dual-encoder system on the reported metrics. One benefit of the triple-encoder +**Speaker**/**F0** system was that it provided an additional F0 VQ codebook, whereas the dual-encoder +**Adversarial** system did not.

5.7 Human Listening Test Evaluation

A MUSHRA test was performed to compare the quality of reconstructed speech across system variants. The results of the MUSHRA test are shown in Figure 5.13 to Figure 5.16. Each figure corresponds to one of the four testing conditions which reflect whether speakers or content were seen or unseen during training. The results for condition1 reflect that system +**F0** performs the best, however this performance drops when unseen conditions are introduced as in condition2, condition3, and condition4. That is partly because the +**F0** system relies on one-hot speaker encodings so it cannot generalize to unseen speakers. The overall best-performing system was +**Adversarial** across all conditions. The performance of +**Adversarial** on this MUSHRA test correlates with the findings from using automatic assessment tools earlier in this chapter (Figures 5.9–5.12). Figure 5.9 shows that +**Adversarial** has higher quality estimation compared to other systems. Figure 5.10 shows that +**Adversarial** generalizes well to unseen speakers better than other systems. Figure 5.11 shows that +**Adversarial** has the lowest word error rate of all system variants across all four testing conditions. And Figure 5.12 shows that +**Adversarial** has the lowest F0 reconstruction error across unseen conditions and is similar to the original VQ-VAE in the seen conditions. While the triple-encoder systems did not perform as well as the dual-encoder systems in general, there was one triple-encoder system that consistently performed better than other triple-encoder systems and that was +**Speaker**/**F0**. While the overall naturalness is lower in the +**Speaker**/**F0** system compared to +**Adversarial**, a third VQ codebook is gained without losing naturalness especially for unseen conditions. Further analysis of codebook content in Chapter 6 will explore this gain and whether the information in the codebooks is meaningful.

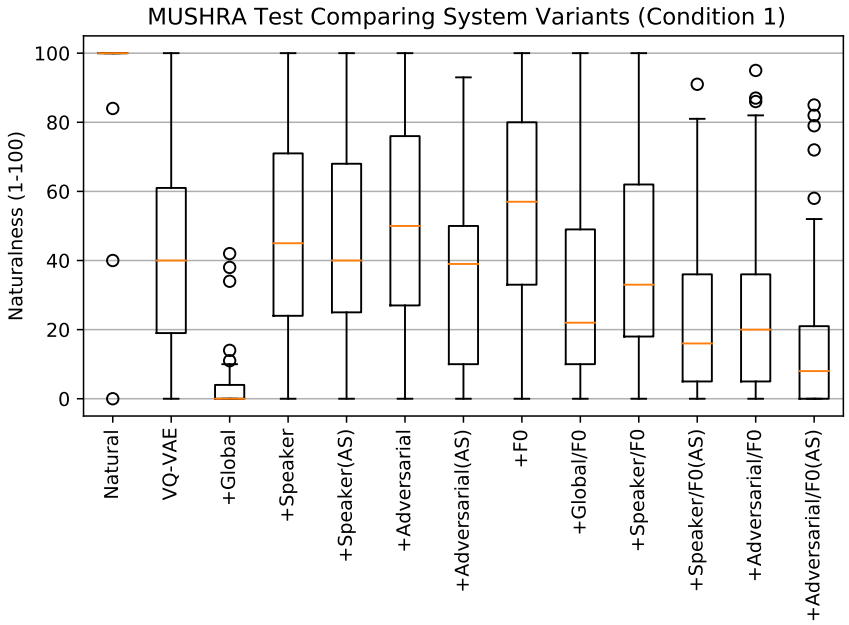


Figure 5.13: MUSHRA listening test results for 12 VQ-VAE system variants and natural speech for testing condition C1. The boxes reflect the lower and upper quartile values along with a line at the median, and open circles reflect outliers.

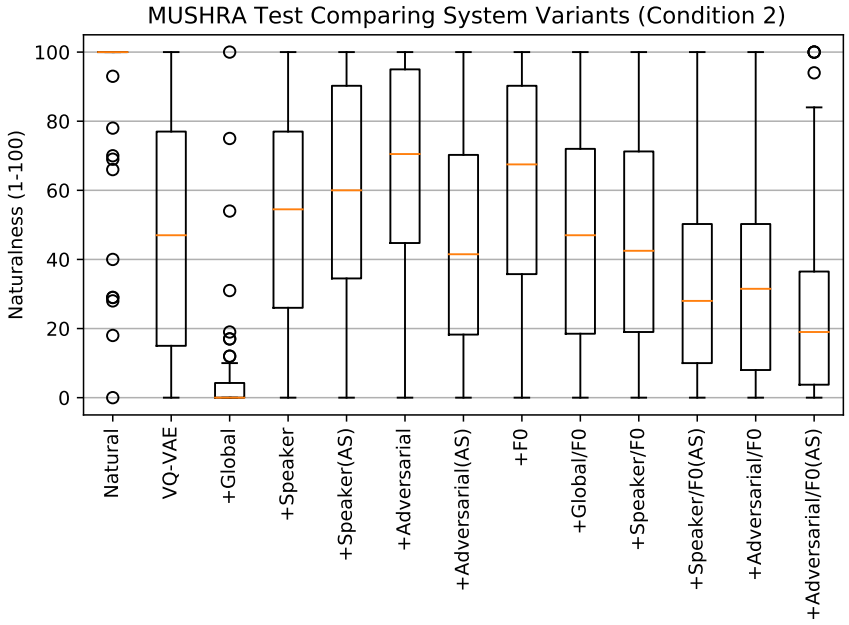


Figure 5.14: MUSHRA listening test results for 12 VQ-VAE system variants and natural speech for testing condition C2. The boxes reflect the lower and upper quartile values along with a line at the median, and open circles reflect outliers.

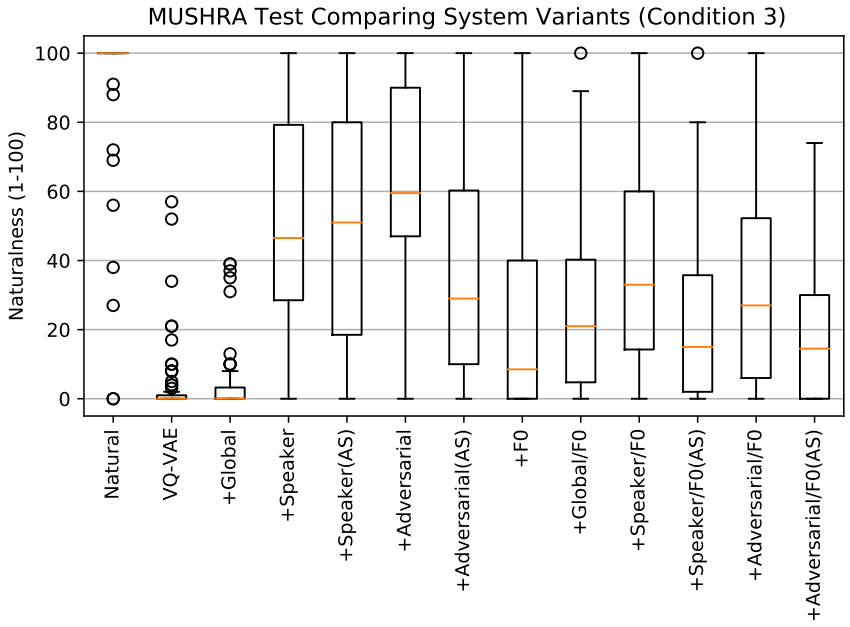


Figure 5.15: MUSHRA listening test results for 12 VQ-VAE system variants and natural speech for testing condition C3. The boxes reflect the lower and upper quartile values along with a line at the median, and open circles reflect outliers.

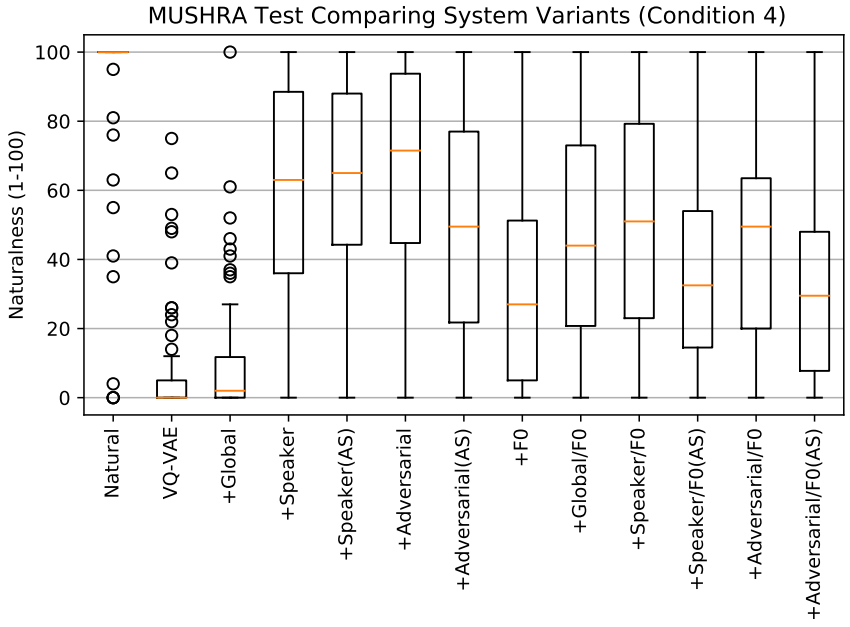


Figure 5.16: MUSHRA listening test results for 12 VQ-VAE system variants and natural speech for testing condition C4. The boxes reflect the lower and upper quartile values along with a line at the median, and open circles reflect outliers.

5.8 Learning Multilingual VQ-VAE

So far this chapter has explored disentanglement using monolingual data from the English VCTK corpus. Several variants of VQ-VAE were developed that learn multiple types of VQ codebooks at the same time. The resulting synthetic speech that was generated by the VQ-VAE variants was also assessed using automatic tools to estimate the quality. Most languages have similar sounds in common [Blasi et al., 2016] so follow-on experiments were designed in this section to find out if similar languages could be exploited in order to learn VQ codebooks that are shared across several languages. The approach taken in this section was to adapt a trained monolingual model (introduced in Section 5.3.4) to work on multilingual data. The ability to generate synthetic speech using a single multi-speaker multilingual model could have far-reaching consequences for advancing speech synthesis. For example, a multilingual model could facilitate speech synthesis that involves code-switching between languages as it would not require language detection alongside multiple single-language models. The idea to try adapting a monolingual model to multi-language data came from observations of the original training strategy (described in Section 5.5) where warm-up models were used to train more complex models. This section presents work toward learning a multilingual VQ-VAE using the best-performing model that was identified from earlier experiments. This model was the dual-encoder model +**Adversarial** with softmax loss. The goal is not to learn to disentangle languages, but to learn representations of content and speaker that are shared across multiple languages. For example, to learn phone VQ representations from multiple languages in a single VQ codebook. At the same time, the disentanglement goal remains to be able to separate content from speaker identity. Successful training of a multilingual model would imply that languages can share the same vector space in VQ clustering.

5.8.1 Dataset Description

The multilingual Spoken Interaction with Interpretation in Switzerland (SIWIS) dataset [Goldman et al., 2016] contains four languages: English, German, French, and Italian. There are 36 unique speakers. Each speaker is bilingual or trilingual and has been recorded in two or three languages. The dataset languages were imbalanced, so the train/test splits also preserved this imbalance as shown in Table 5.2. As before with the training data, all audio was downsampled to 16 kHz and normalized with ITU-T Rec. G.191 sv56[9]. The leading and trailing silence was trimmed using VAD. The pre-processing steps were followed as before.

Table 5.2: SIWIS data splits across languages and speakers.

Language	Training		Validation		Held-out	
	Spk	Utt	Spk	Utt	Spk	Utt
English (EN)	18	2387	18	603	4	16
French (FR)	26	3405	26	841	5	16
German (DE)	13	1719	13	376	4	18
Italian (IT)	13	1689	13	430	3	10

[9] `https://github.com/foss-for-synopsys-dwc-arc-processors/G722`

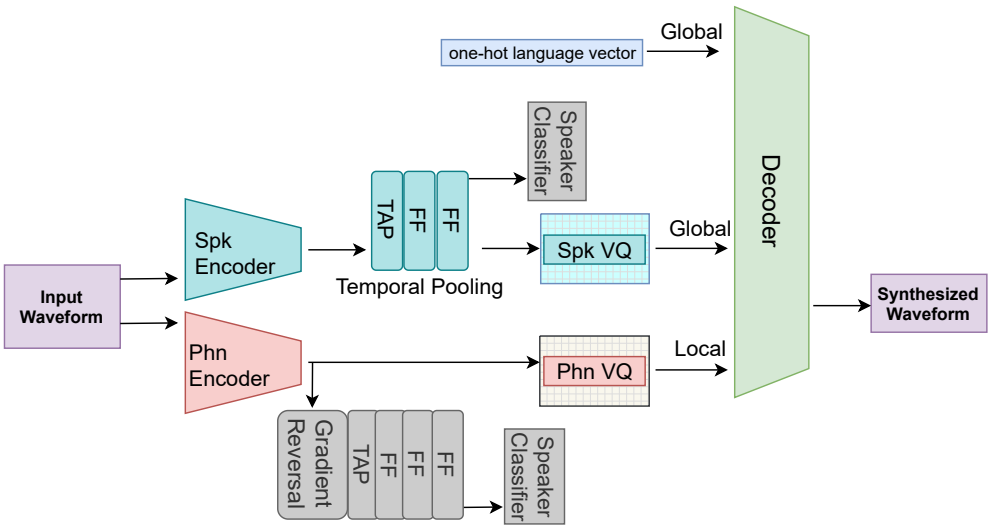


Figure 5.17: Multilingual VQ-VAE overview +**Adversarial-L**, including the global one-hot language vector for global conditioning of the decoder. The base architecture version was +**Adversarial** with softmax. This model includes a temporal average pooling layer (TAP) with two feed-forward layers (FF). There is an auxiliary speaker classifier as well as an adversarial classifier.

5.8.2 Training

Earlier experiments had indicated that the best performing dual-encoder architecture was +**Adversarial** so that model was used as a warm-up for training the multilingual model. It consists of two encoders and two VQ codebooks which modeled speaker as a global condition, and phones as a local condition. Training for this model was semi-supervised using speaker labels and adversarial loss.

To train the multilingual model from the warm-up model, a projection layer from the WaveRNN decoder was first discarded from the trained warm-up model, but all of the other weights and parameters were borrowed from the encoder and VQ clustering. The first version of the multilingual model did not utilize a one-hot language vector for global conditioning, as it is shown in Figure 5.17. The one-hot language vector was introduced to improve the model performance in unseen conditions. Without the language vector, some inconsistency was observed early on in the development process, such as intelligibility issues. When the one-hot language vector was introduced as a global condition to the decoder, it was simply concatenated to the global speaker condition vector. This was an obvious solution to try. Other solutions that were not explored could have been to train four separate language-dependent models, but that would not have been multilingual. In this chapter, the multilingual model was trained on all four languages mixed together for 550k steps while monitoring the validation losses. The input to the system encoders was the same as before. The input to the encoder was a waveform. After the waveform was downsampled by the encoder, it was transformed into a sequence of VQ codes and vectors. The VQ vectors were then provided to the WaveRNN decoder. Finally the output was a reconstructed waveform.

5.8.3 Estimated Naturalness of Synthetic Speech

The purpose of using estimated quality in this model adaptation case was to first identify the best-performing model variant, with or without a one-hot language vector. As before, the following automatic metrics were used to assess the synthetic speech quality (including natural speech for reference): MOS1, speaker similarity, ASR-based word error rate (WER), and F0 error (RMSE). These results are reported in Table 5.3. For this portion of the evaluation, the values for each metric were averaged across all four languages. As before in 5.4, the data was divided into four testing conditions based on seen or unseen content and speakers during training. The results reported in Table 5.3 are averaged across all four testing conditions in order to gain general insight about the quality of the synthetic speech. From the table, the two variants (**NL** and **L**) demonstrate similar performance in the average case.

Further investigation of performance across seen and unseen conditions is reported in Figures 5.18-5.21. The variant utilizing the one-hot language code (+**Adversarial-L**) tended to have better estimated MOS1 quality and intelligibility scores for unseen conditions. This variant also demonstrated better F0 for seen conditions in Figure 5.21. While the performance of these two variants does appear to be similar, especially for speaker similarity in Figure 5.19, the one-hot language code provides some minor improvement such as slightly better MOS1 and WER for unseen conditions. Therefore the +**Adversarial-L** model was selected for further evaluation in human listening tests.

Table 5.3: Speech synthesis quality estimation averaged across four testing conditions on SIWIS data to gain a general insight about the quality of the synthetic speech. The monolingual +**Adversarial** model from earlier experiments was used for adaptation to the multilingual dataset. This table compares estimated performance for model adaptation with a one-hot language vector **L** and without **NL**.

Method		Est. MOS1	Speaker Similarity	WER	log(F0) RMSE
Natural Speech	–	2.5	1.0	19.7	0.0
+ Adversarial	NL	2.9	0.8	63.3	0.4
+ Adversarial	L	3.0	0.8	59.5	0.4

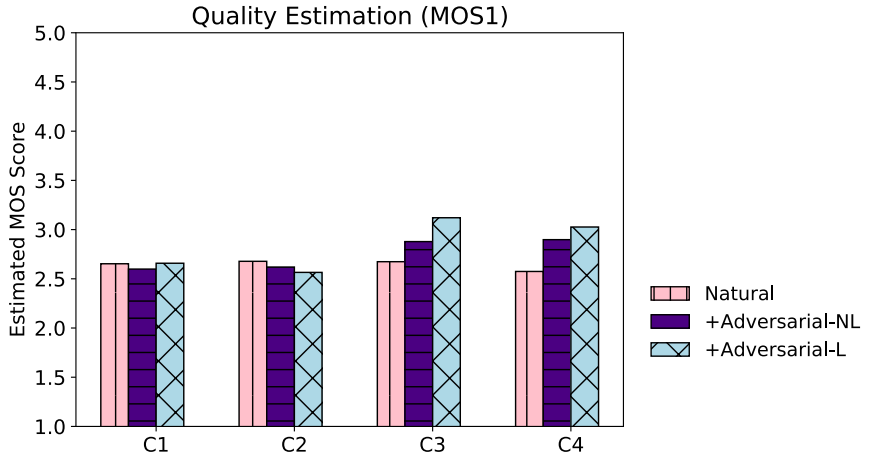


Figure 5.18: Estimated quality (MOS1), showing performance across four different testing conditions. Higher values indicate better performance. Four testing conditions were examined based on which content or speakers were seen during training: C1 is seen speaker/seen content, C2 is seen speakers / unseen content, C3 is unseen speakers / seen content, and C4 is unseen speaker / unseen content.

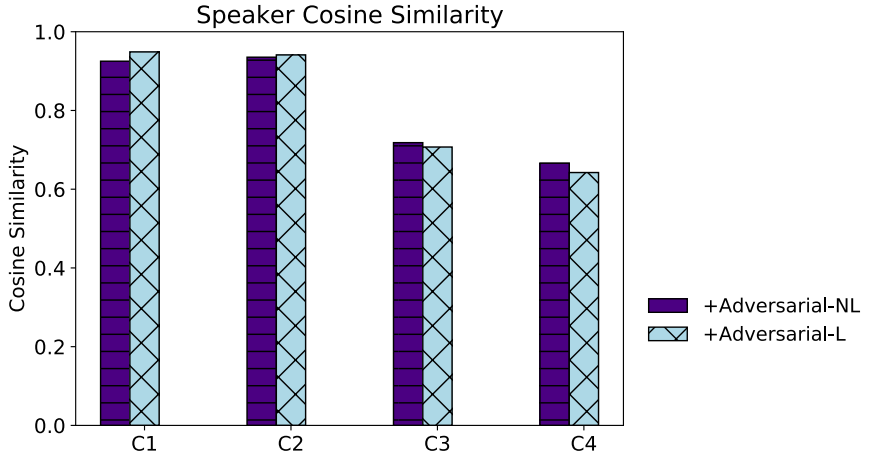


Figure 5.19: Speaker similarity, showing performance across four different testing conditions. (Natural speech has been omitted because the speaker comparison uses natural speech as the reference, so it is always $sim = 1.0$). Higher values indicate better performance. Four testing conditions were examined based on which content or speakers were seen during training: C1 is seen speaker/seen content, C2 is seen speakers / unseen content, C3 is unseen speakers / seen content, and C4 is unseen speaker / unseen content.

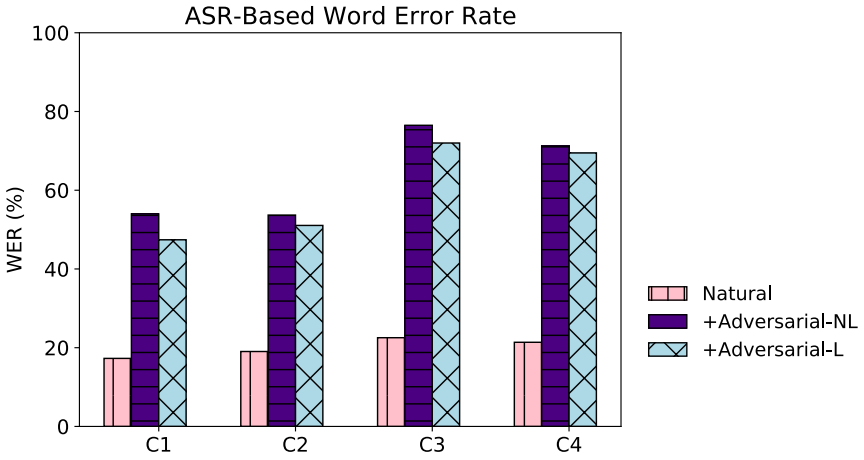


Figure 5.20: ASR-based word error rate (WER), showing performance across four different testing conditions. Lower values indicate better performance. Four testing conditions were examined based on which content or speakers were seen during training: C1 is seen speaker/seen content, C2 is seen speakers / unseen content, C3 is unseen speakers / seen content, and C4 is unseen speaker / unseen content.

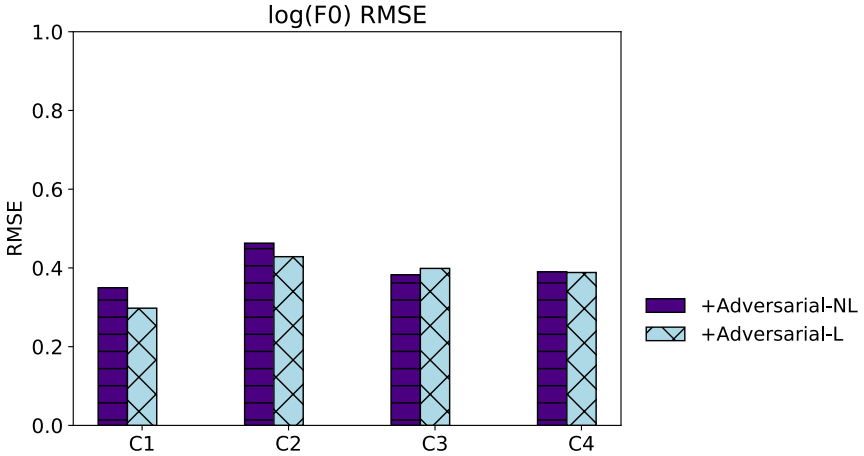


Figure 5.21: F0 error (RMSE), showing performance across four different testing conditions. (Natural speech has been omitted because the speaker comparison uses natural speech as the reference, so it is always $rmse = 0.0$). Lower values indicate better performance. Four testing conditions were examined based on which content or speakers were seen during training: C1 is seen speaker/seen content, C2 is seen speakers / unseen content, C3 is unseen speakers / seen content, and C4 is unseen speaker / unseen content.

5.8.4 Human Listening Test Evaluation

The synthetic speech from the multilingual model was also evaluated using human listening tests. For the listening tests, participants were recruited from the Prolific[10] platform and the listening test materials were hosted by Qualtrics[11]. All of participants self-identified as fluent in their respective languages. 20 participants were recruited and they were compensated at the rate of £ 7.50/hour for their time.

One of the most efficient ways to gauge the quality of the +**Adversarial-L** system is to perform copy-synthesis. If copy-synthesis quality is very good then the internal VQ representations are more likely to also be good, however this is not guaranteed. Likewise if the copy-synthesis quality is poor it is unlikely that the learned representations would be good.

The samples were divided into two conditions: *seen* and *unseen*. The seen condition means that a speaker and the utterance content had been observed during model training, and it corresponds to the C1 condition described earlier. The unseen condition means that the speaker and content were entirely held-out, and it corresponds to the C4 condition described earlier.

Listeners rated the naturalness on a Likert scale of 1-5 (where 5 is natural). For each language, 6 synthetic speech examples were evaluated, for a total of 24 samples. The participants were shown 4 pages of 6 speech samples in randomized order. They were asked to rate the speech on a scale of 1 to 5 where "5 indicates that it sounds like natural speech" and "1 indicates that it sounds like a machine". The average MOS naturalness scores are reported in Table 5.4 for both natural and synthetic speech. The results show similar MOS scores for the monolingual and multilingual models. In the multilingual model, English and German naturalness was significantly lower for the *unseen* condition. German copy-synthesis was particularly good for the *seen* condition. The monolingual model had better MOS in the unseen condition, which could be due to artifacts of selected samples since the natural speech also had increased MOS scores for the unseen condition.

Table 5.4: MOS naturalness scores for copy-synthesis. **C1**: speakers and content were observed during training. **C4**: held-out speakers and content. The changes in scores between natural and synthetic is also indicated with Δ.

	Natural		Synthetic			
Data	C1	C4	C1	Δ	C4	Δ
SIWIS-EN	4.2	4.1	2.3	$\downarrow$ 1.9	1.6	$\downarrow$ 2.5
SIWIS-FR	3.6	3.4	2.9	$\downarrow$ 0.7	2.9	$\downarrow$ 0.5
SIWIS-DE	3.7	3.7	3.5	$\downarrow$ 0.2	2.5	$\downarrow$ 1.2

[10] `https://www.prolific.co/`
[11] `https://www.qualtrics.com/uk/`

5.9 Further Considerations

In this chapter, there were several design considerations that led to using an end-to-end VQ-VAE architecture. An alternative to the architecture presented in this chapter would be to use speech spectrograms as input to VQ-VAE rather than waveform audio as was done for the experiments presented here. Mel-spectrograms are often used in text-to-speech techniques wherein the objective is to learn a mapping between acoustic units and graphemes (or phones). In speech synthesis it is not common to operate directly on waveform audio in the time domain because the data is extremely high dimensional and it is difficult to identify acoustic units in the time domain. It is therefore reasonable to consider an alternative system that utilizes a pre-trained neural vocoder which takes mel-spectrograms as input, and re-structure the VQ-VAE to function as an autoencoder that uses mel-spectrograms as input and output.

In this re-imagined architecture there are several components that would need to be adjusted. First the encoder and decoder layers would need to be adjusted for the dimensionality of mel-spectrogram data instead of the audio waveform. Currently the encoder downsamples the audio waveform by a factor of 64, so this downsampling would need to be adjusted along with the size of the convolutional filter. Similarly with the decoder, it would need to be adjusted to produce a mel-spectrogram from some input, such as a VQ vector. In this re-imagined architecture, the intermediate layers between the encoder and decoder would need to also be characterized differently. For example, there would no longer be a use for temporal average pooling layers. It is not clear if disentanglement between speaker and content could be achieved since it may be difficult to formulate a speaker representation from a mel-spectrogram. The input may need to change so that the speaker encoder operates on an extracted signal-level feature such as mel frequency cepstral coefficients (MFCCs). Therefore the input to the speaker encoder and the input to the phone encoder would be different. Furthermore, the decoder would need to produce a mel-spectrogram as output using global and local conditions and if that is not possible then the idea of global and local conditioning would need to be re-imagined.

Essentially the variant of the system described so far would no longer be end-to-end. It is not clear if having two different information sources would result in disentanglement or not. Ultimately the learned representations would likely be defined as *separately learned representations* but not *disentangled representations*. The idea behind disentanglement is to separate two or more types of information from a single source, so even if this re-imagined system variant performed well as an autoencoder, it may no longer be considered disentanglement. The next chapter will explore what disentanglement means and provide more insight into how this can be evaluated intrinsically.

5.10 Conclusion

The purpose of this chapter was to explore an end-to-end approach to create disentangled representations of speech features. The features that were focused on here were: speaker identity, phone content, and F0. The success of this end-to-end approach has been measured by the quality of the synthetic speech produced by the autoencoder framework. If the quality of the synthetic speech is very low, then it is unlikely that meaningful representations have been learned in the intermediary VQ clustering spaces. Therefore the evaluation of the approach has focused on the quality of the speech output. Three principles were introduced relating to disentanglement:

1. The learned representations are sufficiently rich to model targeted speech features.
2. Representations that model targeted speech features do not also model non-target speech features (e.g., a representation of speaker identity does not also model speech content).
3. Representations have utility outside of the system that disentangles information.

The evaluation conducted in this chapter on re-synthesized speech has made an attempt to address the first principle (#1). The learned representations can model targeted speech features as global and local conditions when these conditions are passed to a trained decoder such as WaveRNN. It is perhaps too early to claim that the learned representations are *sufficiently rich*, as this will be addressed through further analysis in Chapter 6 and Chapter 7.

The best-performing dual-encoder system (+**Adversarial**) was identified alongside the best-performing triple-encoder system (+**Speaker**/**F0**). Adding a semi-supervised VQ component with either speaker labels or adversarial loss improves the speech output quality over fully self-supervised systems. Automatic estimates of speech synthesis naturalness, intelligibility, and speaker similarity can be improved over previous work in seen and unseen conditions. Further experiments involving model adaptation have shown that it is possible to adapt a multi-speaker monolingual model to a new multi-speaker multi-language dataset with reasonable naturalness scores for copy-synthesis. The ability to synthesize speech from multiple languages in a single model is very promising for other multilingual applications such as code-switching. The next important step is to gain an understanding as to whether or not the learned VQ representations are robust and meaningful, and whether or not they are disentangled. To better understand how well the models have performed disentanglement, Chapter 6 will explore the VQ codebooks of the trained systems in further detail, and assess whether or not the learned representations have met any of the proposed disentanglement criteria.

Chapter 6

Intrinsic Analysis of Disentangled Representations

6.1 Introduction

In the previous chapter, a method was presented to re-synthesize speech while also learning representations of speech that correspond to speaker identity, phone content, and prosodic style. Representations were learned using several different variations of a VQ-VAE system wherein each variation utilized a different number and type of VQ codebook, different loss function with additional multi-task classifiers or gradient reversal layers. The purpose of simultaneously learning multiple different types of codebooks (such as speaker and phone) was to encourage disentanglement of the information. Overall, the previous chapter presented three different classes of VQ-VAE systems wherein the differences are distinguished based on how many distinct VQ codebooks were learned simultaneously during the model training process. All of the systems utilize at least a phone codebook, as with the original **VQ-VAE** system:

1. single-encoder: **VQ-VAE**
2. dual-encoder: +**F0**, +**Global**, +**Speaker**, +**Adversarial**
3. triple-encoder: +**Global**/**F0**, +**Speaker**/**F0**, +**Adversarial**/**F0**

The re-synthesized speech from all systems was evaluated based on naturalness from estimated mean opinion scores (MOS), naturalness MUSHRA tests, speaker cosine similarity, ASR-based word error rate, and F0 error.

The original VQ-VAE system (**VQ-VAE**) utilized one single codebook which is thought to contain phone information [van den Oord et al., 2017]. Among the dual-encoder system variants, the best performing model made use of multi-task learning and gradient reversal with softmax loss (+**Adversarial**). Among the triple-encoder system variants, the best performing model made use of an auxiliary speaker classifier and softmax loss (+**Speaker**/**F0**). At the end of Chapter 5, an additional model variant was introduced wherein the monolingual English +**Adversarial** model was adapted for multilingual speech synthesis by adding a one-hot language encoding vector for decoder global conditioning (+**Adversarial-L**). The multilingual model was

trained to re-synthesize speech in four languages combined together during training (English, French, Italian, and German).

While the quality of the re-synthesized speech was evaluated to identify top-performing models, the previous chapter did not address any aspect of evaluating disentanglement among the learned representations. This chapter explores the learned codebook content using intrinsic analysis in an effort to characterize disentanglement. In the previous chapter, three principles of disentanglement were introduced:

1. The learned representations are sufficiently rich to model targeted speech features.
2. Representations that model targeted speech features do not also model non-target speech features (e.g., a representation of speaker identity does not also model speech content).
3. Representations have utility outside of the system that disentangles information.

The analysis presented in this chapter seeks to address the first and second principles (#1 and #2) by examining the content and relationships of learned representations. The learned

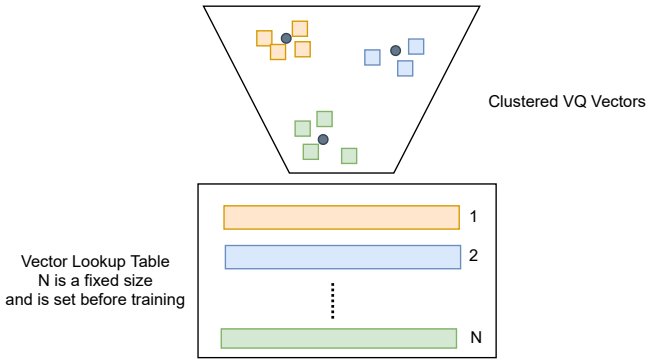


Figure 6.1: Concept of a codebook demonstrated by a lookup table that contains N entries, and a clustering space that was learned during training. The lookup table allows a one-to-one mapping from a codebook index $[1, 2..N]$ (referred to as a *code*) to a 128-*dim* vector representation (referred to as a *vector*).

representations can be found within the trained VQ codebooks. Each VQ codebook is a lookup table where the key is an integer and the value is a 128-*dim* vector. Recall the description of the VQ codebook from Chapter 2, as shown again here in Figure 6.1. A VQ codebook is of size N, where the code $[1, 2..N]$ has a corresponding vector. The result of training is a lookup table where meaningful vectors can be obtained by looking up a code. For systems that utilize more than one encoder (e.g., +**Adversarial**), the phone codebook is learned jointly with the speaker codebook, yet they are two separate codebooks. Therefore the representations learned by the +**Adversarial** system include phone codes and phone vectors, as well as speaker codes and speaker vectors.

A single utterance of speech is represented by a sequence of phone codes, as demonstrated in Figure 6.2. The length of the sequence of phone codes depends on the compression ratio in the encoder as well as the length of the original utterance (described in Section 2.4.1). The compression ratio in all experiments used in this thesis was $ds = 64$, and the downsampling rate was $N/T = 250$ codes/second. For codebooks that represent F0 information, this same sequence

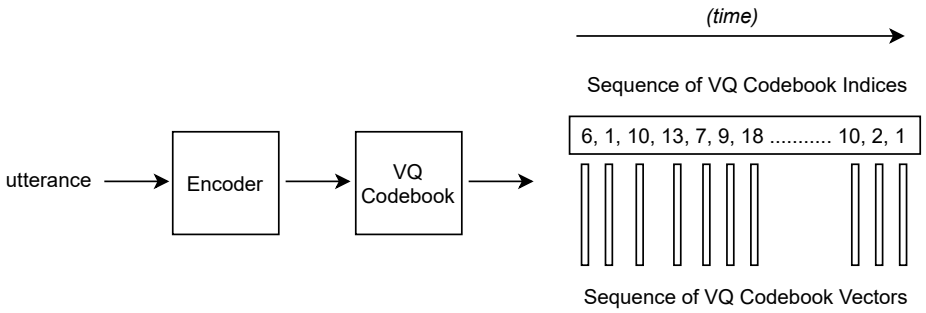


Figure 6.2: Concept of obtaining a sequence of codebook indices (*codes*) versus a sequence of codebook vectors (*vectors*) that can be passed further to the decoder as global or local conditions, or used for analysis in this chapter.

and compression ratio are applied, however the F0 codes correspond to a different downsampling rate due to the frame length of the input F0 used in training the encoder. The input F0 used in training had a frame length of $frame = 5$ ms and the resulting F0 downsampling rate was $N/T = 340$ codes/second. The mismatch between the quantity of F0 codes and the quantity of phone codes is handled with a transposed convolutional layer for up-sampling, as described in Zhao et al. [2020b]. For codebooks that represent speaker information, a single utterance yields a single speaker code due to the temporal average pooling. Therefore a speaker code represents one utterance, regardless of the utterance duration. This chapter explores the content of the sequences of codes from the F0 and phone codebooks, as well as the codes from the speaker codebooks. Four different VQ-VAE system variants were selected for this analysis by identifying the best-performing system among the variants that were introduced in Chapter 5. The best-performing system was determined by the overall highest quality speech synthesis and ability to generalize to unseen speakers/content:

- single-encoder **VQ-VAE**
- dual-encoder +**Adversarial**
- triple-encoder +**Speaker/F0**
- dual-encoder (multilingual) +**Adversarial-L**

The objective in selecting four top-performing systems for this analysis is to understand if any particular system achieves better disentanglement of phone, speaker, and F0 information compared to the baseline **VQ-VAE**. It would not be beneficial to further analyze system variants that resulted in lower-quality speech synthesis as determined in Chapter 5 based on the principle that lower-quality speech synthesis is likely to be related to lower-quality learned representations. Another objective of this chapter is to understand if adding additional codebooks or adapting to multiple languages causes a loss of disentanglement or lower-quality learned representations. It may be possible, for example, that adding additional codebooks does not result in better disentanglement but does allow for learning additional representations without sacrificing any aspect of quality for synthesized speech or for the learned representations. If that is true, in principle the additional F0 representations may be useful in downstream applications that benefit from explicit F0 modeling, such as text-to-speech synthesis.

There is no single analysis that can determine or quantify disentanglement. Therefore this chapter presents multiple different perspectives of analysis with each one probing or testing the relationships between speaker, content, and F0 representations. Taken all together, the analysis presented in this chapter will allow for some conclusions about disentanglement to be drawn based on quantitative evidence. The analysis presented in this chapter will first examine the frequency distributions of VQ codes for a set of utterances. In the case of phone and F0 code sequences, the sequence will be examined for consistency and systematicity. Systematicity in language refers to statistically relevant groupings of words and sounds and the meaningful application and usage of these categories within language [Dingemanse et al., 2015]. If language was not systematic then words and sounds would be arbitrary. The analysis will aim to discover if sequences of VQ codes behave like language or if the sequences appear to be random. In the case of speaker codes, the analysis will examine the mapping between unique speaker from the data and unique codebook entry (referred to as *speaker code*). The interplay between speaker and phone representations will be examined using language modeling and a data probing task. Finally, the vectors of each type of codebook (speaker, phone, F0) will be visualized in a 2D space. In the case of multilingual codebooks from +**Adversarial-L**, additional analysis will be performed to assess similarities and differences between +**Adversarial** and +**Adversarial-L**.

6.2 Related Work

There has been an increasing amount of work that utilizes the VQ-VAE paradigm to improve speech synthesis [Zhao et al., 2020b] or to model F0 or prosody [Wang et al., 2019b; Zhao et al., 2020b; Zhang et al., 2020], and word-level representations [Fong et al., 2021]. In much (but not all) of this related work, the authors present a form of assessment of the learned VQ representations. For example in the work of Wang et al. [2019b], their Figure 11 presents original and estimated F0 contours alongside a trajectory path through the learned F0 codebooks using multiple different levels of linguistic abstraction (word, syllable/mora and phone levels). The trajectory visualizations help the reader to understand how VQ code sequences are related to rising and falling prosody in speech. The visualizations did not, however, inform the reader of how each F0 code sequence is related to other aspects of speech features because F0 was the only speech feature being modeled.

Others, such as Zhao et al. [2020b] and Zhang et al. [2020] have explored learning F0 representations. In the work of Zhao et al. [2020b], an F0 codebook was learned using VQ-VAE and this was applied to Japanese and Chinese, two tonal languages that may benefit from explicit F0 modeling for speech synthesis. The work of Zhang et al. [2020] found that VQ-VAE could learn representations that control prosody through duration, word boundaries, and increasing/decreasing pitch. However, when the self-supervised technique from Zhao et al. [2020b] was used for English data in Section 5 of this chapter, it was found that the VQ F0 codebook only learned one VQ code repeated over and over and it was the same for all English utterances. As with much of the other related work, the assessment of learned representations for VQ-VAE is mostly based on using them to perform tasks. A detailed analysis of intrinsic properties of representations tends to be missing from the literature.

Another way to analyze learned representations from VQ-VAE comes from Fong et al. [2021] where they applied time alignments to sequences of VQ phone codes to create word-level units of

speech using multi-speaker English data. The units were then concatenated together in various ways to change the words in an utterance. They found that the intelligibility of concatenated units was degraded when the speaker was mismatched (meaning that different units came from different speakers). Their analysis is insightful for a variety of speech applications, especially text-to-speech. However, their work was limited to the original VQ-VAE architecture using only a phone codebook. They did not explore whether similar words contained similar sequences of VQ phone codes.

6.3 Codebook Usage and Frequency Distributions

The purpose of analysing codebook usage and frequency distributions is to understand if the phone and F0 code sequences follow similar patterns as natural speech. It also provides a way to assess whether the entirety of the codebook was utilized, as the size was set and fixed during training. The number of entries used in a given codebook depends on training and whether the VQ loss function created many clusters or fewer. At the time of writing, there are no existing guiding principles to determine the necessary size of a codebook before training. The size of the phone codebook was 512, and the size of the F0 codebook was 10, both in line with previous work [Zhao et al., 2020b; van den Oord et al., 2017]. For the new speaker codebook component introduced in this thesis, the size of the speaker codebook was set to 256 in order to be smaller than the phone codebook but large enough for the quantity of unique speakers in the VCTK training data (110 speakers).

6.3.1 Data Preparation

This section describes how the data was pre-processed for the purpose of analyzing codebook usage and frequency. First, the codebook sequences were obtained from speech training data by making a forward pass (inference) through each system and saving the resulting VQ codes and vectors. This resulted in a sequence of phone codes and a sequence of phone vectors for each individual utterance. Similar sequences were obtained for F0 in the case of the triple-encoder system. The speaker information was represented by a single speaker code and vector, except for the original **VQ-VAE** system which did not have a speaker codebook component. The audio and text data were also passed to a forced alignment system to obtain time-aligned speech and text, with alignments at the phone and word levels. For this step, the Montreal forced aligner was used [McAuliffe et al., 2017].

Next, a random sample of 1,000 utterances was selected from the training set. (This random sample corresponds to "condition1" described in the previous chapter, wherein the speaker and content of the utterance were seen during training). These utterances were selected from among 99 speakers. Approximately 10 utterances per speaker were used in the random sample. The sequence of phone and F0 codes was further processed to remove unvoiced/silence. To do this, first the sequence of codes was aligned to the forced alignments from the Montreal forced aligner. The forced alignments were checked for phone silence tokens (/sil/ and /sp/) or word silence tokens ("SILENCE") that correspond to silence padding or pauses within the utterance. The corresponding phone and F0 codes were removed if they were aligned to the word or phone silence tokens.

6.3.2 Phone Codebook

The phone codebook was analyzed in this section using a technique borrowed from corpus linguistics. This technique examines language usage based on the distribution of high and low frequency vocabulary types. The goal is to describe how word types or phone types are distributed within human languages. Further, in this section the same idea will be applied to VQ phone codes to describe how the codes are distributed and make comparisons between VQ codes and words/phones. All human languages exhibit systematicity and the frequency distribution of types will tend to follow a power law called a Zipfian distribution [Zipf, 1970]. This means that some word types (primarily function words such as: to, from, of, in, for, etc) are significantly more frequent than other word types in the language (such as content words). Systematicity is observed because language is highly structured [Gries and Ellis, 2015]. While Zipfian distributions among VQ codes do not directly imply disentanglement, it would suggest that distributions of VQ codes behave similarly to distributions found in natural language. If the VQ codes behave similarly to natural language then it may be possible to find an association between sequences of VQ phone codes and phone content. If VQ phone codes are highly correlated to phone content, then they are less likely to be correlated with individual speakers, and this may suggest disentanglement between content and speaker information.

In this section, the systematicity of phone codes is analysed in terms of the rank-ordered frequency distribution using the sequences of codes that were obtained for all utterances. The distribution of codes is compared to the distribution of words and phones in terms of unique vocabulary. Until this analysis, it has not been determined whether VQ phone codes exhibit systematicity similar to words and phones in natural language. Examining rank-sorted frequency distributions provides only a rough analysis. It does not fully elaborate the relationships between codebook vectors or disentanglement. The vocabulary is examined here as a starting point for further analysis such as predicting which codes correspond to a particular phone, and which vectors are closer or far apart, described in Sections 6.4 and 6.5.

The words, phones, and phone codes will be examined in terms of vocabulary. In the case of words, the vocabulary refers to the set of unique word types. The frequency of a vocabulary item refers to the number of tokens belonging to the same type (for example, the number of times the word "to" occurs in the data). In the case of phones, the vocabulary refers to the set of unique phone types. The phoneset used by the Montreal forced alignments was ARPABET which is provided by default with that tool. In the case of VQ phone codes, the vocabulary refers to the set of unique phone code types that were active for all of the utterances in an inference step (not necessarily the size of the codebook if some codes were not used for the utterances). In addition, three other aspects of vocabulary are explored: bigrams, phone-aligned codes, and sentencepiece. Bigrams are determined as a pair of consecutively occurring tokens. The phone-aligned codes are the sequence of codes that are grouped together based on the time alignments for each instance of each phone. For example, the phone-aligned sequence for the phone $/ah/$ turns out to be the code sequence $[8, 5, 9, 2]$ in one utterance and $[3, 2, 5, 9, 5, 7, 8]$ in another utterance; each code sequence would constitute a separate and unique grouping, and thus a separate and unique vocabulary entry (referred to as Phone-Groups in Table 6.1). In the case of sentencepiece, this refers to the application of the tool called **sentencepiece**[1,2] that

[1] `https://pypi.org/project/sentencepiece/`
[2] `https://github.com/google/sentencepiece#c-from-source`

determines groupings of phone codes based on statistical co-occurrences. It was trained on the sequences of phone codes to create a larger vocabulary. The vocabulary was pre-set and two sizes were compared: 256 and 512 (referred to as SP_256 and SP_512 in Table 6.1).

From Table 6.1 there appears to be a lack of systematicity in the phone codes for all three systems. This is indicated by the very large vocabulary sizes for bigrams as well as phone aligned code groupings. For example, if there are 91 possible unigram types in the original **VQ-VAE**, then for bigrams the 91 possible types are combining as the maximum possible number of bigrams: $91 * 91 = 8281$, if each unigram combines uniquely with each other unigram. Since the table indicates a value of 7878 for bigrams, this suggests that every unigram is combining with almost every other possible unigram. If the phone codes were combining in a way that was more systematic there would be a smaller vocabulary size, and the values in the table would be smaller. An example of this is shown for words types and phone types. The unigram vocabulary size for phones is 58, while the bigram vocabulary size is 693. These values for phone vocabulary indicate that on average, a phone unigram type combines with approximately 11 different other unigram types when forming bigrams. Perplexity will be defined and discussed in a separate analysis in Section 6.4, though it is worthwhile to mention here that the large vocabulary sizes correspond to a high perplexity relative to the linguistic units of words and phones.

A consequence of having low systematicity is there may be some difficulty with modeling or predicting the phone code sequences. However as the previous experiments from Chapter 5 have shown, it is possible to re-synthesize high quality speech so the phone code sequences are probably meaningful even if they do not follow similar systematicity patterns as words and phones. It is an open question of whether the information is found in the way phone codes are distributed or the learned weights of the decoder from training. The ability to predict a phone from a code (and predict a code from a phone) will be further explored in later sections of this chapter (Section 6.4.3). For the phone-aligned codes, the consistently high vocabulary sizes across systems suggests that most instances of phones being represented by unique phone code sequences which also reflect the duration of a phone. For example if an instance of a phone such as /*ah*/ is represented by the VQ phone code sequence $[101, 23, 76, 92]$ then another instance of the same phone /ah/ could (in theory) be represented by the VQ phone code sequence $[5, 13, 21, 76, 92, 64]$. These two sequences would then be treated as separate vocabulary types in this rough analysis of types and tokens. The vocabulary sizes from sentencepiece (SP_256 and SP_512) did not help alleviate this problem because sentencepiece simply imposed a statistical grouping method that created the specified vocabulary size ($v = 256$ and $v = 512$), but further analysis may reveal an interesting frequency distribution based on the vocabulary size.

The next step in observing systematicity and codebook usage is to examine rank-sorted frequency distributions. These are shown in Figures 6.3 to 6.7. To create the plots, first the frequency of each vocabulary item was calculated. The relative frequency was obtained by dividing each frequency value by the total number of tokens. The vocabulary items were then sorted in descending order based on the values for relative frequency. This placed the highest relative frequency vocabulary item in rank 1 and the lowest relative frequency vocabulary item last. These rank-sorted distributions were then plotted. The top 40 rank-sorted phones and words are displayed in Figure 6.3 and Figure 6.4, respectively. Both of these distributions appear to follow the Zipfian power law, even though the data is based only on 1,000 utterances. The phone code distributions for **VQ-VAE** in Figure 6.5 and +**Speaker**/**F0** in Figure 6.7 are not as clearly Zipfian, but also not as flat as the distribution for +**Adversarial** in Figure 6.6. Given

Table 6.1: Comparison of vocabulary sizes based on modeling unique word types, unique phone types and unique VQ phone code types from 1,000 randomly selected utterances across 99 speakers from the VQ-VAE training set. The vocabulary was modeled using unigrams, bigrams, and sentencepiece for VQ phone codes as well as the phone-level forced alignments to obtain groupings of VQ phone codes. Codebooks from three variants of the VQ-VAE systems were examined: **VQ-VAE**, +**Adversarial**, and +**Speaker/F0**. For sentencepiece, two different vocabulary sizes are compared (SP_256 and SP_512).

	Vocabulary Size				
Method	Unigrams	Bigrams	SP_256	SP_512	Phone-Groups
Words	235	386	–	–	–
Phones	58	693	–	–	–
Phone Codes VQ-VAE	91	7878	252	508	60552
Phone Codes +Adversarial	170	26280	252	508	60325
Phone Codes +Speaker/F0	108	10505	252	508	60587

that the best-performing system for speech synthesis was +**Adversarial**, it is possible that the relatively flat distribution of code sequences provides an advantage or the combinations of VQ phone codes exhibit redundancy and systematicity in a different manner not reflected here.

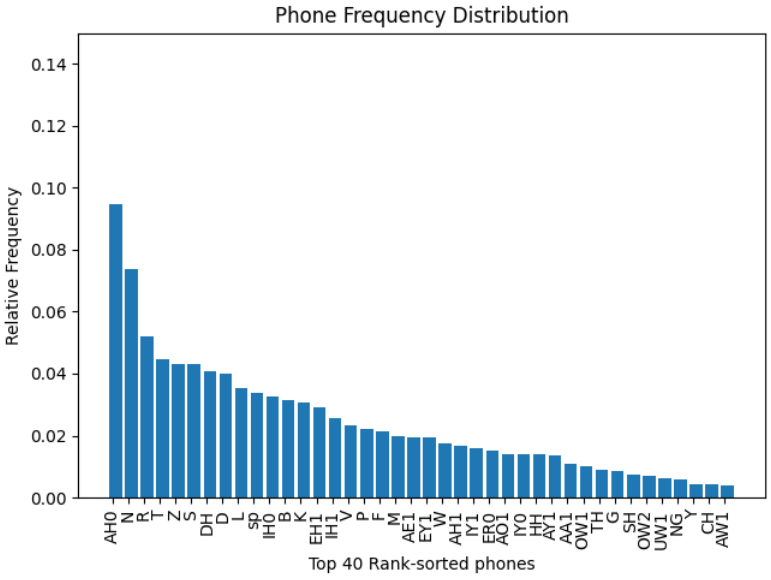


Figure 6.3: Top 40 most frequent phones from 1,000 randomly selected VCTK utterances and using phones from the Montreal forced alignments. Any phones corresponding to silence were removed. This plot shows the relative frequency for the top 40 rank-sorted phones.

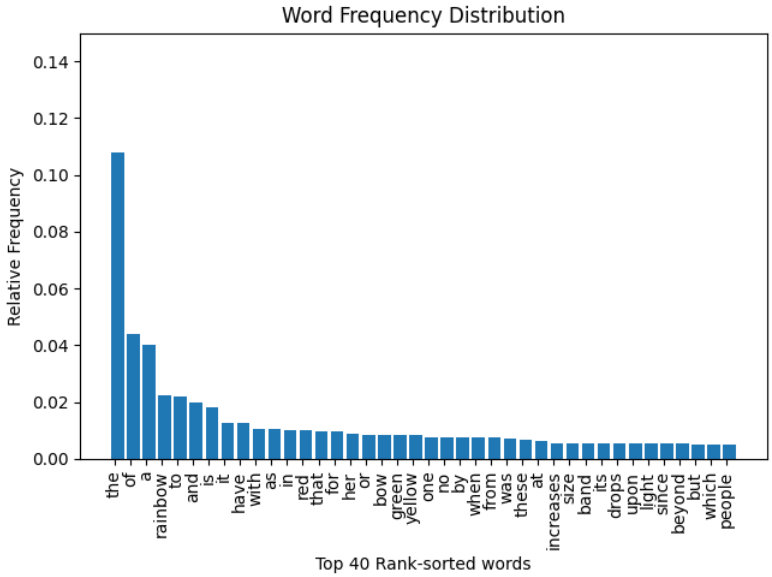


Figure 6.4: Top 40 most frequent words from 1,000 randomly selected VCTK utterances and using words. Any words corresponding to silence were removed. This plot shows the relative frequency for the top 40 rank-sorted words.

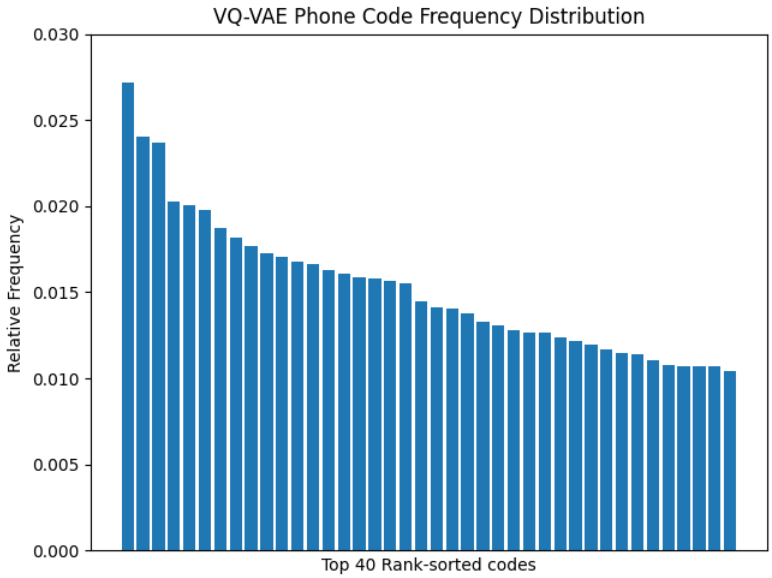


Figure 6.5: Top 40 most frequent phone codes from the **VQ-VAE** model. The distribution is sorted according to the highest value of relative frequency. The distribution appears to be similar to a Zipfian distribution, indicating that some of the VQ phone codes may be used systematically in the VQ phone code sequences for each utterance.

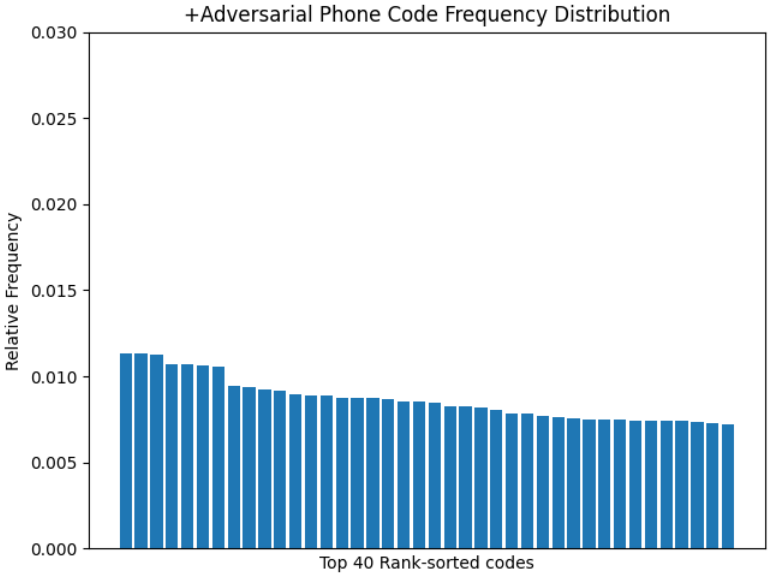


Figure 6.6: Top 40 most frequent phone codes from the +**Adversarial** model. The distribution is sorted according to the highest value of relative frequency. The distribution appears to be more flat for this system variant compared to other system variants. It may be possible that this system does model language systematicity or redundancy, but it is not captured in this analysis.

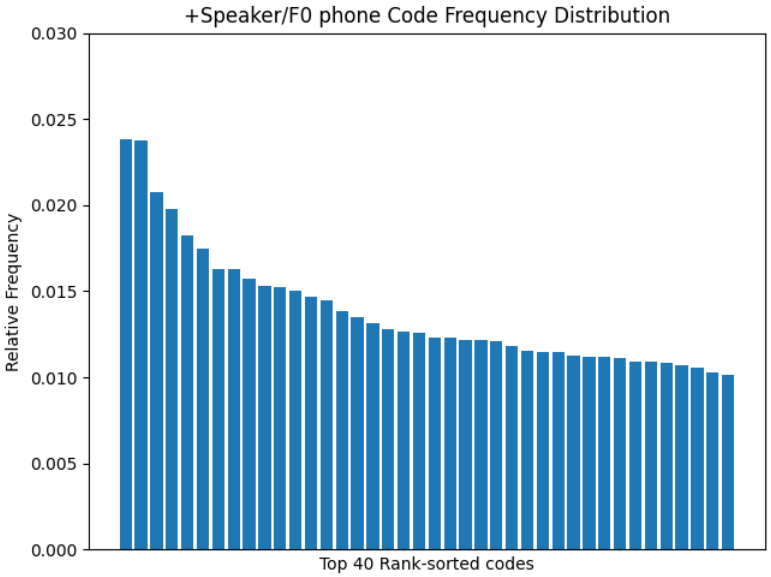


Figure 6.7: Top 40 most frequent phone codes from the +**Speaker/F0** model.The distribution is sorted according to the highest value of relative frequency. The distribution appears to be similar to a Zipfian distribution, indicating that some of the VQ phone codes may be used systematically in the VQ phone code sequences for each utterance.

The rank-frequency distributions are also visualized using a $\log_{10}$-$\log_{10}$ scale and compared side by side for unigrams (including words and phones) in Figure 6.8, for bigrams in Figure 6.9, and for sentencepiece SP_256 in Figure 6.10. For the reference of phones and words in Figure 6.8, it can be seen that the rank-frequency distribution has a clear downward trend though it is not linear. Words and phones that are lower-ranking are also lower-frequency. The lowest ranking items are orders of magnitude less frequent than the highest ranking items. On the other hand, the VQ phone codes for unigrams in Figure 6.8 do not follow a similar pattern. The VQ phone codes for all three systems have little variation in the frequency despite some items being lower-ranking. A similar trend for VQ phone codes is also observed for bigrams in Figure 6.9. The sentencepiece SP_256 vocabulary in Figure 6.10 has a pattern that is slightly similar to words and phones. Overall there appear to be few differences between the three system variants for unigrams, bigram, or sentencepiece. From Figures 6.8–6.10, it can be seen that most VQ phone codes have similar frequency regardless of unigram, bigram, or sentencepiece and regardless of the VQ system. This shows a lack of systematicity and indicates that the VQ phone codes are statistically distributed in an arbitrary manner. This lack of systematicity is not necessarily evidence against disentanglement, but it does not provide any evidence for disentanglement and shows that VQ phone codes do not behave like language. It also shows that the VQ variants +**Adversarial** and +**Speaker/F0** do not significantly change the systematicity from the original **VQ-VAE** system.

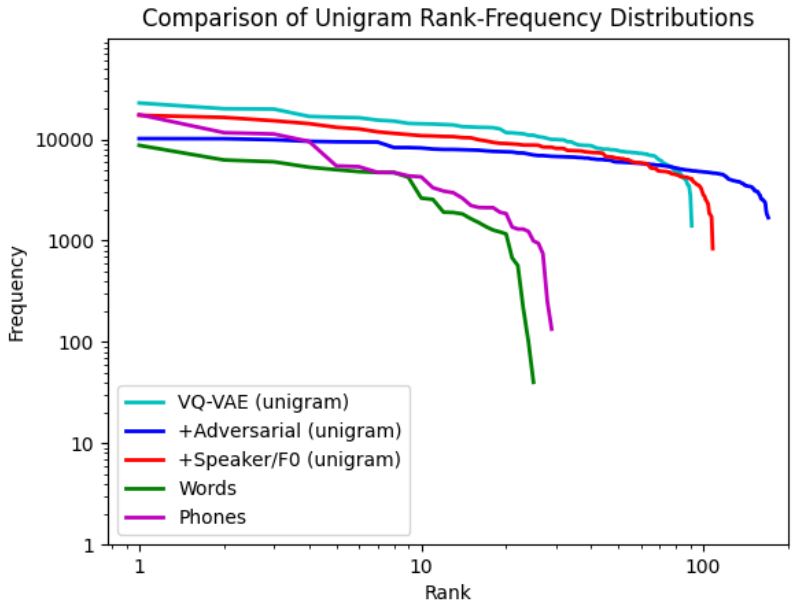


Figure 6.8: Plot of the rank frequency distribution (using a log-log scale) of unigram words, phones, and VQ phone codes from **VQ-VAE**, +**Adversarial**, and +**Speaker/F0**. The VQ phone codes do not follow a similar pattern as the words and phones.

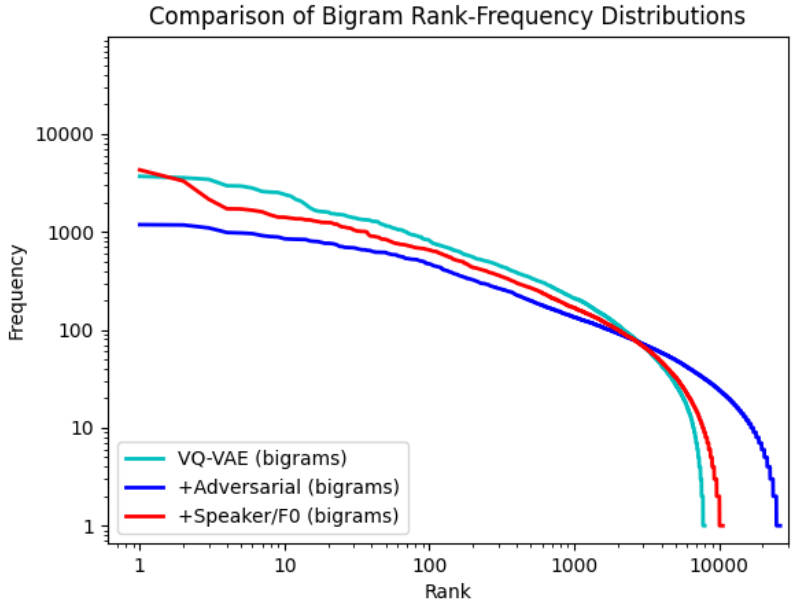


Figure 6.9: Plot of the rank frequency distribution (using a log-log scale) of bigram VQ phone codes from **VQ-VAE**, +**Adversarial**, and +**Speaker/F0**.

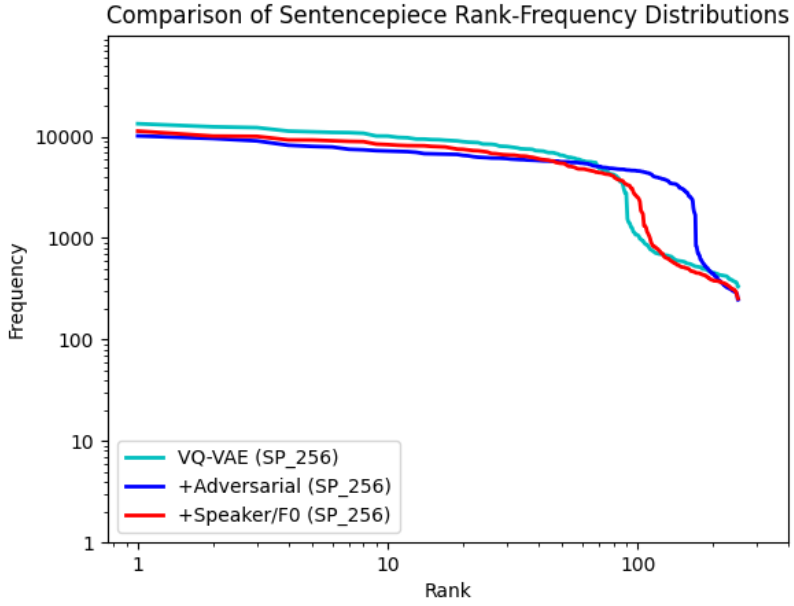


Figure 6.10: Plot of the rank frequency distribution (using a log-log scale) of sentencepiece SP_256 VQ phone codes from **VQ-VAE**, +**Adversarial**, and +**Speaker/F0**.

6.3.3 Speaker Codebook

To examine the speaker codebook usage, only two systems are compared: +**Adversarial** and +**Speaker/F0**. This is because the original **VQ-VAE** system relied on one-hot speaker encodings so there is no speaker codebook to examine. Unlike the phone codebook described earlier, the speaker codebook results in each utterance being modeled by a single speaker code. This is due to the temporal pooling layer in the model. The speaker code is meant to model utterance-level features that correspond to the identity of the speaker.

Table 6.2: Summary of 5 unique VCTK speakers with metadata showing the diversity of speaker codes in the +**Adversarial** and +**Speaker/F0** system variants.

VCTK Speaker	Gender	Age	English Dialect	Speaker Codes	
				+**Adversarial**	+**Speaker/F0**
p306	F	21	American	67, 52	190
p336	F	18	S. African	232, 22	201, 188, 252, 36
p272	M	23	Scottish	30	43, 50, 179
p264	F	23	Scottish	19, 12	201, 50, 188, 237
p226	M	22	English	193, 188, 30, 113	168, 179, 237

From the 1,000 randomly selected utterances that were used for analysis in this section, 5 unique speakers were randomly chosen for examination of their mapping into speaker codebooks. Table 6.2 shows how each speaker is mapped into speaker codes for each system. In general, it appears that a given speaker is mapped into one or more speaker codebook representations regardless of which system is used. This finding of a one-to-many mapping is counter to the notion that the speaker codebook is modeling unique speakers. Later analysis in this chapter will explore the distribution of speaker codes for a given speaker using conditional probabilities (Section 6.4.3) as well as how the codebooks are only partially utilized (Section 6.5). Although the speaker codebook size was initialized to 256 entries for +**Adversarial** and +**Speaker/F0**, each system only learned to utilize 18 and 19 codebook entries during training for 99 unique speakers. In general, based on the information in Table 6.2 it appears that the system variant +**Speaker/F0** may map a given speaker into more VQ codebooks compared to the +**Adversarial** system. To quantify this observation, all of the speaker codes were counted for the 99 speakers who occurred in the selected 1,000 utterances. The unique speaker codes per speaker were determined. On average, the +**Adversarial** system maps a speaker into 2.43 unique speaker codes ($std = 1.13$) out of 18 while the +**Speaker/F0** system maps a speaker into 2.92 different speaker codes ($std = 1.23$) out of 19. For comparison, the unique speakers per speaker code were determined. On average, the +**Adversarial** system uses a single speaker code for 13.05 unique speakers ($std = 8.68$) and the +**Speaker/F0** system uses a single speaker code for 16.89 unique speakers ($std = 8.39$). This finding, that there is not a one-to-one mapping between unique speakers and speaker codes, suggests that the learned representations of speakers and phones are not being fully disentangled during training by either of these two VQ-VAE system variants. If there was disentanglement, each speaker would be represented by a unique code. Instead, the VQ phone codes may contain information that distinguishes between speakers which would be evidence against disentanglement.

6.3.4 F0 Codebook

This section presents an initial view of how the F0 codebook was utilized in the system variant **+Speaker/F0**. Recall from Chapter 5 that quantized F0 was provided to the **+Speaker/F0** system variant during training. The quantization was performed in a manner that associated F0 values with one of 10 categorical bins according to the minimum and maximum range of F0 in the training data.

To review the F0 quantization process, first the F0 was obtained using the Reaper[3] pitch tracking tool. The frame rate was $fr = 5$ fps. The minimum F0 and maximum F0 for the training data was noted and that range was divided into bins. Any F0 values outside of the min/max values were treated as unvoiced. Each of the 10 bins covered equal proportions of the min/max range, meaning that the quantization bins scale linearly to the min/max values. In the VCTK training data, the minimum F0 was determined as $F0_{min} = 29.0$ and the maximum was $F0_{max} = 716.0$, both were determined automatically based on Reaper pitch tracking. After quantization, each F0 value was then represented by a single integer. For example, if three frames in a row had an F0 value of $F0 = 167.0$ before quantization, then the same three frames could be represented as $[3, 3, 3]$ after quantization (because the original F0 value corresponds to bin #3).

To obtain VQ F0 codes, a similar inference process was used as described earlier for phones and speakers. The audio was provided to the system for inference and the resulting F0 codes were saved. The downsampling rate for VQ F0 codes from the F0 encoder was $N/T = 340$ codes/second. For the 1,000 randomly sampled utterances used in this analysis, the average length of F0 code sequences was $L_{avg} = 1757.5$ with $std = 761.2$.

In this analysis, the original, quantized, and VQ F0 codes are plotted and shown for two different speakers saying the same utterance ("Please call Stella"). The speaker in Figure 6.11 and Figure 6.12 is a 22-year-old female who has a Northern Ireland dialect of English. The speaker in Figure 6.13 and Figure 6.14 is a 22-year-old male speaker who has a British English dialect. Figures 6.11 and 6.13 show the F0 from original speech (shown as log(F0)). Figures 6.12 and 6.14 show the quantized F0 as it was provided to the system during training. Each quantization bin is labeled with the learned VQ F0 code.

To demonstrate the content of the F0 codebooks, the unvoiced regions were not masked in Figures 6.12 and 6.14 in order to observe which VQ F0 codes tend to represent unvoiced/silence regions. The VQ codes appear to learn the voice and unvoiced region boundaries well. From the plots of natural log(F0), we observe that the log(F0) is overall lower for the male compared to the female speaker. The gender appears to be reflected in differences for the VQ codes as well (indicated by codes 8 and 5 for the female, and code 3 for the male). Further analysis will attempt to quantify characteristic patterns in the F0 sequences, including aligning the VQ F0 codes with the original quantized bins (see Section 6.4.3.3).

[3] `https://github.com/google/REAPER`

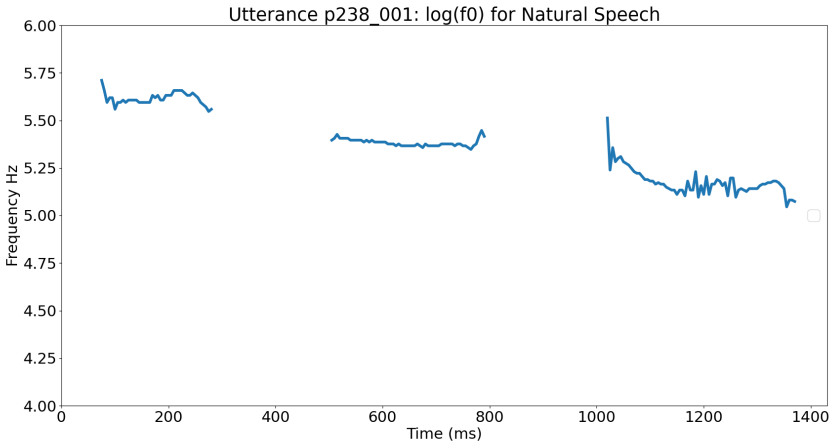


Figure 6.11: Female speaker p238. The F0 extracted from natural/original speech using the reaper tool. This is the log(f0) and unvoiced regions are masked.

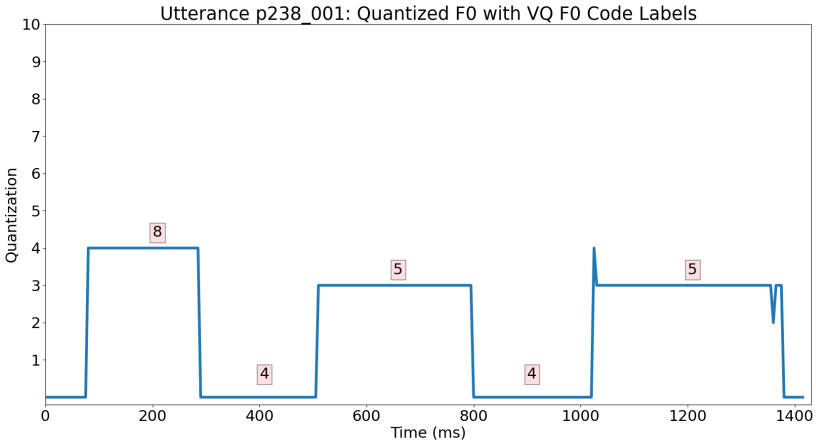


Figure 6.12: Female speaker p238. The quantized F0 is shown and each bin is labeled with the corresponding VQ F0 code. The unvoiced regions are not masked in this view. The VQ F0 codes appear to distinguish between voiced and unvoiced regions. For VQ codes, it appears that VQ code 4 corresponds to the unvoiced regions.

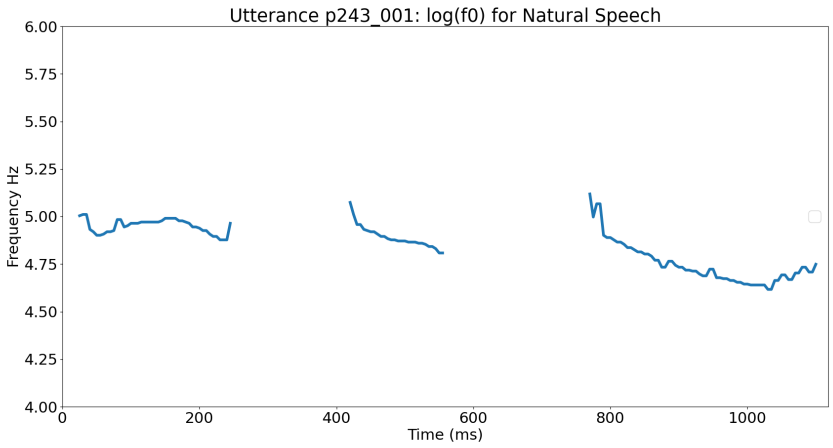


Figure 6.13: Male speaker p243. The F0 extracted from natural/original speech using the reaper tool. This is the log(f0) and unvoiced regions are masked

.

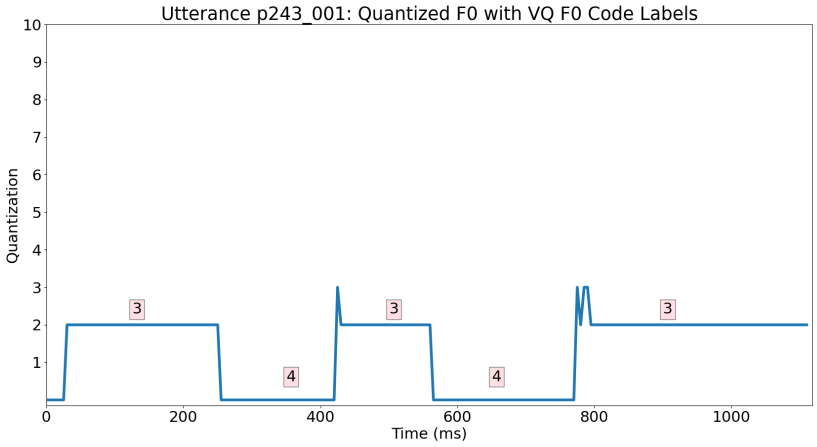


Figure 6.14: Male speaker p243. The quantized F0 is shown and each bin is labeled with the corresponding VQ F0 code. The unvoiced regions are not masked in this view. The VQ F0 codes appear to distinguish between voiced and unvoiced regions. For VQ codes, it appears that VQ code 4 corresponds to the unvoiced regions.

6.3.5 Summary

The analysis presented has so far confirmed that VQ phone codes do not follow statistical distributions that are similar to natural language, specifically for words and phones. In the case of system +**Adversarial**, the rank-ordered frequency distributions are very flat and this indicates that VQ phone codes occur at similar frequencies when representing content information. Flat distributions were not observed for the other systems. There is not a one-to-one mapping between speaker identity labels and VQ speaker codes and this is true for system +**Adversarial** as well as +**Speaker**/**F0**. The VQ F0 codes appear to accurately model voiced/unvoiced regions for male and female speakers, and this is partly due to the consistency of representing unvoiced/silence regions by a single VQ code.

6.4 Codebook Analysis Using Language Modeling

In this section, the analysis involves drawing from concepts that are part of language modeling techniques. In particular, three language modeling techniques are explored: perplexity, conditional probabilities, and distribution similarities. Language modeling in general can be done with various levels of linguistic units. In natural language processing, language modeling can be used to examine word or letter distributions. In theory, English has 26^2 possible pairs of letter combinations, since each letter could combine with each other letter to form a pair. In practice, however, some pairs are much more likely to occur than others in the English language. For example, in the English language the letter 'q' is commonly followed immediately by the letter 'u', but not the letter 'z'. By applying this concept to data and calculating the conditional probability of letters following one another, a language model can be created [Nadkarni et al., 2011]. This same concept of language modeling can be applied to sequences of letters, phones, words, VQ phone codes, or VQ F0 codes.

The values that are represented in a language model are entirely dependent on the data that is used for modeling. For example, if the language model is created on one body of text but applied to an unseen body of text, then the language model may potentially be mismatched to the unseen text. Different bodies of text utilize words at different frequencies, and in many cases they use different types of words altogether [Stamatatos et al., 2000]. Consider that a document about bird migration uses different words than a document about finance. The mismatch between the vocabulary causes language models to degrade in predictable ways, namely that it is relatively easy to detect that a mismatch has occurred.

Perplexity can be calculated when applying a language model to new data. The perplexity roughly represents how well the language model fits to new data [Chen and Goodman, 1999]. In fact, the new data is referred to as the test set T in the definition of perplexity (PP) provided in Equation 6.1

$$PP_T(P_M) = \frac{1}{(\prod_{i=1}^{t} P_M(w_i|w_1...w_{i-1}))^{\frac{1}{t}}} \tag{6.1}$$

where T is the test set (a text document) with words $T = \{w_1...w_t\}$, P_M is computed by the language model for (next word w|history h). The language model may be estimated on an initial text that is potentially very different from the test set. In that case, the perplexity will be relatively high because of the mismatch. The analysis that will be presented in this section aims to examine the mismatch between a language model and data through a careful construction of two data subsets. In one data subset, the speakers have all said the same set of utterances. In another subset, different speakers have each said different utterances. In the next section, the dataset will be described in more detail.

The perplexity analysis will be shown in more details in Section 6.4.2. Additional language modeling techniques include conditional probabilities in Section 6.4.3 to measure the most probable outcome of an event (such as a phone or a word) occurring in sequence given some history. Finally in Section 6.4.4, subsets of the data will be analyzed according to the distributions of phone types per speaker to better understand the relationships between phone and speaker information.

6.4.1 Data Preparation

When speakers use the same words or phones in a sentence, it can be said that their language usage is similar and a language model could capture this similarity. For this analysis the data was selected in a way that controls which utterances are shared among a group of speakers and which utterances are not shared. All of the selected speakers and utterances that were used to create data subsets had been seen during system training (this is referred to as condition1 in Chapter 5). Two groups of speakers were selected and two groups of utterances were selected: Set1 and Set2. The utterances in Set1 consist of a set of sentences that are spoken by all speakers in both groups of speakers. The Set1 utterances are provided in Table 6.3. For Set2, each utterance is different for each speaker. In addition, the speakers are divided into GroupA and GroupB. The purpose of dividing speakers and utterances is to better understand the similarities and differences among speakers given the content. Each task in this analysis probes a slightly different aspect of the problem of disentanglement.

The main factor in deciding the speaker groups was based on utterance match and mismatch. However there were some additional considerations. The speakers in GroupA were purposefully selected to be more homogeneous for English dialect whereas the speakers in GroupB were purposefully selected to be more heterogeneous for English dialect. This decision was made to help create 'matched' and 'unmatched' speaker sets. In addition, the speaking rates were considered for each of the speakers and speakers who tended to speak very fast were omitted from the GroupA and GroupB. The speaking rates were considered because it can potentially affect how much information is contained within a particular VQ phone code. If an utterance is spoken very fast, then it becomes more difficult to associate a VQ phone code with a phone, for example. Also when examining similarities and differences across speakers, the goal is not to model speaking rate as a feature. By pre-selecting speakers who have similar speaking rates it reduces any possibility that speaking rate would be modeled unintentionally. The metadata and speaking rate for GroupA speakers is provided in Table 6.4 and for GroupB this is provided in Table 6.5. The metadata consists of age, gender and English dialect.

Table 6.3: Summary of utterances in Set1 that are shared across all speakers in GroupA and GroupB. This table shows the VCTK sentence identifier as well as the content of each utterance.

Sent ID	Content
008	These take the shape of a long round arch, with its path high above, and its two ends apparently beyond the horizon.
009	There is, according to legend, a boiling pot of gold at one end.
013	Some have accepted it as a miracle without physical explanation.
015	The Greeks used to imagine that it was a sign from the gods to foretell war or heavy rain.
016	The Norsemen considered the rainbow as a bridge over which the gods passed from earth to their home in the sky.
018	Aristotle thought that the rainbow was caused by reflection of the sun's rays by the rain.
019	Since then physicists have found that it is not reflection, but refraction by the raindrops which causes the rainbows.
020	Many complicated ideas about the rainbow have been formed.
023	If the red of the second bow falls upon the green of the first, the result is to give a bow with an abnormally wide yellow band, since red and green light when mixed form yellow.
024	This is a very common type of bow, one showing mainly red and yellow, with little or no green or blue.

Table 6.4: Summary of speakers belonging to GroupA (16 speakers) and their respective metadata (age, gender, English dialect) as well as their average speaking rate (words per second).

VCTK Speaker	Gender	Age	English Dialect	Average Speaking Rate
p226	M	22	English	2.3
p227	M	38	English	2.6
p228	F	22	English	2.5
p229	F	23	English	3.1
p231	F	23	English	3.4
p232	M	23	English	3.2
p233	F	23	English	2.7
p234	F	22	Scottish	2.9
p237	M	22	Scottish	2.4
p238	F	22	Northern Irish	2.6
p239	F	22	English	3.0
p240	F	21	English	2.9
p243	M	22	English	2.8
p244	F	22	English	2.7
p245	M	25	Irish	2.4
p246	M	22	Scottish	2.6

Table 6.5: Summary of speakers belonging to GroupB (16 speakers) and their respective metadata (age, gender, English dialect) as well as their average speaking rate (words per second).

VCTK Speaker	Gender	Age	English Dialect	Average Speaking Rate
p248	F	23	Indian	2.8
p249	F	22	Scottish	2.6
p250	F	22	English	3.4
p251	M	26	Indian	2.8
p252	M	22	Scottish	2.1
p253	F	22	Welsh	3.0
p254	M	21	English	3.2
p255	M	19	Scottish	2.8
p257	F	24	English	3.5
p258	M	22	English	3.0
p259	M	23	English	2.6
p261	F	26	Northern Irish	3.1
p262	F	23	Scottish	2.8
p263	M	22	Scottish	3.2
p264	F	23	Scottish	2.5
p265	F	23	Scottish	2.2

6.4.2 Perplexity

One way to understand the sequences of phone codes is through the lens of language modeling, treating each unigram, bigram, and trigram of VQ codes in the same way that a sequence of words are treated in a sentence. In this section, the matched and unmatched speaker and utterance groups will be utilized to model phone code sequences using language modeling. As mentioned earlier, language models of English can be used to understand which letter is most likely to follow another letter in a word, or which word is most likely to follow another word in a sentence. By modeling sequences of VQ phone codes with language modeling, it is possible to assess whether the VQ phone codes may be speaker-dependent. The two groups of speakers (Group A and Group B) as well as the two sets of sentences (Set 1 and Set 2) have properties such as overlapping content that allow some predictions to be made regarding language modeling.

The construction of a language model also requires a smoothing technique in order to account for any n-grams that occur in the testing data, but which were not seen during training. The Lidstone smoothing method was selected because it designates a very small probability to unseen n-grams making it more responsive for smaller sized datasets such as the subsets used for this analysis [Chen and Goodman, 1999]. The smoothing parameter used with Lidstone was set to $\alpha = 0.5$. This value was selected after searching within the values $1.0, 0.5, 0.05, 0.005$ and there were no large changes in perplexity for the smaller values. Without smoothing, the perplexity could potentially be mathematically undefined or infinite due to the mismatch of vocabulary between the language model training data and the testing data. The language models and Lidstone smoothing were implemented using the Python NLTK library [Bird et al., 2009].

Language models were created on data from the speakers in GroupA and the content from Set1. In total, 15 language models were created using n-grams in the range of $[1, 2, 3]$ for words, phones and VQ phone codes from each of the three systems. As a test and sanity check, language models were first created using the words of the utterances and then a separate model was created using the phones of the utterances (the phones came from Montreal forced alignments). These two different types of language models were further explored using different n-grams (unigrams, bigrams, and trigrams). The GroupA/Set1 language model was then evaluated on four different test sets: GroupA/Set1 (for reference), GroupB/Set1, GroupA/Set2, and GroupB/Set2. The results are reported in Table 6.6 (bold values indicate the lowest perplexity for a given linguistic unit).

The expected result for words is that the perplexity for GroupA/Set1 and GroupB/Set1 would be lower than the other test sets. Because GroupA/Set1 and GroupB/Set1 contain the same sentences. The perplexity values in Table 6.6 should only be compared in relative terms. For both words and phones in Table 6.6, the perplexity decreases when the n-gram size increases. This is because the larger n-grams are modeling more context and providing a better language model. The perplexity values for GroupA/Set2 and GroupB/Set2 are higher indicating that the model is not a good fit. This makes sense because the sentences in Set2 are different from the sentences in Set1, and in some cases they share few words in common.

The expected result for phones is similar to the expected result for words. While the perplexity is relatively lower for larger n-gram sizes than for unigrams in the matched GroupB/Set1 condition, the perplexity remains high for unmatched conditions. Overall, the results from words and phones confirm that the content is having a greater affect on language modeling than the speakers.

Table 6.6: Summary of perplexity values for language modeling of words, phones, and VQ phone codes using language models based on unigrams, bigrams and trigrams. The language model was constructed from speaker Group A and utterances Set 1. That language model was then evaluated to the other speaker groups and utterances sets (including GroupA/Set1 for reference). The fitness of the language model was evaluated with measures of perplexity. Bold values indicate the lowest perplexity for a given linguistic unit.

Linguistic Unit	n-gram size	GroupA Set1	GroupB Set1	GroupA Set2	GroupB Set2
words	1	89	89	1523	1666
	2	6	6	156	157
	3	**4**	**4**	**134**	**135**
phones	1	33	33	39	41
	2	9	9	47	48
	3	**4**	**4**	**42**	**42**
VQ-VAE codes	1	82	82	83	83
	2	**44**	**44**	**47**	**47**
	3	52	52	55	55
+**Adversarial** codes	1	157	157	163	160
	2	**91**	**91**	**93**	**95**
	3	132	131	132	135
+**Speaker/F0** codes	1	95	95	99	99
	2	**53**	**53**	**54**	**53**
	3	68	68	69	69

Further in Table 6.6, results for language models estimated on sequences of VQ phone codes from each system variant: **VQ-VAE**, +**Adversarial**, and +**Speaker/F0**. Unlike the perplexity values for words and phones, there is not a clear trend of relative perplexity values across the different test sets for speaker and content. That is an unexpected finding and it suggests that the VQ phone codes are not especially consistent regardless if the content is the same or different. However, the perplexity does decrease by nearly half when comparing the unigrams with bigrams. The perplexity increases when comparing bigrams to trigrams. This suggests that bigrams are modeling context better than unigrams or trigrams. All three system variants (**VQ-VAE**, +**Adversarial**, +**Speaker/F0**) appear to have similar perplexity values in all cases and therefore this analysis does not distinguish one system variant over another. One potential interpretation of these results is that if the speaker and phone information was more disentangled then the perplexity values for test data GroupB/Set1 would be lower than for the other test sets.

6.4.3 Conditional Probabilities

Building from the ideas of frequency and perplexity that were introduced in Section 6.3 and Section 6.4.2, this section explores an approach to codebook analysis that measures how likely it is that a given code will co-occur with phones, speakers, or F0 quantization bins. In this section, conditional probabilities are calculated for the purpose of quantifying associations between learned codebook representations and features of the training data. For phones, this associates the phones from Montreal forced alignments with VQ phone codes. For speaker, this associates the speaker identity labels from the VCTK dataset with the speaker VQ codes. And for F0, this associates the quantization bin (from the pre-processing in Chapter 5) with VQ F0 codes.

Conditional probabilities can be used to measure the most likely outcome of an event

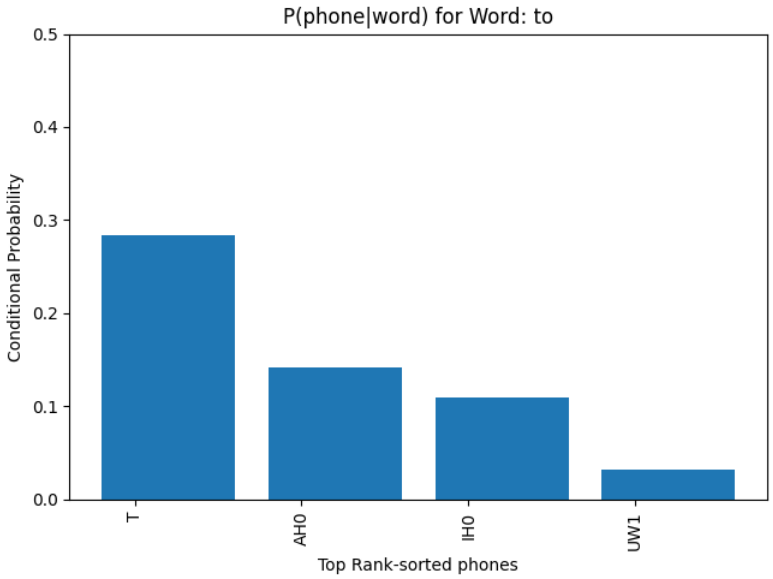


Figure 6.15: Conditional probability of phones given a word "to". Four phones were identified that correspond to occurrences of the word "to" in the subset of data. The phone /T/ is most probable. For the vowels there are three potential phones with the most probable phone being /AH0/ and least likely being /UW1/.

depending on another variable, such as how likely it is that a particular phone will occur given a word. The calculation for conditional probability is described in Equation 6.2. From this equation, the probability of an event occurring given another event can be determined by counting the number of times each type of event occurs in the dataset. The term A refers to the *event* A and it may reflect the occurrence of a word, while the term B refers to the *event* B and it may reflect the occurrence of a phone. The relationship is not symmetrical.

$$P(A|B) = \frac{P(A \text{ given } B)}{P(B)} \tag{6.2}$$

While the conditional probabilities shown in Figure 6.15 and Figure 6.16 suggest a very strong relationship between the phone /T/ and the word "to", there may not be such a strong relationship between the phone /AH0/ and the word "to"because /AHO/ is used often in other contexts. An example of the conditional probability of phones given the word "to" is shown in Figure 6.15. This figure indicates that the phone /T/ is the most probable phone given the word "to". The conditional probability can also be computed in the other direction to obtain the probability of the word given the phone. An example of the conditional probability of words given the phone /T/ is shown in Figure 6.16.

6.4.3.1 Phone Analysis

The data subset used for this analysis consists of GroupA speakers and Set1 utterances. This combination of speakers and utterances was chosen to ensure that there are multiple examples of similar content that is spoken by different speakers. The result is multiple instances of the same phone in context. If the speaker and phone information are perfectly disentangled, then the VQ phone codes will consistently be associated with a particular phone. However, finding a mapping

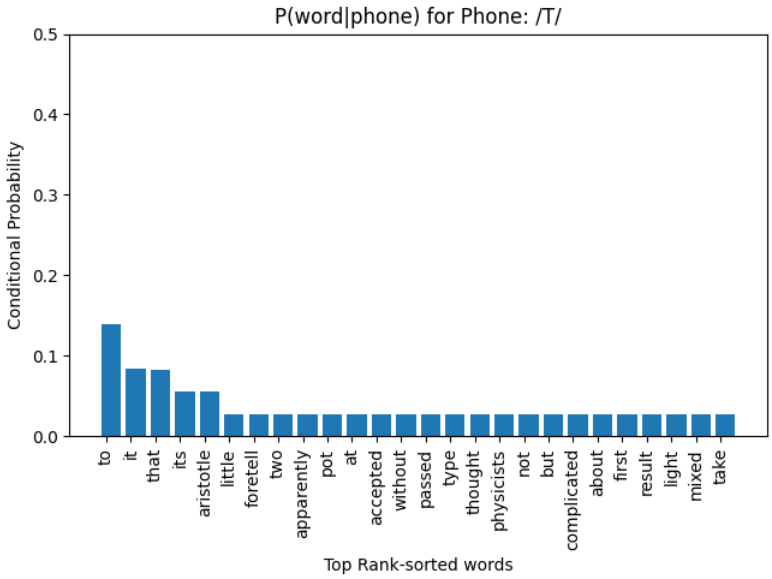
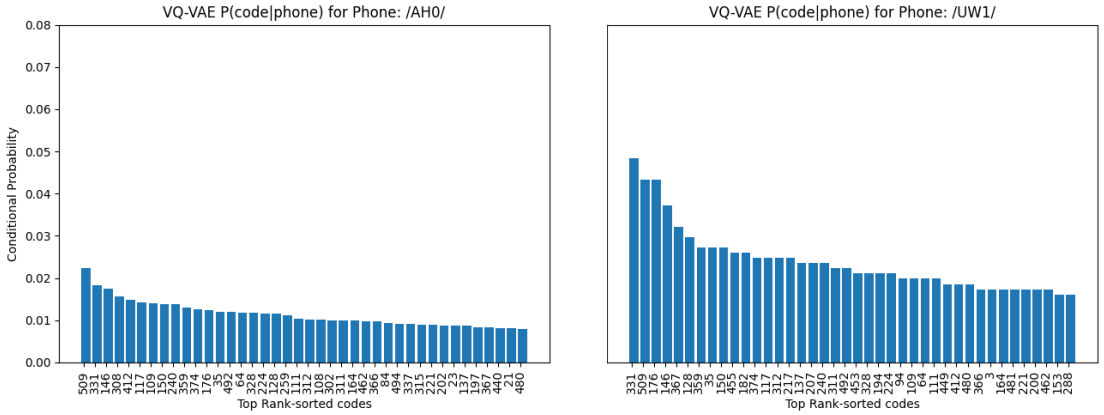


Figure 6.16: Conditional probability of words given a phone /T/. In total, 25 words were identified that correspond to occurrences of the phone /T/. The word "to" is most probable, followed by the word "it" and "that". Words such as "complicated" and "light" are less likely in this subset of the data.

between VQ phone codes and phones is a necessary but insufficient condition for disentanglement. Disentanglement would also require some evidence that VQ phone code sequences are not also correlated with speaker identity (such as the analysis that will be explored in Section 6.4.4). To analyze the VQ phone code sequences, they are used to measure the most probable code and phone as a conditional probability using Equation 6.2. The analysis was run in both directions, first treating the phone as *event A* and the VQ phone code as *event B*, then again treating the VQ phone code as *event A* and the phone as *event B*.

Along with the conditional probability, the code and phone frequency will also be taken into consideration in this analysis. Using the earlier example of the high frequency function word "to" from Figure 6.15, this section will examine conditional probability of a high-frequency phone and a low-frequency phone. The conditional probability will help to determine the most probable VQ phone code given a particular phone. From Figure 6.15, it can be seen that given the word "to" the phone /AH0/ is more likely to occur and /UW1/ is less likely to occur. The following analysis will examine conditional probabilities involving these two phones (/AH0/ and /UW1/) using VQ phone codes from each of the three system variants (**VQ-VAE**, +**Adversarial**, and +**Speaker/F0**). The purpose of this examination is to understand if high-frequency phones also correspond to high-frequency VQ phone codes. If the VQ phone codes are modeling language based only on content information (and disregarding speaker information) then there should be some similarities between high/low frequency phones and high/low frequency VQ codes.

The conditional probability distributions over VQ phone codes for each of the three system variants are shown in Figure 6.17, Figure 6.18, and Figure 6.19 for the phones /AH0/ and /UW1/. In each system variant, it can be seen that the distribution of conditional probabilities relating to the low frequency phone /UW1/ is different compared to the distribution relating to the high frequency phone /AH0/. Across all three systems, the distributions for /AH0/ are

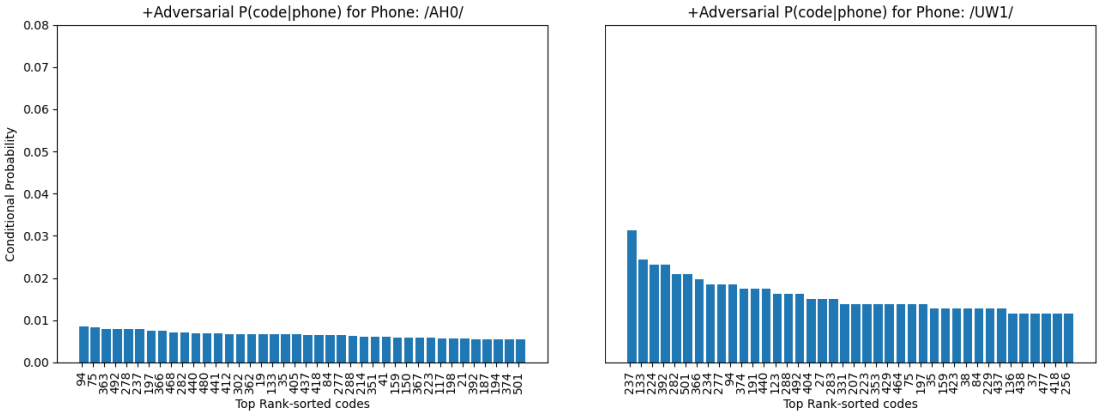


(a) (b)

Figure 6.17: Conditional probability distributions for the VQ phone codes from system **VQ-VAE**, given the phone: /AH0/ or /UW1/. The distribution for the phone /AH0/ is more flat compared to that of /UW1/. The most probable VQ phone codes are 509 and 331 for both phones.

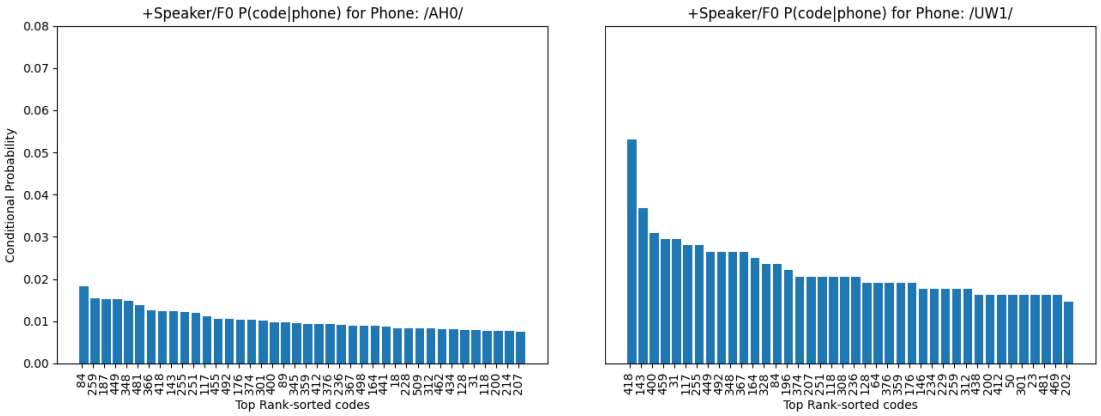
generally more flat compared to /UW1/. Furthermore, the **VQ-VAE** system in Figure 6.17 indicates that for both phones, the VQ codes 509 and 331 are the most probable. This suggests that the **VQ-VAE** system variant may not be modeling phones because two different VQ phone codes are highly likely to occur for each phone. A deeper inspection of the VQ phone codebook vectors in Section 6.5 can provide more insight about the relationship between the 509 and 331 VQ phone codes for that system.

Recall that from Chapter 5, the best-performing overall system variant was +**Adversarial** across multiple measures of speech synthesis quality. The observable flatness of conditional probabilities indicated by the high probability phone /AH0/ in Figure 6.18 may provide insight as to why +**Adversarial** performed better. It is possible that some of the the learned VQ phone codes were repeated regularly throughout all sequences regardless of content. If that is the case, then perhaps the repetition behaved in a manner that enabled the decoder to learn to produce more consistent speech. It is possible that repetition of codes enabled +**Adversarial** to generalize better to unseen content and speakers during the evaluation of speech synthesis in Chapter 5. An aspect of this will be investigated later in Section 6.4.4, where VQ phone code distributions will be compared across speakers.



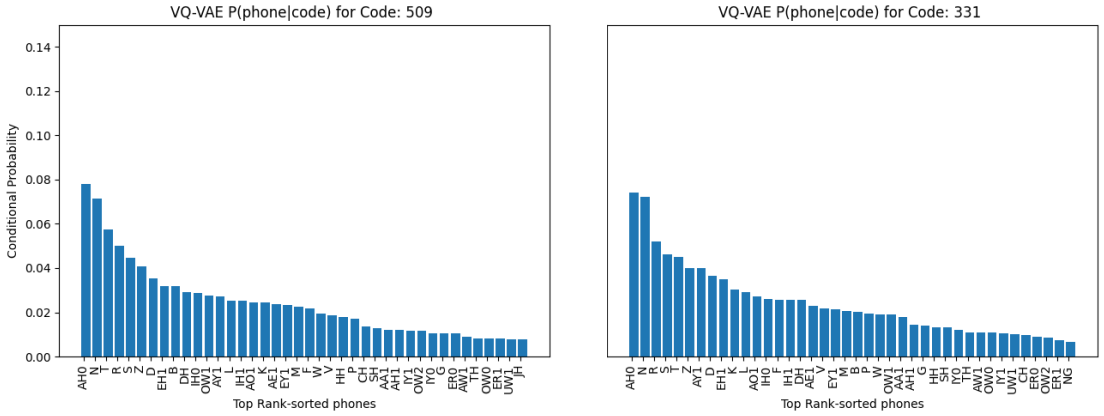
(a) (b)

Figure 6.18: Conditional probability distributions for the VQ phone codes from system +**Adversarial**, given the phone: /AH0/ or /UW1/. The distribution for the phone /AH0/ is more flat compared to that of /UW1/. The most probable VQ phone codes are different for each of the two phones.



(a) (b)

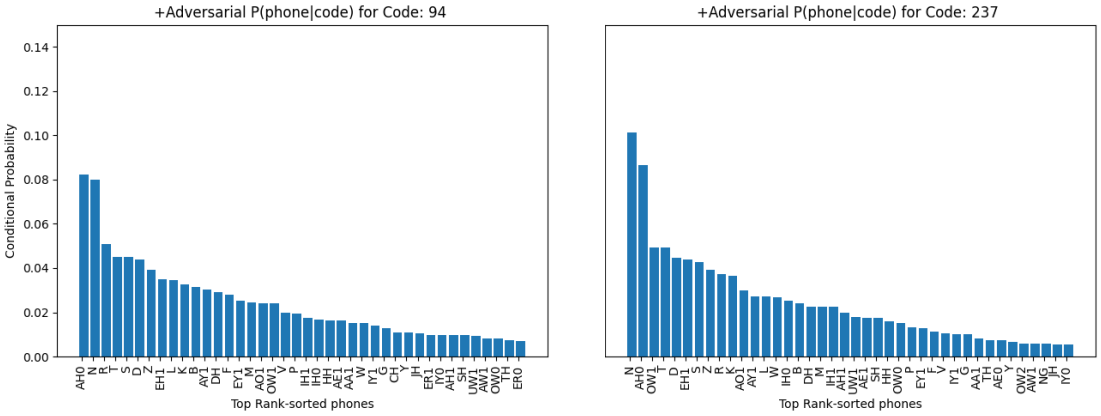
Figure 6.19: Conditional probability distributions for the VQ phone codes from system +**Speaker/F0**, given the phone: /AH0/ or /UW1/. The distribution for the phone /AH0/ is more flat compared to that of /UW1/. The most probable VQ phone codes are different for each of the two phones.



(a) (b)

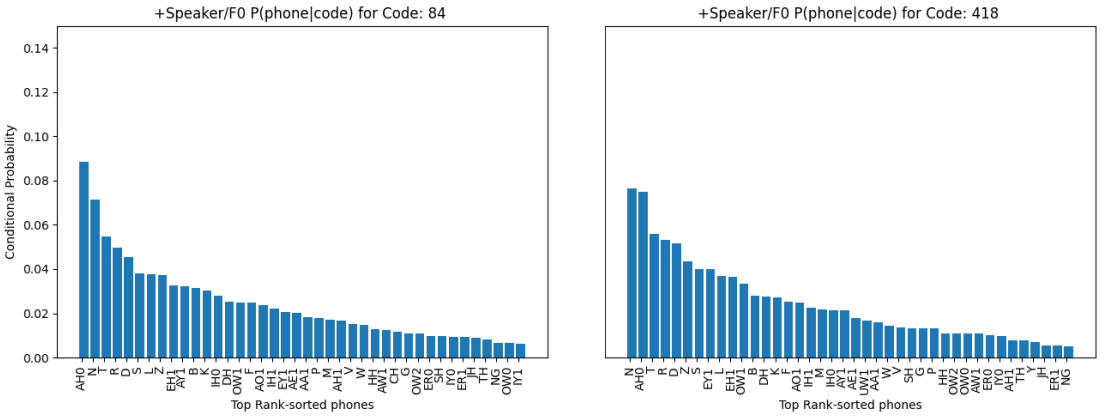
Figure 6.20: Conditional probability distributions for VQ phone codes from system **VQ-VAE**, given the code: 509 or 331. Code 509 was most probable given the phone /AH0/, and code 331 was most probable given the phone /UW1/.

Next, the conditional probability will be examined in the reverse direction to understand the conditional probability distributions over phones as shown in Figures 6.20, Figure 6.21, and Figure 6.22. For each system, the corresponding top-ranking VQ phone code was used from each of the two phones /AH0/ and /UW1/. In all three system variants, the same set of phones /AH0/, /N/, /R/, /T/, and /S/ are the most probable phone given that code, except for +**Adversarial** with VQ phone code 237 in Figure 6.21(b). Unlike the conditional probabilities for VQ phone codes, none of the figures for phones have a flat shape. This suggests that the high frequency phones like /AH0/, /N/, /R/, /T/, and /S/ might be associated with any of the VQ phone codes in this data subset, and this appears to be the case for all three system variants. This was an unexpected finding, especially seeing little difference among the system variants. It could be the result of how self-supervised learning works, or a result of the codebook parameters such as the initial codebook size. The distributions in Figures 6.20 and Figures 6.22 appear very similar, compared to the distributions reported in Figure 6.21. This is another example of the +**Adversarial** system standing apart from the other two systems.



(a) (b)

Figure 6.21: Conditional probability distributions for VQ phone codes from system +**Adversarial**, given the code: 94 or 237. Code 94 was the most probable code given the phone /AH0/ and code 237 was the most probable code given the phone /UW1/.



(a) (b)

Figure 6.22: Conditional probability distributions for VQ phone codes from system +**Speaker/F0**, given the code: 84 or 418. Code 84 was most probable given the phone /AH0/ and code 481 was most probable given the phone /UW1/.

Finally in this analysis there was an investigation about the relationship between high/low frequency phones and high/low frequency VQ phone codes, based on the question: *Are the most probable phones associated with the most probable VQ phone codes and vice versa?* Table 6.7 shows the most probable VQ phone code for the phones /AH0/ and /UW1/. These two phones /AH0/ and /UW1/ have very different probabilities of occurring: $p(AH0) = 0.104$, and $p(UW1) = 0.005$. One interpretation of the results in Table 6.7 is that high-probability phones are correlated with high-probability VQ phone codes, but low-probability phones are also correlated with high-probability VQ phone codes. This is an unexpected finding. Building from the earlier conditional probabilities outcomes, it seems that two statements could be made:

1. High frequency phones can be associated with any VQ phone code.
2. Low frequency phones can be associated with high frequency VQ phone codes.

Table 6.7: Most probable VQ phone code given the phones /AH0/ and /UW1/ for each system.

	/AH0/		/UW1/	
System	p(code)	Code	p(code)	Code
VQ-VAE	0.029	509	0.025	331
+Adversarial	0.011	94	0.009	237
+Speaker/F0	0.022	84	0.017	418

While this finding helps describe the relationship between phones and VQ phone codes, it does make it more difficult to claim that the VQ phone codebooks are interpretable. In fact the evidence so far points to the fact that the VQ phone codebooks are not interpretable because it is difficult to pinpoint a mapping between phones and VQ phone codes. In Table 6.8, the least probable VQ phone code is shown for each system for, and the most probable phone given the VQ phone code. Note that in Table 6.8 the code 477 from **VQ-VAE** is different than the code 477 from **+Speaker/F0** because the codes came from two completely separate phone codebooks. Two systems share /N/ as one of the the most probable phones in these conditions. The VQ phone codes that have been identified in this section will be further discussed in terms of the distances between VQ phone vectors in Section 6.5.

Table 6.8: Least probable VQ phone code for each system and the most probable phones given the VQ phone code.

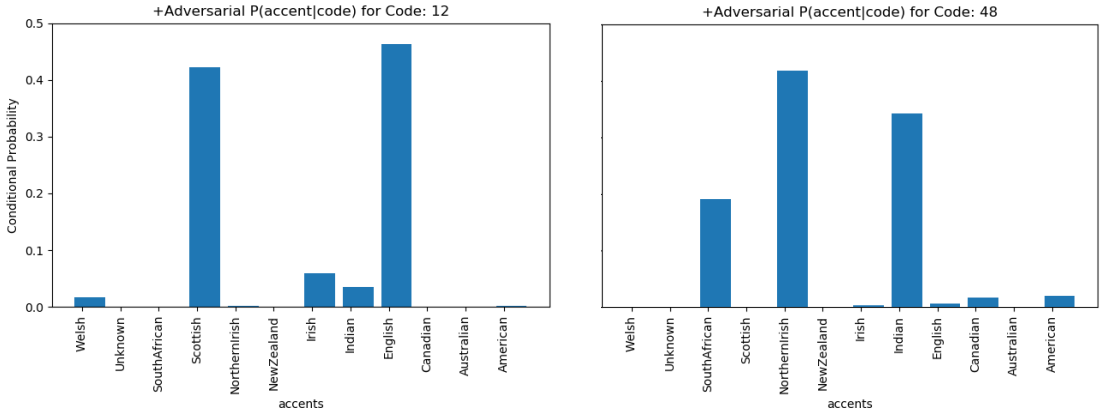
	Lowest		Most probable
System	p(code)	Code	Phones
VQ-VAE	0.002	477	/N/, /AH0/
+Adversarial	0.002	82	/N/, /K/
+Speaker/F0	0.001	477	/S/, /T/

6.4.3.2 Speaker Analysis

The speaker analysis in this section will use the speakers from GroupA and utterances from Set1 (similar to the phone analysis of the previous section). The objective of this analysis is to use conditional probability modeling to determine if speakers who are mapped into the same VQ speaker code tend to share any properties. The speaker properties were obtained through the metadata provided with the VCTK dataset. This section focuses specifically on accent. Two system variants are explored for the speaker analysis +**Adversarial** and +**Speaker/F0**. The **VQ-VAE** system was omitted because the speaker representations in that system are one-hot encodings and the system did not utilize a speaker codebook.

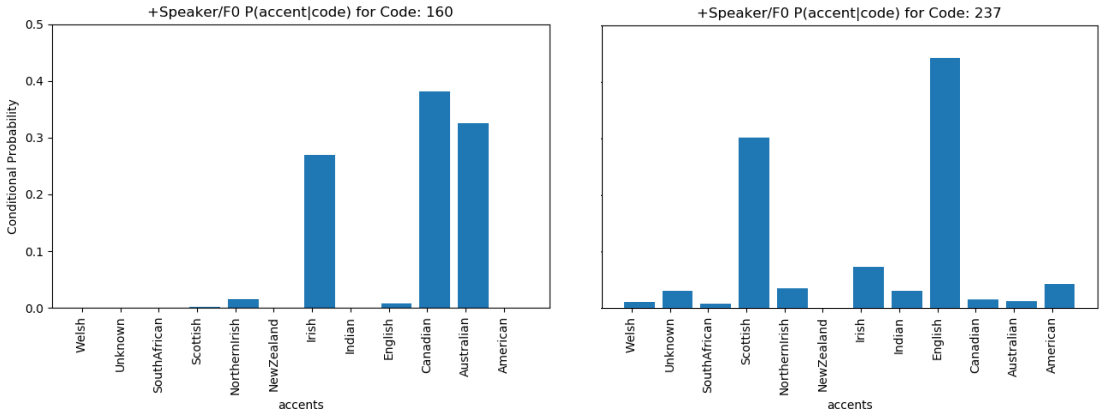
Figure 6.23 and Figure 6.24 show the likely accents given a VQ speaker code for each system variant. Within a particular system, there is little overlap of the accent distributions. This suggests that the VQ speaker codes are representing different types of speakers. This is an interesting finding in light of the one-to-many mapping that was uncovered earlier in Section 6.3.3 (Table 6.2). So although there is not a one-to-one mapping of speakers to VQ speaker codes in any system, the conditional probabilities suggest that the VQ speaker codes are representing information about accent.

The finding of meaningful VQ speaker codes associated with accent is further strengthened from the results in Figure 6.25 and Figure 6.26 which both show very little overlap of VQ speaker codes given an accent. There were no noticeable differences between the +**Adversarial** and +**Speaker/F0** systems. From these findings, further analysis in Section 6.5 will explore whether VQ speaker codes with similar accents are also represented by VQ speaker vectors that have high similarity.



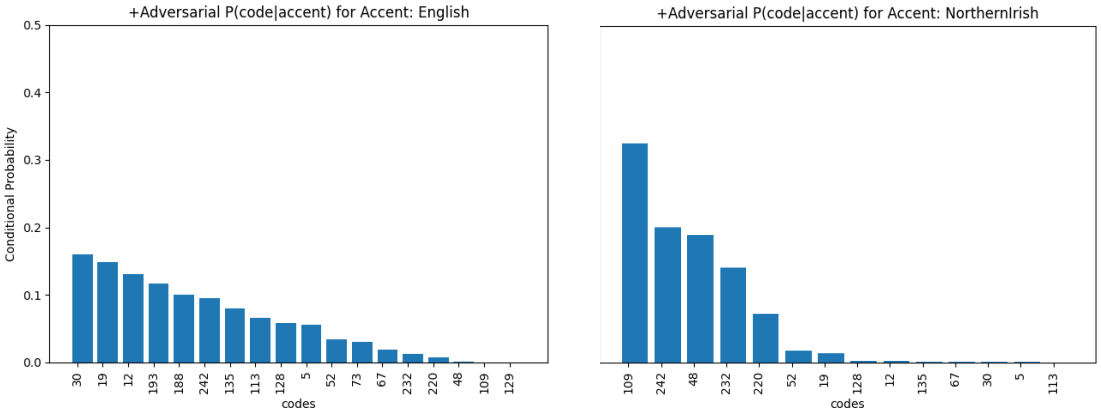
(a) (b)

Figure 6.23: Distributions of accent given two different VQ speaker codes in system +**Adversarial**. There is relatively little overlap among the accents for the two VQ speaker codes, suggesting that the VQ speaker codes are representing different accents.



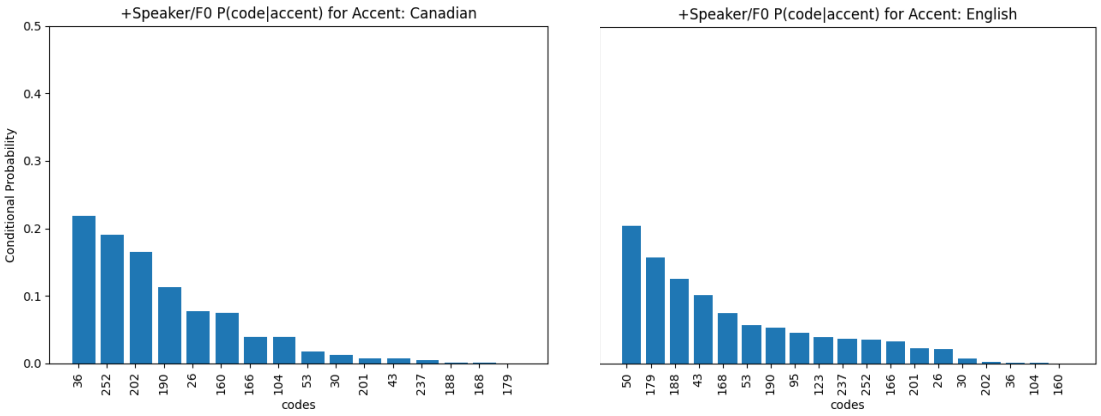
(a) (b)

Figure 6.24: Distributions of accent given two different VQ speaker codes in system +**Speaker/F0**. There is relatively little overlap among the accents for the two VQ speaker codes, suggesting that the VQ speaker codes are representing different accents.



(a) (b)

Figure 6.25: The likely VQ speaker codes given the accents of English and Northern Irish for system +**Adversarial**. The distributions over the VQ speaker codes suggest little overlap between English and Northern Irish accents.



(a) (b)

Figure 6.26: The likely VQ speaker codes given the accents of English and Northern Irish for system +**Speaker/F0**. The distributions over the VQ speaker codes suggest little overlap between English and Canadian accents.

6.4.3.3 F0 Analysis

The data that will be used for conditional probability modeling of VQ F0 codes for system +**Speaker**/**F0** includes data from both speaker groups (GroupA and GroupB) as well as both sets of utterances (Set1 and Set2). The decision to use all of the speakers and utterances combined was in order to maximize the amount of data for analysis. While the analysis in this section does not explicitly address disentanglement, it does attempt to uncover whether the VQ F0 codes are interpretable and whether the codes are predictive of speaker gender.

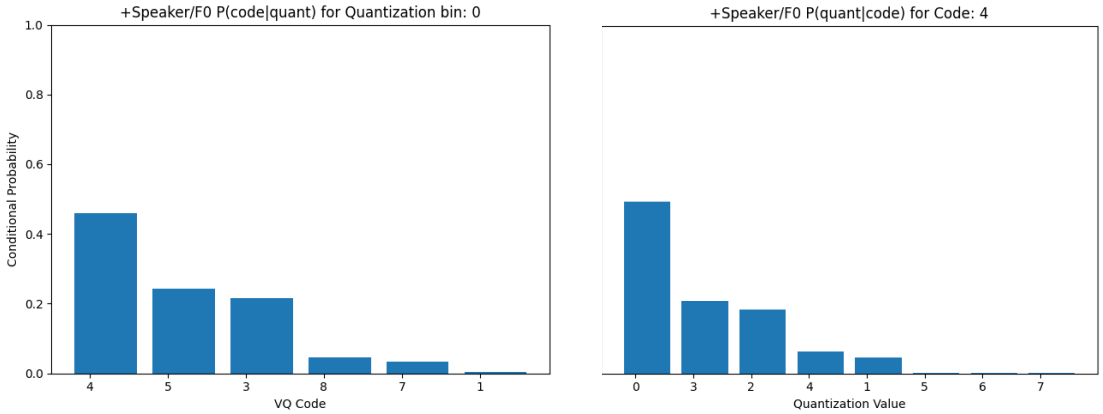
Two renditions of F0 are used in this analysis: quantized bins (referred to as *quant*) and VQ F0 codes (referred to as *codes*). One of the challenges of analyzing or comparing quantization bins and VQ F0 codes is that they follow different frame rates. The quant code frame rate was $fr = 5$ fps while the downsample rate for VQ F0 codes was $N/T = 340$ codes/second. The first step is to align the VQ F0 codes to the quantization bins. The quantized bins are grouped according to values that repeat in sequence. For example, if the original quantization bins sequence was: $[1, 2, 2, 3, 1, 4, 4, 3]$ then the grouped quantization bins becomes: $[[1], [2, 2], [3], [1], [4, 4], [3]]$. Using the grouped quantization bins as a reference along with the duration of each bin (5ms), the start time and end time of each group is calculated. Those start and end times are then used to create similar groups of VQ F0 codes, resulting in time-aligned groupings. From there, the most frequent code in each group is determined, separately for the VQ codes and the quantization bins. The result is two arrays of integers that are aligned, as shown in Figure 6.27.

$$Quant[0, 2, 0, 2, 1, 2, 1, 0, 2, 0, 2, 1, 2, 0, 1, 2, 1, 2, 1, 0]$$
$$VQ[4, 3, 4, 3, 3, 3, 3, 4, 3, 3, 3, 7, 7, 3, 4, 4, 7, 7, 7, 4]$$

Figure 6.27: Final result of aligning the VQ F0 codes to the quantization bins, even though they had two different sample rates.

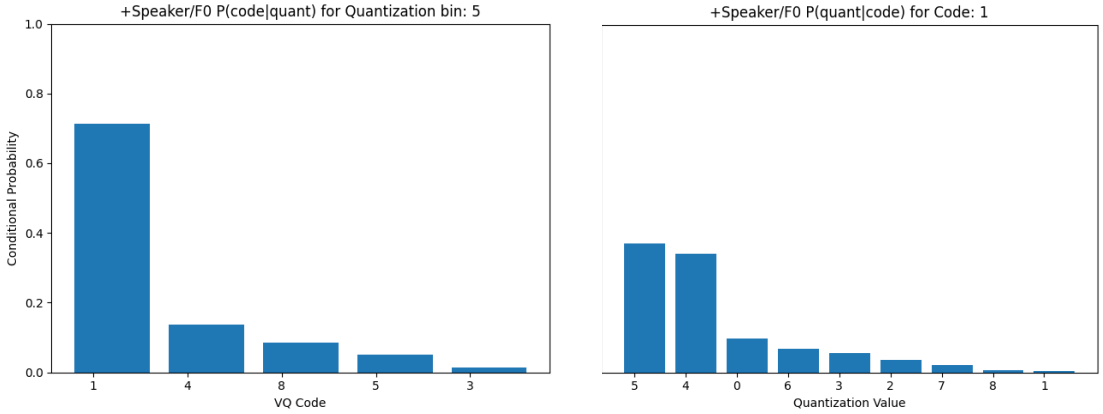
The array of *quant* values in Figure 6.27 shows that the F0 value alternates between regions of bin #0 (this is unvoiced/silence in the audio file). Overall the values are interpretable for the *quant* array because the quantization bins reflect real F0 ranges (i.e., the quantized bins are monotonic). The speaker that Figure 6.27 is based on never produces an F0 value that is high enough to utilize quantization bin #3. On the other hand, it appears that the VQ code of 4 (and occasionally also 3) corresponds to the unvoiced/silence regions of the audio because they are aligned with the zero-regions of the *quant* array (i.e., the VQ F0 codes are not monotonic). From here, the task is to describe the relationship between the two sets of values quantitatively using conditional probability. This analysis will help to understand whether particular VQ F0 codes are likely to correspond to particular quantization bins.

Figure 6.28 shows the conditional probability for quantization bin #0 (unvoiced/silence) and the VQ F0 code 4. The VQ code 4 is highly likely to occur given the quantization bin #0, and the quantization bin #0 is highly likely to occur given VQ code 4. It is reasonable to draw the conclusion that quantization bin #0 and VQ code 4 are related and also that since bin #0 corresponds to unvoiced/silence, the VQ code of 4 may also correspond to unvoiced/silence. A similar finding is shown in Figure 6.29, for a voiced region and the quantization bin #5 with VQ code 1. Additional analysis with different bins and codes indicated that quantization bins #5,



(a) (b)

Figure 6.28: Conditional probability of quantization bin #0 (unvoiced/silence) given the VQ F0 code 4. The F0 code 4 appears to be correlated with unvoiced/silence regions.



(a) (b)

Figure 6.29: Conditional probability of quantization bin #5 (voiced regions) given the VQ F0 code 1. The F0 code 1 appears to be correlated with quantization bin #5.

#6, #7, and #8 all tend to correspond to VQ code 1. As Table 6.9(b) shows, quantization bins #5, #6, #7, and #8 are lower frequency compared to other quantization bins. And VQ code 1 is lower frequency compared to other VQ codes. This relationship will be explored further in Section 6.5 to better understand if the VQ codes of 4 and 1 have a large distance between them. If the VQ vectors are interpretable, then a relatively large distance between the vectors for VQ code 4 and VQ code 1 would be expected.

Additional analysis was done to find out if particular VQ F0 codes correspond to gender. Figure 6.30 shows how likely it is that the VQ F0 code 3 corresponds to male speakers (and lower F0 range), while the VQ F0 code 5 corresponds to female speakers (and higher F0 range). This finding suggests that the VQ F0 codes may also be useful for controlling prosody though control over high and low F0 when synthesizing speech with WaveRNN. Of course quantized F0 bins also correspond to gender, but it is not clear how to use the quantized bins for F0 control during speech synthesis in a VQ-VAE paradigm.

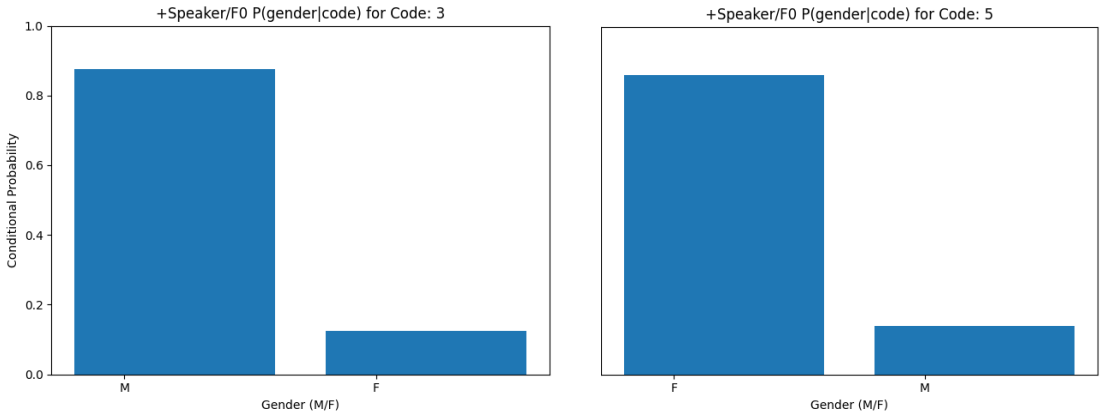
Table 6.9: Frequencies of the VQ F0 codes (a) and the frequencies of the quantization bins (b). In both groups, some VQ codes or quantization bins have very high or very low frequency.

VQ Code	Frequency
1	270
2	0
3	6746
4	8318
5	8859
6	0
7	1189
8	2674
9	0
10	0

(a) Frequencies of VQ F0 codes

Quant Bin#	Frequency
0	8936
1	1857
2	7598
3	7314
4	2163
5	140
6	31
7	14
8	3
9	0

(b) Frequencies of Quantization Bins



(a) (b)

Figure 6.30: Conditional probability of gender given VQ F0 code. The code 3 is highly likely for males and code 5 is highly likely for females.

6.4.4 Speaker/Phone Separation

When considering disentanglement, one important aspect is to try and uncover whether the speaker information is tied to the phone information. The VQ phone code usage can be tallied as a probability distribution for each individual speaker. Then a speaker-to-speaker comparison can be made based on the probability distributions of the VQ phone codes. The motivation behind this analysis is to learn if speakers share similar or different distributions of VQ phone codes. The data used in this analysis is speaker GroupA or GroupB and utterances from Set1 or Set2. The following combinations are considered:

- GroupA/Set1 speakers say the same utterances
- GroupA/Set2 speakers say different utterances
- GroupB/Set1 speakers say the same utterances
- GroupB/Set2 speakers say different utterances

If there is good disentanglement between speaker and content information, then it is expected that the phone code probability distributions would be similar across speakers when the utterances are from Set1. Likewise if there is good disentanglement, then the phone code probability distribution would be dissimilar when the utterances are from Set2. Speakers from GroupA (Table 6.4) and GroupB (Table 6.5) will be compared because of their accent makeup, with GroupA being more homogeneous and GroupB being more heterogeneous. If the GroupB/Set1 has more accent dissimilarity than the GroupA/Set1, this is a reflection of the fact that GroupB speakers are more dissimilar. The analysis that will be presented in this section explores whether the accent dissimilarity is also reflected by the VQ phone codes.

The probability distributions over VQ phone codes will be compared on a speaker-to-speaker basis using the Kullback-Liebler Divergence (also known as KL divergence) [Joyce, 2011]. The KL Divergence is defined in Equation 6.3

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) log_2 \left(\frac{P(x)}{Q(x)} \right) \tag{6.3}$$

where $P(x)$ is a probability distribution over the variables $x \in X$ and $Q(x)$ is another probability distribution over the variables $x \in X$. In this case, the set X corresponds to either the words/phones or the VQ phone codes. An analysis using words/phones is presented first as a sanity-check and to demonstrate expected results for natural language, then the VQ phone codes are analyzed. Since some speakers utilize a different set of words/phones or VQ codes than the speaker being compared, smoothing was necessary to avoid errors from dividing by zero (the value would be undefined). Lidstone smoothing [Chen and Goodman, 1999] was used and the parameter was set to $\alpha = 0.000001$. This value was chosen after visually inspecting the order of magnitude for the probability distributions, but no other values were tried. The KL divergence measure between the two probability distributions is not symmetrical. To deal with the lack of symmetry in this analysis, the KL divergence was calculated in both directions and an average was taken. That means $D_{KL}(P||Q)$ was calculated as well as $D_{KL}(Q||P)$. This is widely known as "symmetric KL divergence". Comparing two different speakers may show a dissimilarity between the two speakers (reflected by a high KL-divergence) because their VQ code probability distributions are different.

The analysis technique presented here is conceptually similar to probing the VQ codebooks using data, wherein the data contains features that lead to expected results based on assumptions about phone/speaker disentanglement. The data was used to probe differences between speakers using the four types of speaker and utterances groupings that was introduced earlier in Section 6.4.1: GroupA/Set1, GroupA/Set2, GroupB/Set1, and GroupB/Set2. If the speaker/phone information has been disentangled then there would be more similarity among speakers in GroupA/Set1 and GroupB/Set1 compared to using utterances from Set2 because Set1 utterances are the same for all speakers and Set2 utterances are different for each speaker. In addition, there should be more similarity among speakers in GroupA/Set1 compared to speakers in GroupB/Set1 because the accents are more similar in GroupA than in GroupB. As with other analysis shown in this chapter, this analysis will first be performed using words and phone to establish some expected results that are easily interpretable before examining the VQ phone codes.

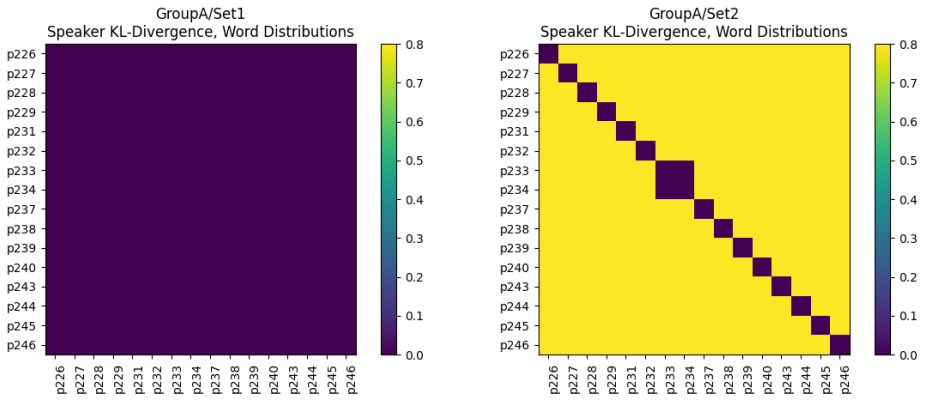


Figure 6.31: Per-speaker KL divergence for words in matched condition GroupA/Set1 (left) and unmatched condition GroupA/Set2 (right). The matched condition has very low KL divergence and the unmatched condition has very high KL divergence between all speaker pairs. The extremes observed in this figure are the expected result for words. Speaker p233 and p234 share the same sentences

Figure 6.31 shows the two scenarios of matched and unmatched content for speakers in GroupA. The matched content (left) has low divergence because the content words are the same for all speakers. The unmatched content (right) has high divergence because the content words are different for each speaker. The low KL divergence in GroupA/Set2 for speakers p233 and p234 is due to them sharing the same set of utterances despite efforts to ensure as little overlap as possible. The same pattern is observed in Figure 6.32 for phones. However, at the phone level, there are some slight differences in the matched condition which are likely due to individual variation in pronunciation or due to accent. Also the unmatched condition at the phone level generally displays lower KL divergence compared to the unmatched condition at the word level. The reason is because some phones are common across multiple different words and that creates redundancy within the phone probability distributions.

The probability distributions were also calculated for VQ phone codes for each of the speakers in GroupA, for both system variants: +**Adversarial** and +**Speaker/F0**. Note that **VQ-VAE** was omitted because it only contains a phone codebook and is not being assessed for disentanglement between speaker/phone representations. Looking at the plots in Figure 6.33

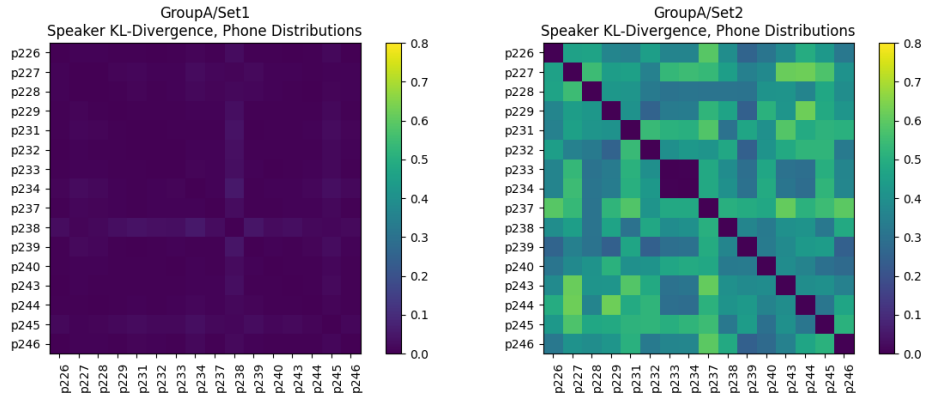


Figure 6.32: Per-speaker KL divergence for phones in matched condition GroupA/Set1 (left) and unmatched condition GroupA/Set2 (right). The matched condition has very low KL divergence and the unmatched condition has high KL divergence between all speaker pairs, though not as much divergence as for words.

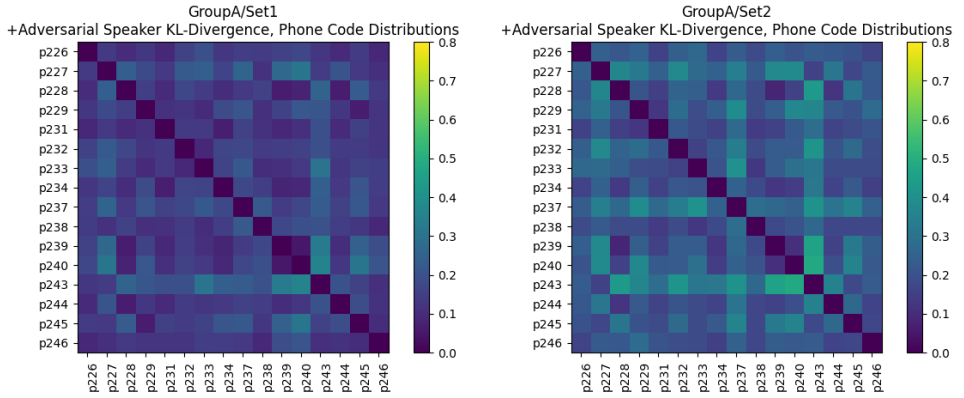


Figure 6.33: Per-speaker KL divergence for VQ phone codes from +**Adversarial** in matched condition GroupA/Set1 (left) and unmatched condition GroupA/Set2 (right).

and Figure 6.34, there is a similar trend that the unmatched condition (right side) generally has higher KL divergence compared to matched condition (left side). Overall the results from the matched condition indicate that there is some disentanglement between phone and speaker information occurring.

The +**Adversarial** system shown in Figure 6.33 appears to have a better result for the matched condition (low KL divergence) compared to the +**Speaker**/**F0** system in Figure 6.34. On the other hand, +**Speaker**/**F0** appears to have a better result for the unmatched condition as it shows more dissimilarity. The result for +**Adversarial** is encouraging because it appears to perform disentanglement better than +**Speaker**/**F0** based on the low KL divergence in matched conditions.

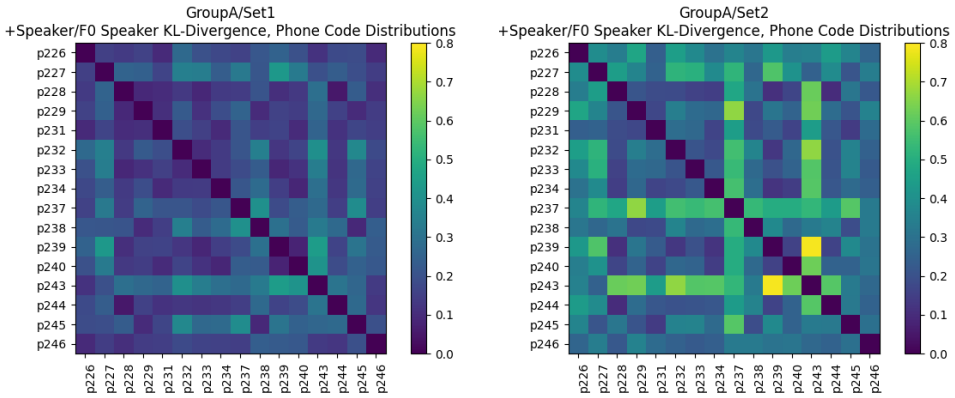


Figure 6.34: Per-speaker KL divergence for VQ phone codes from +**Speaker**/**F0** in matched condition GroupA/Set1 (left) and unmatched condition GroupA/Set2 (right).

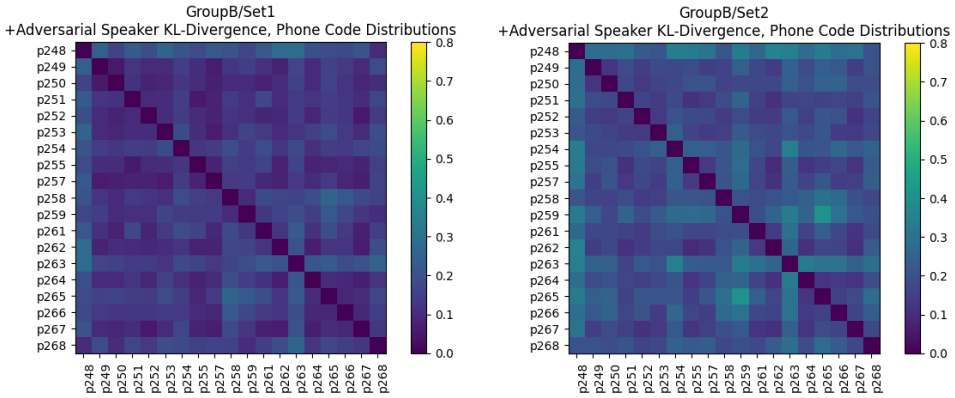


Figure 6.35: Per-speaker KL divergence for VQ phone codes from +**Adversarial** in matched condition GroupB/Set1 (left) and unmatched condition GroupB/Set2 (right).

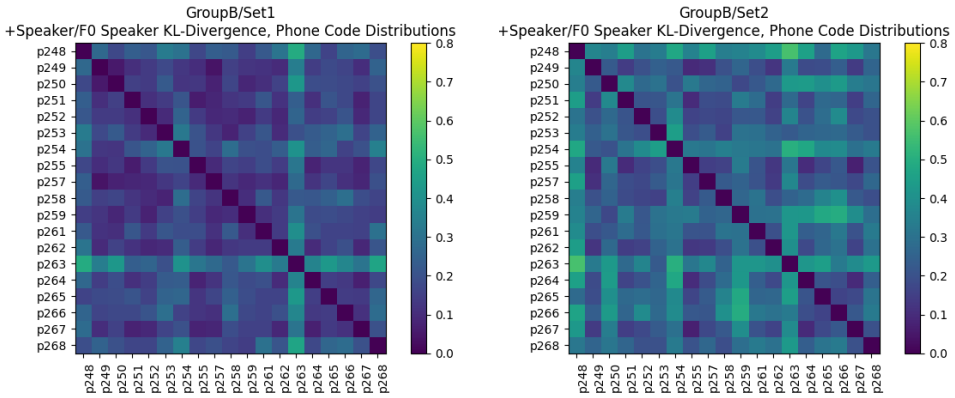


Figure 6.36: Per-speaker KL divergence for VQ phone codes from +**Speaker**/**F0** in matched condition GroupB/Set1 (left) and unmatched condition GroupB/Set2 (right).

Next, the two systems +**Adversarial** and +**Speaker/F0** were compared for utterances and speakers in GroupB/Set1 and GroupB/Set2. The results are shown in Figure 6.35 and Figure 6.36. The purpose of examining GroupB speakers is to understand if the VQ phone codes capture information about accent. The speakers from GroupB are more heterogeneous with regard to accent compared to the GroupA speakers. Compare Figure 6.33 with Figure 6.35 and observe that Figure 6.35 has lower KL divergence. Next, compare Figure 6.34 with Figure 6.36 and observe that Figure 6.36 also has lower KL divergence. The lowered KL divergence can be interpreted to mean that the phone content is well separated from speaker information even when the speakers are more heterogeneous, and this is true for both the matched and unmatched conditions. Overall, the +**Adversarial** system achieves lower KL divergence, and thus better phone/speaker disentanglement, compared to the +**Speaker/F0** system in this comparison.

6.4.5 Summary

The data probing tasks sought to measure and visually display similarities and differences among groups of speakers in matched and unmatched conditions. If the phone and speaker information was being fully disentangled, then the matched condition would indicate more similarity among speakers because they say the same sentences. It was found that the +**Adversarial** system generally has lower KL divergence among speakers in the matched condition compared to +**Speaker/F0**. Therefore it can be said that the +**Adversarial** system is performing better disentanglement of speaker/phone information relative to the other system. This finding also coincides with the performance evaluation from Chapter 5, which found that +**Adversarial** generated the highest quality speech across multiple metrics. Further analysis in Section 6.5 will explore similarities and differences among the VQ vectors.

6.5 Codebook Visualizations

The analysis that will be presented in this section is intended to supplement other findings from earlier in this chapter. The objective earlier in this chapter has been to explore whether the learned representations exhibit disentanglement. In particular, this chapter has been exploring whether representations model target speech features but do not also model non-target speech features. The codebooks for VQ phones, VQ speakers, and VQ F0 vectors will now each be visualized with several different methods including visualizing distance between vectors in a 2D space.

Until now, the analysis in this chapter has considered the VQ codes (or sequences of codes for phones/F0). At this point, instead of analyzing the codes, the 128-*dim* vectors will be analyzed. Recall that the codebooks correspond to a lookup table (Figure 6.1). Each type of codebook was set to be of a particular size before training. The phone codebook size was fixed at 512, the speaker codebook size was fixed at 256, and the F0 codebook size was fixed at 10. Each entry in a codebook a vector. For example, the phone codebook contains 512 vectors each of 128-*dim*. Likewise, the F0 codebook contains 10 vectors each of 128-*dim*.

6.5.1 Data Preparation

The data used in this analysis consists of vectors from the trained VQ codebooks. Table 6.10 shows each system and the number of codebook entries that were active during training. To determine which codebooks entries were active during training, all of the training data was passed through the trained model (as a separate inference step) to obtain the VQ codes associated with the training data. It can be seen from Table 6.10 that none of the system variants made use of the full quantity of possible codebook entries. The codebook usage, or lack thereof, will be discussed in more details when visualizing each codebook.

Table 6.10: How many codebook entries were active during training for each system and codebook type. System **VQ-VAE** only has a phone codebook, whereas +**Adversarial** has a speaker and phone codebook, and system +**Speaker/F0** has a speaker, phone, and F0 codebook.

System	Phone Codebook	Speaker Codebook	F0 Codebook
VQ-VAE	92	–	–
+**Adversarial**	170	18	–
+**Speaker/F0**	109	19	6

6.5.2 Phone Codebook

The full capacity of the phone codebooks was not active. For example, in the phone codebook for **VQ-VAE** there were 512 possible entries, and the training data activated only 92 of those entries. The other remaining 420 codes in that system have values initialized but were never used. For the training data, only 17% of the full codebook was utilized. In fact, a similarity matrix based on cosine similarity between codebook vectors reveals that many of the codebook vectors are similar, as seen in Figure 6.37. Cosine similarity values of $sim = 1.0$ indicate high similarity whereas $sim = 0$ indicates no similarity.

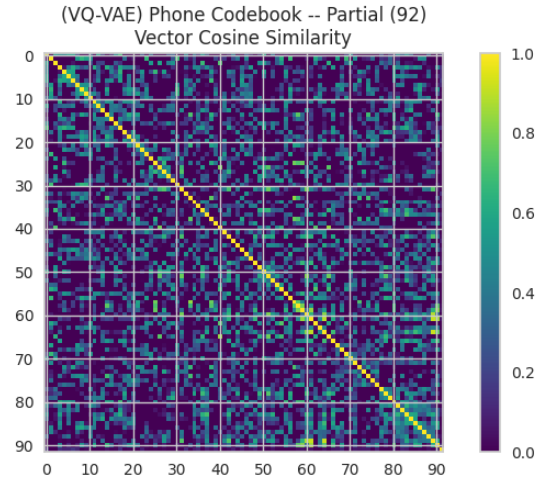


Figure 6.37: Cosine similarity between codebook vectors for **VQ-VAE** phone codebook. The codebook was partially active for the training data, so only 92 phone vectors are shown (only 17% of the codes were active).

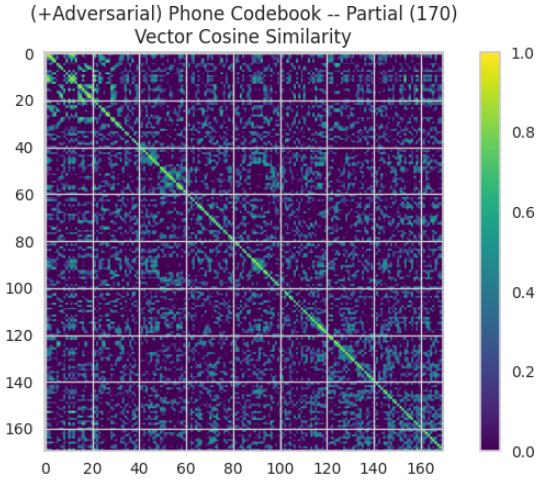


Figure 6.38: Cosine similarity between codebook vectors for +**Adversarial** phone codebook. The codebook was partially active for the training data, so only 170 phone vectors are shown (only 33% of the codes were active).

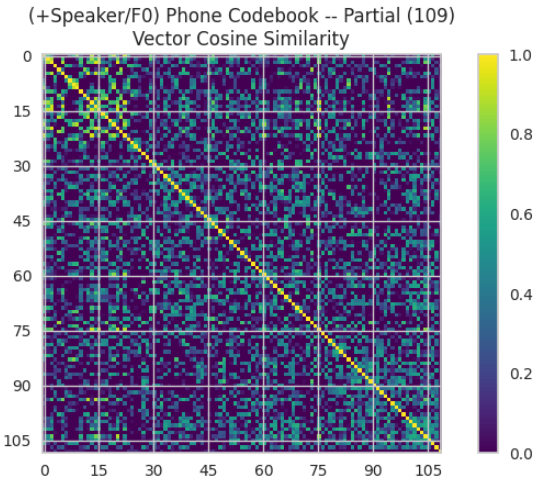


Figure 6.39: Cosine similarity between codebook vectors for +**Speaker**/**F0** phone codebook. The codebook was partially active for the training data, so only 109 phone vectors are shown (only 21% of the phone vectors were active).

The phone codebook in system +**Adversarial** utilized 170 of 512 codes for the training data (33%) in Figure 6.38 and the +**Speaker**/**F0** system utilized 109 of 512 codes (21%) shown in Figure 6.39. Given that system +**Adversarial** was one of the best-performing systems for speech synthesis quality (based on evaluation in Chapter 5), it is possible that the higher quality speech is a result of the +**Adversarial** system utilizing more codes relative to other system variants. In fact, one of the machine learning constraints on the +**Adversarial** system was to use adversarial layers (gradient reversal) on the phone encoder to encourage the phone codebook to ignore speaker-specific information.

An additional view of the **VQ-VAE** phone codebook is shown in Figure 6.40 based on the tSNE dimensionality reduction technique (t-distributed Stochastic Neighbor Embedding) from the Python library *sklearn*[4]. The tSNE technique reduces dimensionality from high dimensions (phone codebook vectors are 128-*dim*) into a lower-dimensional representation and 2D was selected for this analysis. The tSNE algorithm is highly sensitive to hyper-parameters such as number of iterations and perplexity. For that reason, values were explored in the range of 5-100 perplexity and iterations was set to maximum of 5,000. These values were recommended from the sklearn library[5]. The tSNE visualization shown in Figure 6.40 is based on $perplexity = 5$ and $iterations = 5,000$ for system **VQ-VAE**. The gray items represent vectors not active from training (not used) and the blue dots are vectors that were active for the training data in the final model (used). Interpreting the distances in a tSNE plot can be misleading, especially for interpreting distances using proportions. Unfortunately the visualization in Figure 6.40 does not provide information about similarities and differences between phone vectors. The only cluster that can be identified is the cluster showing codebook vectors that are used. The other

[4]https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
[5]https://distill.pub/2016/misread-tsne/

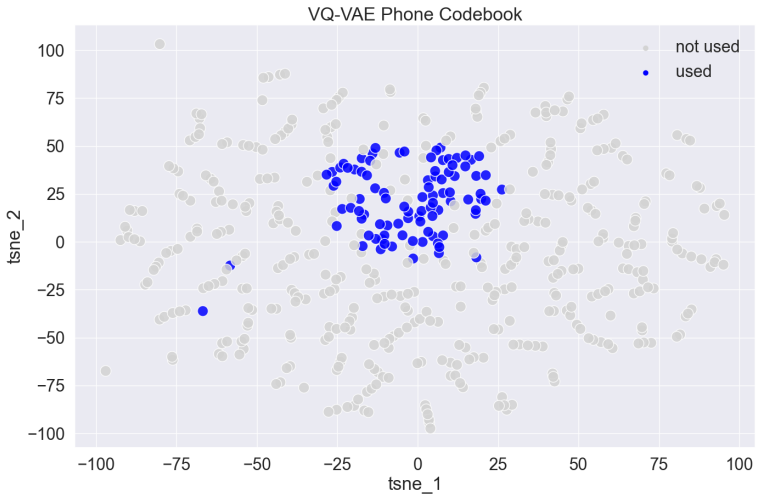


Figure 6.40: This is a tSNE plot showing the phone codebook vectors from system **VQ-VAE**. The pale gray dots represent vectors that were not used for training data whereas the dark blue dots represent the vectors that were used for training data. The distances reflected in the tSNE plot are not interpretable. The cluster represents codebook vectors used during training.

system variants have similar visualisations of the phone codebook, for systems +**Adversarial** and +**Speaker**/**F0**.

As the tSNE visualization from Figure 6.40 has not been informative, and was difficult to interpret therefore a different type of visualization will be done. The next visualization will consider only the codebook vectors that were used by the training data. The goal of the next visualisation is to explore unsupervised clustering of the VQ vectors. It is noted that the VQ-VAE architecture does create clusters during training, and the VQ vectors represent centroids of those clusters. However, k-means clustering was applied in this analysis in order to better understand the relationships among the VQ vectors. First the VQ vector dimensionality was reduced using principal components analysis (PCA) [Pearson, 1901] with default parameters[6] and reduced to 2 components. Then k-means clustering was performed using the implementation provided in Python sklearn[7]. Several values for k in k-means were explored and the highest possible k was chosen that did not produce any singleton clusters. The clustered VQ vectors are presented for each system in Figure 6.41, Figure 6.42, and Figure 6.43.

Figure 6.41 shows the clusters for system **VQ-VAE**. In total there were 15 clusters. The cluster membership was analyzed. The VQ codes 509 and 331 from earlier analysis on conditionals (Section 6.4.3.1) are somewhat close but they end up in separate clusters. Taking the four most probable VQ phone codes for each of the two phone plots (/AH0/ and /UW1/) in Figure 6.17 as [509, 331, 176, 146, 308], these VQ codes end up in clusters [7, 2, 9, 3, 9] respectively. These clusters [7, 2, 9 and 3] are adjacent in Figure 6.17, suggesting that the VQ vectors are close in distance. The least probable VQ code for **VQ-VAE** was code 477 and it ended up being clustered into cluster #14, which is centrally located relative to other clusters. So the VQ vector for the least probable code does not appear to be an outlier even though that might be expected.

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`
[7] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.k-means.html`

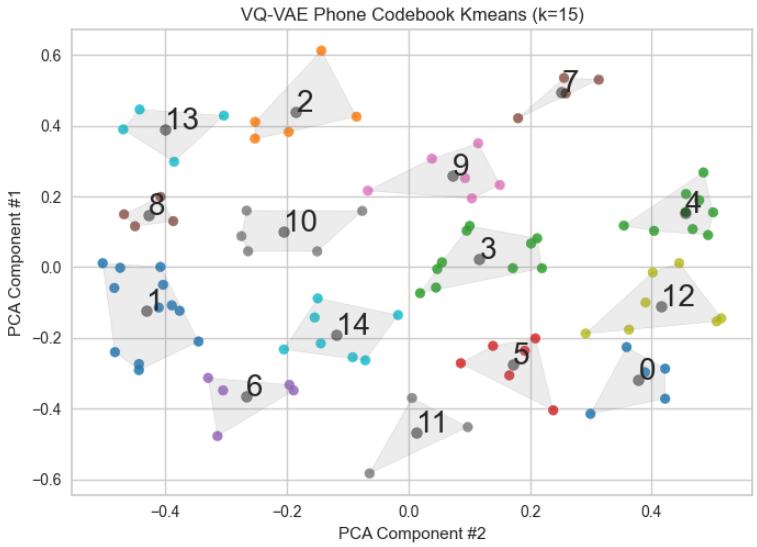


Figure 6.41: Result of PCA (2 components) followed by k-means clustering ($k = 15$) for the **VQ-VAE** phone codebook, for VQ vectors that were active for the training data. Cluster centroids are labeled with integers.

Figure 6.42 shows the clusters for system +**Adversarial**. In total there were 10 clusters. The cluster membership was analyzed. The VQ codes 94 and 82 from earlier analysis in Section 6.4.3.1 were examined. VQ code 94 had the highest probability for this system and it was placed into cluster #1. On the other hand, VQ code 82 had the lowest probability and it was placed into cluster #8. Cluster #1 and #8 are relatively far apart based on the first principal components (on the y-axis), meaning also that the least probable and most probable codes are far apart in this space. VQ code 237 was also examined as it corresponded to a high probability and it was also the most probable VQ code for the most probable phone /UW1/. VQ code 237 was found in cluster #9 and far from cluster #1 based on the second principal component (on the x-axis). There was no evidence of a consistent relationship between least probable phones and least probable VQ codes.

Figure 6.43 shows the clusters for system +**Speaker/F0**. In total there were 10 clusters. The cluster membership was analyzed. The VQ codes 477 and 82 from earlier analysis in Section 6.4.3.1 were examined. VQ code 477 had lower probability and was found in cluster #4. VQ code 84 was more probable and was found in cluster #15. Finally VQ code 418 was the most probable code given the phone /UW1/ and it was placed into cluster #16. The clusters (#4, #15, and #16) are not very far apart from each other in Figure 6.43. Again there was no evidence of a consistent relationship between rare phones and rare VQ codes.

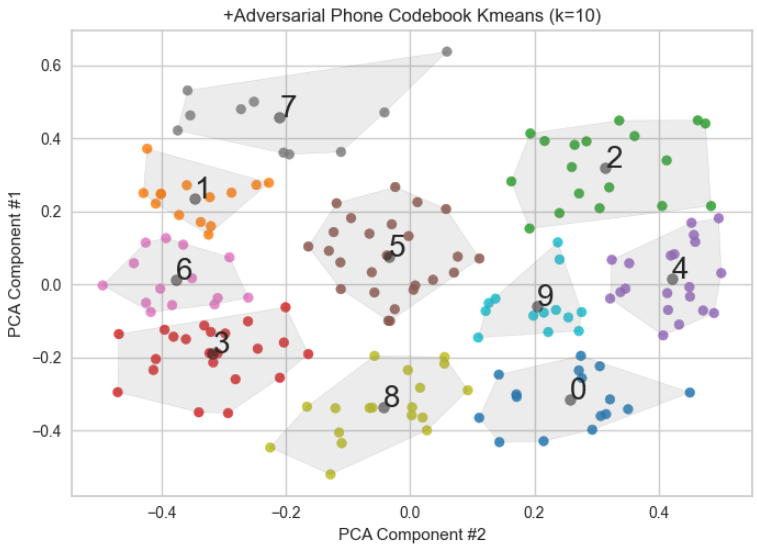


Figure 6.42: Result of PCA (2 components) followed by k-means clustering ($k = 10$) for the +**Adversarial** phone codebook, for VQ vectors that were active for the training data. Cluster centroids are labeled with integers.

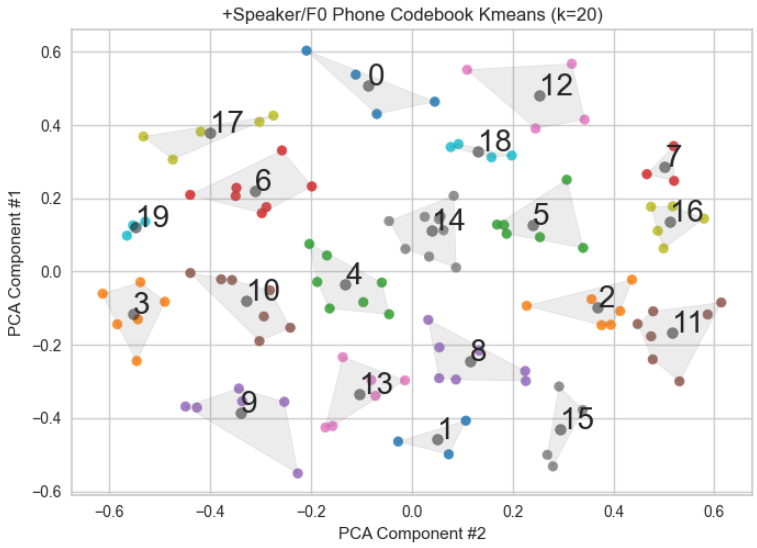


Figure 6.43: Result of PCA (2 components) followed by k-means clustering ($k = 20$) for the +**Speaker/F0** phone codebook, for VQ vectors that were active for the training data. Cluster centroids are labeled with integers.

6.5.3 Speaker Codebook

Similar to the VQ phone codebook visualization in the previous section, the speaker codebooks were analyzed for two system variants, +**Adversarial** and +**Speaker/F0**, using k-means clustering and PCA. First, Figure 6.44 and Figure 6.45 show the cosine similarity between each vector in the codebook for each system. Cosine similarity values of $sim = 1.0$ indicate high similarity between two vectors, whereas $sim = 0$ indicates no similarity. The full codebook contains many vectors that are similar because those vectors were not active during training and has similar values. When looking at only those codes that were active for the training data, it is clear that the speaker vectors for the +**Speaker/F0** system (Figure 6.45) were more diverse compared to the speaker vectors for the +**Adversarial** system (Figure 6.44). Recall from Section 6.3, the rank-sorted frequency distributions for VQ phone codes in the +**Adversarial** system was more flat compared to other system variants (see Figure 6.6). It is possible that the flat frequency distribution for VQ phone codes is related to the lower diversity of VQ speaker vectors. As for disentanglement, this relationship suggests that there +**Speaker/F0** is representing speaker information with speaker vectors better than +**Adversarial** and therefore has better disentanglement.

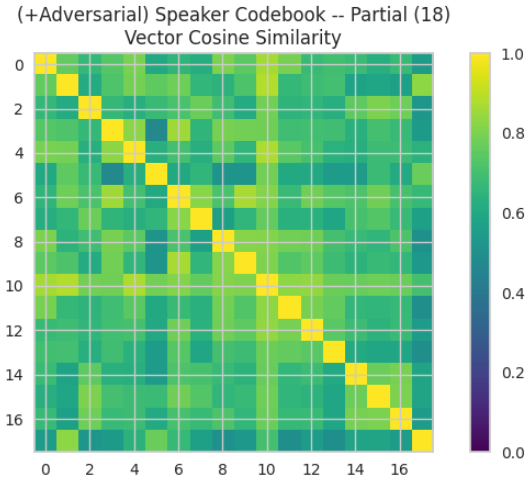


Figure 6.44: Cosine similarity between codebook vectors for +**Adversarial** speaker codebook. The codebook was partially active for the training data, so only 18 speaker vectors are shown (only 7% of the speaker vectors were active).

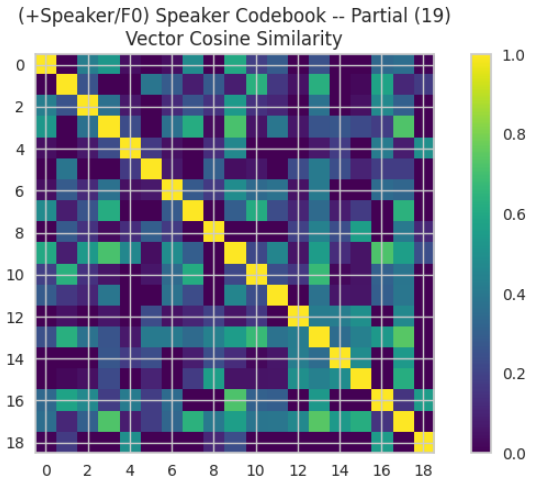


Figure 6.45: Cosine similarity between codebook vectors for +**Speaker**/**F0** speaker codebook. The codebook was partially active for the training data, so only 19 speaker vectors are shown (only 7% of the speaker vectors were active).

Figure 6.46 shows the result of applying PCA and k-means clustering to the speaker codebook vectors for system +**Adversarial**. The number of clusters k was selected such that there were no singleton clusters resulting from k-means. The resulting clusters ($k = 4$) also seem to correlate with the findings for accent earlier in Section 6.4.3.2 for the VQ speaker codes 12 and 48 (c.f. Figure 6.23). Code 12 tended to correspond to the accents: English, Scottish, and Irish. Code 48 tended to correspond to the accents: Northern Irish, Indian, and South African. Based on the clustering shown in Figure 6.46, these VQ codes occur in separate clusters (code 12 is in cluster #0, and code 48 is in cluster#2). This indicates that the speaker codes for +**Adversarial** are modeling speaker accent to some extent.

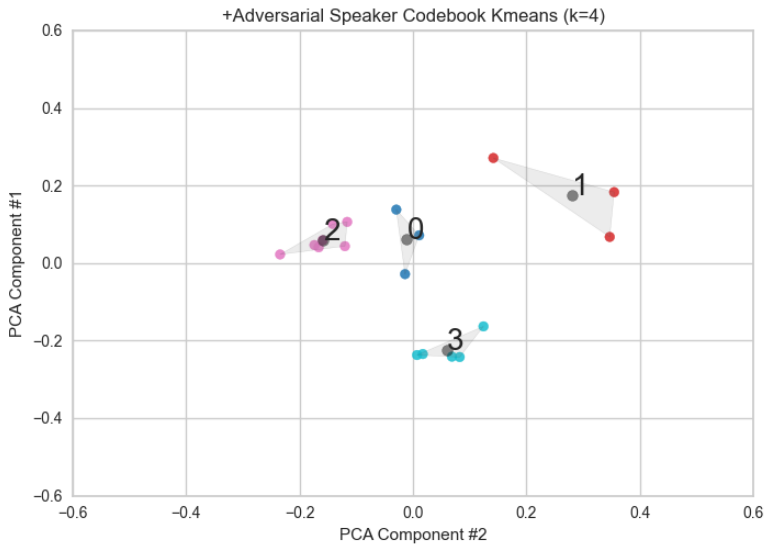


Figure 6.46: Result of PCA followed by k-means clustering ($k = 4$) for the +**Adversarial** speaker codebook, for VQ vectors that were active for the training data. The average intra-cluster distance was $d_{intra} = 0.068$. Cluster centroids are labeled with integers.

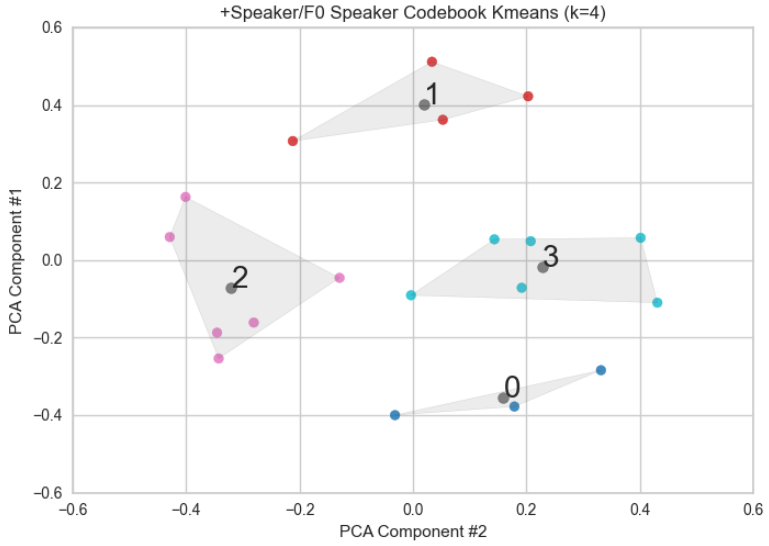


Figure 6.47: Result of PCA followed by k-means clustering ($k = 4$) for the +**Speaker**/**F0** speaker codebook, for VQ vectors that were active for the training data. The average intra-cluster distance was $d_{intra} = 0.15$. Cluster centroids are labeled with integers.

Figure 6.47 shows the result of applying PCA and k-means clustering to the speaker codebook vectors for system +**Speaker**/**F0**. The resulting clusters ($k = 4$) also seem to correlate with the findings for accent earlier in Section 6.4.3.2 for the VQ speaker codes 160 and 237 (c.f. Figure 6.24). Code 160 tended to correspond to the accents: Canadian, Australian, and Irish. Code 237 tended to correspond to the accents: English, Scottish, and Irish. Based on the clustering shown in Figure 6.47, these VQ codes occur in separate clusters (code 160 is in cluster #1, and code 237 is in cluster#2). This indicates that the speaker codes for +**Speaker**/**F0** are modeling speaker accent to some extent.

Comparing the clusters across the two systems in Figure 6.46 and Figure 6.47, it appears that the codes within clusters for +**Adversarial** are less spread out because the data points within each cluster are closer to the centroid compared to the data points in +**Speaker**/**F0**. On the other hand, the inter-cluster distances for +**Adversarial** appear smaller compared to the inter-cluster distances for +**Speaker**/**F0**. The space was not calibrated between the two systems, apart from setting the axes to be in the same range, and comparisons between the two clusterings are difficult. The intra-cluster distance was calculated between each data point and its assigned cluster centroid using Euclidean distance, and then the distances were averaged across all clusters for each system. It was found that the +**Adversarial** system had an average intra-cluster distance of $d_{intra} = 0.07$ whereas the +**Speaker**/**F0** system had an average intra-cluster distance of $d_{intra} = 0.15$. This could be an indication that the clusters for +**Adversarial** are modeling speaker groups, such as accent, better than the other system. Given that +**Adversarial** was the best-performing system from Chapter 5, this result may explain why +**Adversarial** was better at generalizing to new unseen speakers. However, it's not clear if lower intra-cluster distance is preferable for a speaker codebook.

6.5.4 F0 Codebook

Figure 6.48 shows the results of PCA applied to the VQ F0 codebooks that were active for the training data. Cross-referencing the findings from Section 6.4.3.3, there are some notable relationships. First, recall from the conditional probability modeling of VQ F0 code and gender VQ code 3 was likely to correspond to male and VQ code 5 was likely to correspond to female (c.f. Figure 6.30). The vector representations of these two VQ codes appear quite far according to the first principal component (on the y-axis) of Figure 6.48. The vector corresponding to VQ code 4 is a clear outlier and in the earlier analysis using conditional probability, the VQ code 4 is likely to be associated with unvoiced/silence regions (c.f. Figure 6.28). From this analysis, the learned F0 codes are partially interpretable.

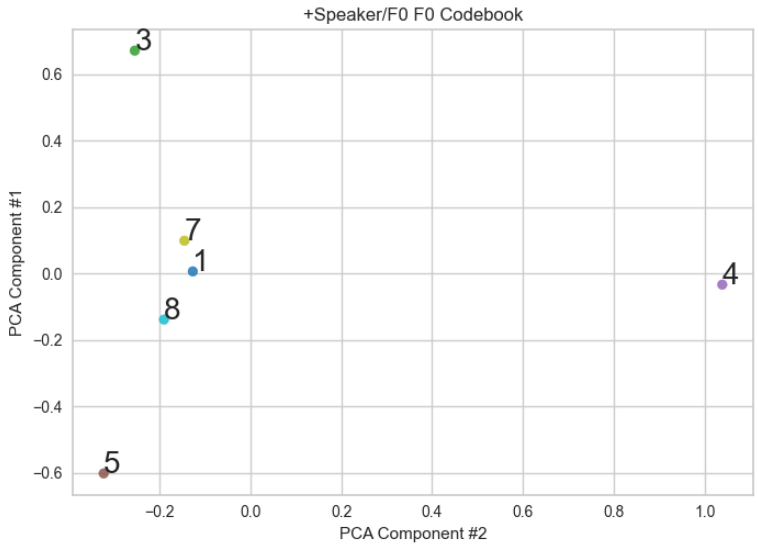


Figure 6.48: Result of PCA for the +**Speaker**/**F0** F0 codebook, for VQ vectors that were active for the training data. Note that VQ code 4 was likely to correspond to quantization bin for unvoiced/silence and in this figure it is the clear outlier. Clusters are singletons and labeled with integers.

6.5.5 Summary

The visualizations that were provided in this analysis examine similarities and differences between VQ vectors rather than codes. The vectors allowed for visualizations such as clustering, similarity matrices, and observing distance relationships in 2D. For all of the phone and speaker codebooks, most of the VQ codes are not active for the training data. While the unused VQ codes in each codebook and each system contained values that were initialized (i.e., not zero) the unused vectors within a codebook were very similar to each other. The inactive VQ codes had vectors of initialized values and were not useful to visualize or measure since the values in all inactive VQ vectors were similar.

The VQ phone codebooks were clustered with k-means, and it was found that the +**Adversarial** system showed some expected results for clustering. In particular the example VQ phone code with high probability appeared in a cluster that was very far from the example VQ phone code with low probability. However, the clusters do not seem to be particularly meaningful or interpretable. A cross-reference was made between the k-means clusters and the earlier conditional probabilities. In each system there were mixed results compared to expectations. For example, the VQ code with low probability for **VQ-VAE** was not an outlier as expected. In another example, three clusters were identified for +**Speaker**/**F0** that coincided with very likely codes as well as high and low probability codes and all three clusters were relatively close to each other instead of relatively far apart. Overall, the phone codebook visualization supports the finding that the phone codebooks are not interpretable – at least the VQ phone code categories do not map into phone categories.

The VQ speaker codebooks were also clustered with k-means. When the clusters were cross-referenced with findings from the earlier conditional probability analysis, it was found

that clusters appear to correspond with speaker accent. Both systems, +**Adversarial** and +**Speaker/F0**, exhibit some relationship with speaker vector clusters and groups of accents. The k-means clusters for +**Adversarial** have smaller intra-cluster distances compared to +**Speaker/F0**. This could potentially be evidence that +**Adversarial** is modeling distinct speaker groups better than +**Speaker/F0**. The speaker vectors in +**Adversarial** are more diverse overall than the speaker vectors from +**Speaker/F0** and this was evidenced by the cosine similarity heatmaps. It is possible that the diversity of the speaker codebook in +**Adversarial** is what allowed this system to generalize better to unseen speakers in the Chapter 5 evaluation.

The VQ F0 codebook from +**Speaker/F0** was visualized with PCA, but the codes were not clustered as there were only six VQ F0 codes that were "used" for the training data. The F0 codebook visualization was cross-referenced with the conditional probability analysis. The conditional probability analysis had indicated that a specific VQ code was highly likely to correspond to unvoiced/silence regions, and that same VQ code was found to be a clear outlier in the 2D visualization. In addition, particular codes were identified that were likely to correspond to male or female genders and these codes were also very far apart in terms of one of the principal components for PCA.

6.6 Additional Analysis of Multilingual Model

This section introduces an additional analysis of the multilingual system. At the end of Chapter 5, an additional model variant was introduced wherein the monolingual English model +**Adversarial** was adapted for multilingual speech synthesis by adding a one-hot language encoding vector for decoder global conditioning. The resulting multilingual system is +**Adversarial-L** since it is based on the original +**Adversarial** model but fine-tuned to multilingual data. The analysis presented here will compare the phone and speaker codebooks between the +**Adversarial** system and the +**Adversarial-L** system.

6.6.1 Data Preparation

The data used in this analysis was the SIWIS dataset that consisted of four languages: English, German, French, and Italian. The +**Adversarial-L** system had been trained on data in all four languages combined. Each speaker in the dataset is bilingual or trilingual. Because of this overlap, the amount of data per-language and per-speaker was slightly skewed. The same training set that was used for creating the +**Adversarial-L** system was used also in this analysis. As with the other systems used in this chapter, the data was used in a forward pass through the model (as a separate inference step after training) and the VQ codebooks for phone and speaker were each saved to files. The data that is analyzed in this chapter is based on the VQ phone codes and vectors that were obtained from the forward pass. Additionally, since comparisons are made in this section between the +**Adversarial-L** and +**Adversarial** systems, the VQ codes for English VCTK were also obtained for the entire English VCTK training data set.

6.6.2 Phone Codebook

The first task is to assess how the phone codebook changed between the +**Adversarial** system and the fine-tuned multilingual +**Adversarial-L** system. The VQ code frequency distributions

for English-only (from +**Adversarial**) were calculated over the VCTK training data. The VQ code frequency distributions for the multilingual data (from +**Adversarial-L**) were calculated over the SIWIS training data. Table 6.11 shows the relative frequency of VQ phone codes for the original +**Adversarial** model as well as the adapted +**Adversarial-L** multilingual model. The top 3 most frequent VQ codes are shown for each language and dataset, as well as the relative frequency for each particular language group. All four languages in the SIWIS data share code 81 as a frequent phone code. As Table 6.11 confirms, there is no overlap between the VCTK and SIWIS phone code usage for the top 3 most common VQ phone codes as they are entirely different models. For the multilingual model +**Adversarial-L** 33% of the phone codebook was active during training, whereas for the monolingual model +**Adversarial** 31% of the phone codebook was active during training.

Table 6.11: Relative frequency of VQ phone codes for the original +**Adversarial** model trained on VCTK English, as well as the adapted +**Adversarial-L** model trained on multilingual SIWIS data. The top 3 most frequent VQ codes are shown for each language, as well as the relative frequency given each particular language.

		Rank1		Rank2		Rank3	
	Language	VQ Code	Freq	VQ Code	Freq	VQ Code	Freq
VCTK	English (EN)	363	0.13	418	0.13	214	0.12
SIWIS	English (EN)	259	0.17	81	0.16	440	0.15
	French (FR)	259	0.16	81	0.15	191	0.15
	German (DE)	81	0.16	89	0.14	56	0.14
	Italian (IT)	440	0.18	81	0.15	259	0.14

6.6.3 Speaker Codebook

In this analysis the VQ speaker codebooks were compared between the monolingual and multilingual systems. Table 6.12 shows the distribution of VQ speaker codes for each language including VCTK English (for the +**Adversarial** system), as well as the four languages from SIWIS (for the +**Adversarial-L** system). Due to the fact that speakers in SIWIS are bilingual/trilingual, some overlap in the VQ speaker codes is expected across the four SIWIS languages. However, none of the VQ speaker codes representing speakers in the four languages from SIWIS data are shared with the VQ speaker codes from the original VCTK data. This was unexpected but could be a consequence of fine-tuning where it is easier for the model to learn new speaker codes from scratch than to adjust the values for speaker codes that had already been learned. The monolingual system (+**Adversarial**) utilized 18 unique VQ speaker codes for 99 unique speakers (7% of codebook). After the system was adapted to multilingual data the system utilized 11 unique speaker codes for 36 unique speakers (4% of codebook). Table 6.12 confirms that the two systems did not overlap or re-use speaker codes. Fine-tuning +**Adversarial-L** led to new speaker codes being learned. Again the speaker codebook did not learn a one-to-one mapping between VQ codes and unique speakers, similar to what was observed for the monolingual +**Adversarial** and +**Speaker/F0** systems. However the multilingual model learned more speaker codes in proportion to the number of unique speakers compared to the other models which could have been due to the diversity of languages.

Table 6.12: Comparison of the VQ speaker codes that were learned in the +**Adversarial** VCTK English model with the +**Adversarial-L** SIWIS multilingual model. The models utilize different codes. The same VQ codes are used in each of the four SIWIS languages.

	Language	# Spk	# Utt	Unique VQ Speaker Codes
VCTK	English (EN)	99	40314	5, 12, 19, 30, 48, 52, 67, 73, 109, 113, 128, 129, 135, 188, 193, 220, 232, 242
SIWIS	English (EN)	22	2387	42, 65, 84, 85, 131, 165, 179, 192, 216, 238, 248
	French (FR)	31	3405	42, 65, 84, 85, 131, 165, 179, 192, 216, 238, 248
	German (DE)	17	1719	42, 65, 84, 85, 131, 165, 179, 192, 216, 238, 248
	Italian (IT)	16	1689	42, 65, 84, 85, 131, 165, 179, 192, 216, 238, 248

6.7 Further Considerations

There are several important considerations regarding this analysis. First the compression ratio that was used in all of the VQ-VAE system variants made the sequences of phone codes extremely long. When combined with the fact that the phone codebook was also very large, it is difficult to analyze the phone codebooks with the objective of finding a mapping between VQ codes and phones. It is not clear what the VQ phone codes are modeling.

The VQ phone vocabulary usage analysis at the beginning of this chapter showed that the vocabulary was difficult to model compared to natural language. Using the phone code sequences as a history for predicting the next code in sequence would be equally difficult because the phone code sequences do not have systematicity. The conditional probability analysis only examined prediction with unigrams. It may be possible to utilize skip-grams or another type of token sequence history. However, it is not clear what the benefits would be except for potential applications such as text-to-speech synthesis, which is not explored in this thesis. When analyzing conditional probabilities, the phone and accent analysis did not produce strong conclusions. However there was evidence that accent information is associated with VQ speaker codes therefore a further analysis of predicting speaker from phone codes (or predicting phone codes from speaker) is not necessary.

The VQ speaker vectors appear to be modeling accent. It may be possible to attempt further disentanglement of speaker and accent information however it may require a completely different architecture than ones tried in this thesis. One potential benefit of separating speaker and accent information relates to voice privacy, such as concealing an accent [Noé et al., 2021], or controlling or reducing accent through voice conversion [Aryal and Gutierrez-Osuna, 2014]. Such applications are beyond the scope of this thesis.

6.8 Conclusion

This chapter has provided several different analyses that contribute to a deeper understanding of the learned VQ codebooks as well as disentanglement. The objective was to explore two of the three proposed principles of disentanglement:

1. The learned representations are sufficiently rich to model targeted speech features.
2. Representations that model targeted speech features do not also model non-target speech features (e.g., a representation of speaker identity does not also model speech content).
3. Representations have utility outside of the system that disentangles information.

The analysis in Section 6.3 and Section 6.6 addressed aspects of the first principle by showing that there is not a one-to-one mapping between unique speakers and learned VQ speaker codes. Section 6.3 also provided some evidence that VQ F0 codes may be modeling unvoiced/silence regions consistently while Section 6.4.3.3 showed that VQ F0 codes correspond to speaker gender. The second principle was addressed through a data probing analysis in Section 6.4.4. It was shown that speakers who are more similar in accent and who speak the same set of sentences have a low KL divergence for VQ phone code distributions. On the other hand speakers who are dissimilar in accent and who speaker different sets of sentences have higher KL divergence for VQ phone code distributions.

There is no single measure of disentanglement. Taken all together, the analysis presented in this chapter allows for some conclusions to be drawn based on quantitative evidence. The VQ phone codebooks are not interpretable. Evidence from the vocabulary analysis and perplexity indicate that VQ phone codes are not functioning similarly to natural language words and phones. Even when a tool such as sentencepiece was used to identify statistically relevant groupings of VQ codes, this did not uncover any patterns or groupings that are similar to natural language. More evidence that the VQ phone codes are not interpretable comes from the analysis using conditional probabilities. Some VQ codes are highly likely to occur with a given phone, but also those same VQ codes are highly likely to occur whether the phone is common or rare and whether it is a vowel or consonant.

The VQ F0 codes are slightly interpretable. A particular VQ F0 code was identified as being highly likely to correspond to unvoiced/silence regions of speech. Another different VQ F0 code was identified to correspond to female speakers, while yet another different VQ code was identified to correspond to male speakers. These findings were based on analysis with conditional probability that aligned the quantized F0 bins with VQ F0 codes. The VQ code corresponding to female speakers is likely to represent higher ranges of F0 while the VQ code corresponding to male speakers is likely to represent lower ranges of F0. These findings were confirmed again later during the codebook analysis. The VQ code for silence was a clear outlier when visualized in 2D space. The VQ F0 codes corresponding to male and female speakers were very distant from each other in the 2D space.

The VQ speaker codebooks carry accent information. This was uncovered in the analysis using conditional probability with speaker codes and accent. It was also confirmed in the k-means clustering of the speaker codebooks. This association between speaker codes and accents occurs in both systems that are attempting disentanglement through the use of phone and speaker codebooks. VQ speaker vectors that were likely to occur given a particular accent

were also found close in adjacent clusters. The association of accent information with the speaker representations is interesting because accents are often realized through phone representations. No evidence was found linking the phone codes to accent, even in the data probing tasks that compared phone distributions among groups of speakers. Even when a group of speakers who were more heterogeneous for accent were compared, they were determined to be highly similar based only on the spoken content.

Any claims of disentanglement must be considered as a relative claim, such as between two systems. This chapter did not provide absolute evidence that speaker/phone/F0 information is fully disentangled. There is some evidence that the +**Adversarial** system stood out from the other two systems **VQ-VAE** and +**Speaker/F0**, but it's not clear if the reason is because +**Adversarial** had better overall disentanglement. The active codes from the VQ speaker codebook for +**Adversarial** have higher cosine similarity compared to +**Speaker/F0**. The distributions of VQ phone codes in +**Adversarial** indicate that many codes are equally probable even though +**Adversarial** had more active VQ phone codes than the other systems. From the analysis it is not clear if the +**Adversarial** system has better speech synthesis and generalization to new speakers and content (from Chapter 5) because it has better disentanglement or because the system learns to ignore information. Finally it was shown that fine-tuning a system from monolingual to multilingual changes the speaker and phone codebooks entirely. The next chapter will begin to address the remaining principles of disentanglement. A set of speech tasks that require phone and speaker representations will be evaluated. The principle that will be explored is: *do the learned representations have utility and function outside of the system that disentangled the information?*

Chapter 7

Extrinsic Analysis of Disentangled Representations

7.1 Introduction

The analysis that was presented in the previous chapter (Chapter 6) explored various techniques to probe the learned VQ codebooks using VQ-VAE system variants from Chapter 5. The analysis revealed that one system variant in particular, +**Adversarial**, tended to learn better separation of speaker and content information. That system was also found to generate the highest quality synthetic speech based on the evaluation of speech synthesis provided in Chapter 5. So far, the following claims can be made regarding disentanglement in this thesis:

- Individual VQ phone codes are not interpretable
- VQ speaker codes appear to carry speaker identity as well as speaker accent information
- Disentanglement is relative and there is no single metric to quantify this
- Fine-tuning to multilingual data resulted in learning entirely different VQ codes

In this chapter, the analysis of the VQ codebooks will be expanded to task-based assessment. For task-based assessment, the learned VQ representations will be utilized in a series of tasks that inherently require some form of disentanglement in order to perform the task. For example, voice conversion requires that the content information remain unchanged while only the speaker voice information changes. Recall the three main principles of speech disentanglement under investigation in this thesis:

1. The learned representations are sufficiently rich to model targeted speech features.
2. Representations that model targeted speech features do not also model non-target speech features (e.g., a representation of speaker identity does not also model speech content).
3. Representations have utility outside of the system that disentangles information.

The present chapter will be focused on the third principle. The goal for this chapter will be to explore how well the learned representations can be utilized in other tasks besides speech

synthesis. Each of the tasks presented in this chapter should be considered a toy problem and they will be explored for the purpose of establishing a proof-of-concept. Some of the tasks purposefully utilize utterances that were seen in training of VQ-VAE systems (from Chapter 5) to demonstrate task feasibility in the least challenging case: the goal is not to engineer systems that generalize well in unseen conditions, rather the goal is to demonstrate and analyze disentanglement that was learned during training. If the disentanglement or utility of learned representations is poor for utterances seen in training, there is no reason to expect that the utility would generalize for unseen speakers or content. Ideally, the dataset used for training VQ-VAE system variants in Chapter 5 would have been large enough to create several different held-out partitions that could be used in downstream tasks. One such dataset was considered: LibriTTS [Zen et al., 2019]. However, early experiments from Chapter 5 proved that training VQ-VAE to synthesize speech from LibriTTS was difficult and the speech was often unintelligible. The low quality synthesis was likely due to variable quality of recordings in the LibriTTS dataset. VCTK was an obvious alternative because of the studio-level recording quality. None of the tasks presented in this chapter will undergo a full-scale evaluation. The idea behind these tasks is to observe what is possible with the learned VQ representations and to get a sense of the limitations in downstream tasks.

Similar to Chapter 6, the system variants being explored here will vary depending on the task. Five different tasks will be presented: phone recognition, speaker diarization, content masking for privacy, voice transformation, and language code-switching. Phone recognition and speaker diarization tasks seek to compare performance between the original VQ-VAE and the self-supervised/semi-supervised system variants. The purpose of limiting the speaker diarization and phone recognition tasks to these systems is to focus on speaker/content separation and to better understand the performance of self-supervised and semi-supervised systems. For phone recognition in particular, the dual-encoder systems demonstrated better intelligibility after synthesis compared to the triple-encoder systems (Chapter 5, Table 5.1). Likewise the dual-encoder system achieved higher KL divergence than the triple-encoder system when comparing per-speaker phone code distributions (Chapter 6, Figures 6.33 to 6.36) so the speaker diarization tasks here will be limited to dual-encoder systems. The dual-encoder models also had better naturalness across all four testing conditions in the MUSHRA test (Chapter 5, Figure 5.13 to Figure 5.16). The remaining tasks will be focused on the system +**Adversarial** because that system was identified as the best-performing system overall for speech synthesis in Chapter 5, and stood out in several ways in the analysis of Chapter 6. The content privacy task explores methods that conceal specific targeted words in an utterance without modifying the remaining portions of the utterance. The voice transformation task explores using the VQ speaker codebooks to transform voice in both the monolingual +**Adversarial** and multilingual +**Adversarial-L** models. Finally, the +**Adversarial-L** model will be used to explore linguistic code-switching.

7.2 Related Work

In speech applications, the term disentanglement is used to describe the process of identifying distinct informational factors from speech inputs [Peri et al., 2020a; Ravanelli et al., 2020; Ebbers et al., 2021; Polyak et al., 2021]. The primary way of identifying distinct informational factors is to use probing tasks that treat disentangled representations as inputs to a machine learning

task, such as classification, and measuring how well the representations perform for the task. Disentanglement for speech originated from the desire to create speaker embeddings that were more "pure", and that also minimized irrelevant information. Speaker embeddings were then evaluated in machine learning tasks such as emotion and sentiment recognition [Williams and King, 2019; Raj et al., 2019; Peri et al., 2020a]. The classification accuracy on these tasks was monitored. Disentanglement was then claimed to be achieved based on a gain or loss of accuracy.

A similar probing approach was also taken in the work of Ebbers et al. [2021] as they explored methods to purposefully separate content and speaker information. They evaluated the technique of contrastive predictive coding (CPC) using phone recognition and speaker recognition tasks. They claimed that disentanglement was achieved based on the evidence that phone embeddings performed poorly at speaker recognition and speaker embeddings performed poorly at phone recognition. Apart from these two contrasting downstream tasks, no other measures or perspectives of disentanglement were offered.

7.3 Phone Recognition

In this section, a phone recognition task is introduced. The purpose of the phone recognition task is to build upon the analysis from Chapter 6 which had demonstrated that the VQ phone codebooks were not interpretable. However, in that analysis the VQ phone codes were considered in terms of a zero history because the analysis involved unigrams. Further, the analysis from Chapter 6 examined only the +**Adversarial** system but did not further examine the other self-supervised system variants. In this phone recognition task, the performance will be compared for the original **VQ-VAE** system, along with a baseline that used audio features, as well as three dual-encoder system variants: +**Global VQ**, +**Speaker**, and +**Adversarial**. The +**Global VQ** system was fully unsupervised. The +**Speaker** and +**Adversarial** systems were both semi-supervised. For both of the semi-supervised systems, the softmax and angular-softmax variants (introduced in Chapter 5.3) will be included for comparison in this phone recognition task.

7.3.1 Data Preparation

The TIMIT dataset was designed for the purpose of automatic speech recognition [Garofolo, 1993] and that is why it was chosen for this phone recognition task. The data contains read speech from 630 different speakers of English from 8 different American English accents. Each speaker read 10 sentences. All of the speech has corresponding transcription including time-aligned phones and time-aligned words. When the dataset was first created, the transcriptions had been verified by hand before the dataset was released. The sentences are considered to be "phonetically rich" meaning that each utterance demonstrates a diversity of phones used. Overall, the TIMIT dataset contains 63 unique phone types. The TIMIT dataset is split by default into a training and test set by default wherein the training set contains 10 utterances from 462 speakers (total 4,620 utterances) and the test set contains 10 utterances from 168 speakers (total 1,680 utterances), with no speaker overlap.

Table 7.1: Number of training steps that were performed for each system variant when adapting the model to TIMIT data as well as the number of unique VQ phone codes from each system that were active during training. (S: softmax, AS: angular-softmax.)

Method		# VQ Phone Codes	Fine-tune Steps
VQ-VAE		140	126k
+Global VQ		119	408k
+Speaker	**S**	139	102k
	AS	138	102k
+Adversarial	**S**	176	126k
	AS	154	126k

7.3.2 Experiments

The experiments in this section are based on VQ phone codes that were obtained from the TIMIT dataset. The first step was to fine-tune each of the VQ-VAE model variants to the TIMIT dataset. This fine-tuning step is necessary because the speech synthesis quality is low for TIMIT data without fine-tuning. Table 7.1 shows how many additional training steps are performed on each system variant as well as the number of unique VQ phone codes that are used for the TIMIT data. The fine-tuning stopping criterion is based on monitoring the overall loss for that system. For the purpose of fine-tuning, the default dataset is divided so that the training portion of the TIMIT dataset (4620 utterances) is used for both training and validation in fine-tuning. This is done to preserve the test portion of the TIMIT dataset as a fully held-out set. Each of the training and validation sets contain 5 utterances per speaker for 462 speakers (the speakers are overlapping between the training and validation splits). Therefore the training set has 2,310 utterances and the validation set has a different set of 2,310 utterances. These train/valid/test splits are the same for all of the VQ-VAE system variants that are fine-tuned in this task.

After fine-tuning each system variant, the TIMIT data can be used to obtain VQ phone code sequences for a downstream phone recognition task. To obtain the VQ phone code sequences, the TIMIT data is passed through each system variant in an inference step separate from model training to obtain sequences of VQ phone codes. The sequences of VQ codes are converted into sequences of one-hot vectors (Figure 7.1). The dimensionality for one-hot vectors is different for each system and depends on the number of unique VQ phone codes used (Table 7.1).

The phone recognition task is done using a simple LSTM encoder-decoder architecture. To train the LSTM a slightly different data partition from the VQ-VAE fine-tuning is used because the partitions from fine-tuning are not adequate for LSTM training. The LSTM benefits from more training examples and fewer validation examples. The held-out test set remains completely held-out, however the training and validation split is adjusted to 3,694 utterances for training and 924 utterances for validation (note: 2 files are omitted from LSTM training because they caused an error when obtaining VQ phone code sequences).

The LSTM input consists of a sequence of one-hot VQ phone codes and the LSTM output is a string of the recognized phones. To be clear, the integer indices of the VQ phone codes are transformed into one-hot vectors rather than using VQ phone vectors. This is done to simplify

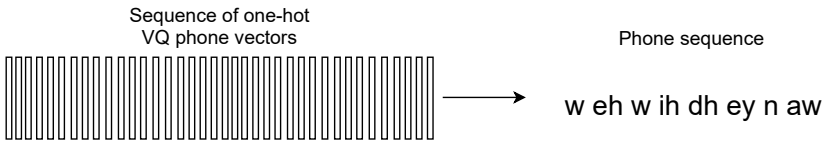


Figure 7.1: The input to LSTM is a sequence of VQ phone code indices represented by one-hot vectors (or mel-scale filterbank features for the audio baseline) and the output is a string of phones.

the task and to conceptualize that a sequence of integers could be mapped to a sequence of phones. This strategy also constrains the problem to be sensitive to the "vocabulary size" of active VQ phone codes (recall that not all VQ phone codes were active when the VQ-VAE systems were trained). In the case of the baseline, the input is audio features represented as mel-scale filterbank features using 80 mel bins. For implementation of the LSTM network, the ESPNet toolkit[1] is used [Inaguma et al., 2020]. The recipe is the CMU AN4 ASR [Kim et al., 2017] based on a very simple LSTM model with 64 units. For all system variants being tested in this experiment (including the audio baseline) the LSTM is trained to 100 epochs. The best LSTM model is selected based on connectionist temporal classification (CTC) loss ($mtlalpha = 1.0$). The decoder beam size is 20.

7.3.3 Results

Table 7.2 shows the mean phone error rate (PER) for the held-out test set averaged across all speakers. The PER is the sum of phone substitution, insertion, and deletion. Bold values signify the audio baseline (best performance) as well as the best-performing VQ-VAE system variant +**Speaker AS**. All of the system variants offer error reductions for insertion and deletion compared to the strong audio baseline which has fewer substitutions overall. The +**Speaker** system with angular-softmax (**AS**) has better overall performance compared with the +**Global VQ** system. These results show it is possible to add global-level speaker components to the original VQ-VAE model without sacrificing the utility of VQ phone codebook representations. The VQ-VAE system variants all tend to make errors for substitution more often than the audio baseline.

7.3.4 Summary

The phone recognition task has shown that all of the VQ-VAE system variants make fewer errors for insertion and deletion compared to an audio baseline. However, the VQ-VAE system variants make significantly more substitution errors and these substitution errors resulted in overall worse PER performance. The dual-encoder +**Speaker** system achieved similar performance to original single-encoder **VQ-VAE** system, which suggests that the addition of a VQ speaker codebook did not negatively affect the performance of the VQ phone codebook.

[1] `https://github.com/espnet/espnet`

Table 7.2: Phone error rate (% PER) on TIMIT test set from sequences of one-hot VQ phone codes or mel-scale filterbank features for the audio baseline. (S: softmax, AS: angular-softmax). The TIMIT test set consists of 168 speakers each with 10 utterances each and using a mix of 8 different American English accents. Bold values signify the audio baseline (best performance) as well as the best-performing VQ-VAE system variant +**Speaker AS**.

		% PER↓			
Method		Substitution	Insertion	Deletion	Overall
Audio Baseline		13.8	9.4	7.4	30.6
VQ-VAE		**26.6**	**8.3**	**6.0**	**40.9**
+**Global VQ**		28.0	7.7	6.4	42.1
+**Speaker**	**S**	28.1	9.6	5.8	43.4
	AS	**27.6**	**8.0**	**6.3**	**41.9**
+**Adversarial**	**S**	28.0	8.6	6.5	43.1
	AS	30.4	9.6	6.5	46.5

7.4 Speaker Diarization

The speaker diarization task that will be described in this section presents a technique for using VQ speaker codes for labeling speaker regions in audio that has two different speakers. The proposed technique will be compared with performance of an x-vector baseline from the 2nd DIHARD Challenge[2]. The goal of speaker diarization is to annotate specific regions of speech that belong to different speakers. This task is chosen in lieu of speaker recognition on the basis that Chapter 6 analysis revealed that the VQ speaker codebook did not create a one-to-one mapping of speakers and VQ speaker codes. Speaker diarization in this chapter only requires comparative mappings that reflect differences between two speakers. Recall from Chapter 6.3.3 that on average, each speaker is represented by ∼2 different VQ codes and each VQ code is used for ∼15 different speakers. In order to apply the VQ speaker codebooks to a diarization task, first the VQ speaker codes for a given speaker are used to create a reference based on code frequency. The process of using VQ speaker codes to create a reference guide will be described in more detail in Section 7.4.2.

The VCTK data that was used for training VQ-VAE systems (Chapter 5) always contains one speaker per utterance. In order to use VQ-VAE system variants in a speaker diarization task on VCTK data, the diarization will be simulated by concatenating speech from different speakers. In that sense the turns between speakers are created artificially. The benefit to simulating speaker turns with VCTK data is that it does not require fine-tuning VQ-VAE systems on (potentially very noisy) realistic conversational-style data[3].

7.4.1 Data Preparation

The VCTK data is used in the speaker diarization task as well as the testing conditions that were specified in Chapter 5. There are four testing conditions which represented different combinations of seen and unseen speaker/content during VQ-VAE system training:

[2] `https://dihardchallenge.github.io/dihard2/`

[3] Early attempts to fine-tune to the BU Radio Corpus [Ostendorf et al., 1995] resulted in consistently unintelligible speech.

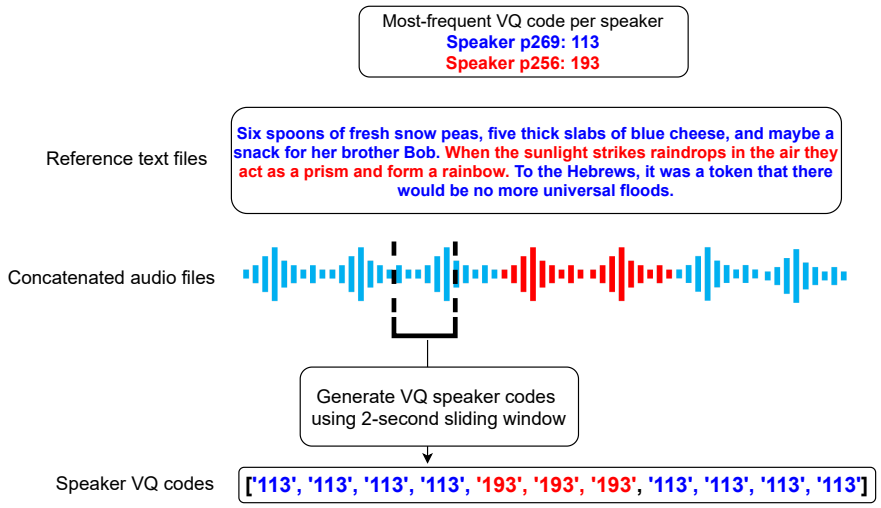


Figure 7.2: The most frequent VQ speaker code for a speaker is determined for reference. Audio files from two different speakers are concatenated together. VQ speaker codes are generated at each 2-second sliding window (using 250ms overlap). The VQ speaker codes determine which regions of speech belong to the different speakers. In this example, speakerA is **113** and speakerB **193**. All concatenated audio files use two turns between the speakers.

- C1 seen speakers / seen content
- C2 seen speakers / unseen content
- C3 unseen speakers / seen content
- C4 unseen speakers / unseen content

The C1 condition should result in the best quality of speech because it was seen during training. The remaining conditions help to gauge how well the models generalize to unseen conditions. The C4 condition should be more challenging because it represents full held-out conditions in terms of the content as well as the speaker.

The speaker diarization task in this section is simulated by concatenating audio files from different speakers so that each new audio file contains 2 speakers and 2 turns. This process is shown in Figure 7.2. When simulating speech with multiple speakers, the speakers are always chosen from similar testing conditions.

7.4.2 Experiments

Since the VQ speaker codebooks do not have a one-to-one mapping between VQ codes and speaker labels, a reference guide was created to map each speaker to a single VQ speaker code. This was done by examining the VQ speaker codes for audio files containing each single speakers and identifying the most frequent VQ speaker code. For example, if the most frequent VQ code for *speakerA* was **113** then code **113** was used as a reference guide for identifying speech that belongs to *speakerA*.

To obtain diarization labels from the concatenated multi-speaker audio, the speaker VQ codes were extracted at short intervals. A 2-second sliding window (250ms overlap) was used. For example, if the speaker code *113* was extracted in a 2-second window and that code belonged

to *speakerA* based on the reference guide, then that region of speech was labeled as *speakerA*. This process is demonstrated in Figure 7.2. In all VQ-VAE systems, sometimes two different speakers were mapped into the same VQ code and it was not possible to decide the most frequent VQ speaker code. In cases where the speaker could not be determined *speakerA* and *speakerB* were selected at random.

The VQ-VAE system variants were compared with the Track #1 x-vector baseline from the 2019 DIHARD challenge. The x-vector baseline extracted an x-vector using a 2-second sliding window (with 250ms overlap) and used PLDA scoring and agglomerative hierarchical clustering [Sell et al., 2018]. The x-vector extraction used a pre-trained model from VoxCeleb[4]. The baseline system clustering was trained on the official DIHARD data development set with knowledge of 2 speakers per audio file for the purpose of clustering. The baseline system was evaluated on the concatenated VCTK audio files (2 speakers, 2 turns).

7.4.3 Results

The diarization error rate (DER) is reported in Table 7.3. The DER score measures the proportion of time in an audio file wherein: non-speech was incorrectly labeled as speech (false alarm), or a speech region was incorrectly labeled as non-speech (miss), or an incorrect speaker label was generated (error). The +**Speaker** and +**Adversarial** systems performed better than the strong x-vector baseline on average. The self-supervised system (+**Global VQ**) did not learn a diverse VQ space, and it was observed that the same VQ speaker code was generated for all speakers. The term that has been recently adopted to describe this phenomenon is "codebook collapse" from Dhariwal et al. [2020] though no formal definition exists. As mentioned earlier, in this case the speaker regions were decided randomly so the scores for +**Global VQ** reflect random guessing. Overall the +**Adversarial** system performed the best across all four conditions. The x-vector baseline performed particularly well in conditions 1 and 3, and much worse in conditions 2 and 4. Since the x-vector baseline was not trained on any portion of the VCTK data, it is not due to seen vs. unseen content. It is possible that the particular speakers in conditions 2 and 4 had characteristics that were not well modeled by the pre-trained x-vector extractor, such as speaking rate or accent. In general, the performance in condition 3 was also best for +**Adversarial** and +**Speaker** when softmax was used in the system variants.

7.4.4 Summary

The results from speaker diarization demonstrate the learned VQ speaker codebooks can be applied to this complex task. Overall, two of the VQ-VAE system variants outperformed the strong x-vector baseline. It is possible that the x-vector baseline was influenced by accent or speaking rate. The +**Global VQ** system did not learn a diverse codebook and suffered from "codebook collapse" (a term borrowed from Dhariwal et al. [2020]) wherein the same VQ code was generated for all speakers.

[4] `https://kaldi-asr.org/models/m7`

Table 7.3: Speaker diarization error (DER) scores on concatenated VCTK audio. Each audio file contained 2 speakers and 2 turns. Speakers were selected from within the same testing condition. Four testing conditions were examined in order to explore performance in unseen conditions. (S: softmax, AS: angular-softmax).

		%DER↓ per Condition				
Method		C1	C2	C3	C4	Avg
x-vector		24.3	44.6	27.4	46.7	35.8
VQ-VAE		–	–	–	–	–
+Global VQ		44.4	39.1	44.7	39.6	42.0
+Speaker	**S**	32.4	32.2	31.0	33.1	32.2
	AS	34.6	35.9	36.4	35.9	35.7
+Adversarial	**S**	**32.2**	**32.3**	**30.5**	**32.9**	**31.9**
	AS	37.2	35.6	36.1	35.2	36.0

7.5 Content Masking for Privacy

Content-based privacy involves creating a capability that conceals certain sensitive words or phrases using techniques that do not disrupt the normal flow and feel of a speech utterance. Traditionally, speech content privacy has been rooted in the idea that a signal-emitting device can be set up within a physical space, such as an office room, to conceal what people say during private conversations. This device emits a special type of noise to mask semantically-relevant speech sounds, like words or phonemes [Akagi and Irie, 2012; Kondo and Sakurai, 2014; Donley et al., 2016]. The process of concealing content can also be understood in terms of speech synthesis. For example, in speech privacy applications that target *content*, the masking of certain sensitive phrases should not result in degraded speech or speaker information elsewhere in the masked utterance. In some cases, it might be preferable if a content mask does not disrupt the overall listening experience such as with a loud beep or white noise [Chen et al., 2008]. Masking targeted content could potentially take place while speech is in a compressed state, similar to how it is represented in a VQ-VAE system with VQ phone codes [Casebeer et al., 2021].

The content masking task in this chapter will explore the possibility of replacing VQ phone codes that represent content with VQ codes that represent noise. An assumption will be made that the targeted content (words) and location (start and end times) have already been determined. The task that will be presented in this chapter explores whether it is possible to replace targeted sub-sequences of VQ phone codes with noise and how that affects the overall quality of surrounding unmasked phrases. Further, the placement of the mask near the beginning or end of an utterance will be explored in order to observe any degradation in speech synthesis quality. Two different types of masking will be presented: noise and reversal. In this content masking task, only the **+Adversarial** system variant will be used as it has been shown to produce the highest-quality speech synthesis (from Chapter 5).

7.5.1 Data Preparation

The data that will be used for this task is the training portion of the VCTK data that was used for training the VQ-VAE system variants in Chapter 5. The training data was chosen because it

enables an exploration of content masking in terms of the best possible learned representations. The objective for this toy problem is not to evaluate the robustness of the VQ-VAE system across domains or test if the model can generalize. By using the same training data from VCTK, this problem can establish an upper bound for content privacy performance with VQ-VAE. The VCTK Montreal forced alignments were also used since they provide word-level start and end times [McAuliffe et al., 2017].

Utterances from the VCTK training set were passed to the +**Adversarial** system variant to obtain the VQ phone and speaker codes as an inference step separate from training. One of the masks that will be explored in this task involves replacing VQ phone codes with noise. Actually, the VQ phone codes are replace with VQ phone codes corresponding to noise. The noise that will be used is a waveform consisting of a temporally-modulated speech-shaped noise (SSN) masker (ICRA noise 9 from Cooke et al. [2013]). The noise was also passed to the +**Adversarial** system in an inference step, and the VQ phone codes for the noise were extracted. The noise VQ phone codes are suitable to use when replacing content VQ phone codes in the experiments.

7.5.2 Experiments

The first masking method was to replace original content VQ phone codes of the target phrase with VQ phone codes from speech-shaped noise (SSN). The idea for using SSN came from Dreschler et al. [2001]. The speech-shaped noise offers a non-recoverable masking, which is useful for applications where speech content redaction must be persistent. Even though the noise does not truly contain phones, the resulting VQ phone code sequence represented the noise quite well (as judged by the author of this thesis from listening to the original noise and synthesized noise side by side). The second masking method was to simply reverse the order of the original content VQ phone codes for the target phrase, while leaving the remaining VQ phone codes intact. The VQ code reversal method does render the target phrase unintelligible.

For a given utterance, two different masking positions were explored. Mask position #1 occurs early in the sentence and mask position #2 occurs later in the sentence. The masking process occurs at the VQ phone code level, so that the VQ phone codes are modified to perform masking. Montreal forced alignments were used for determining the word boundaries within the VQ phone code sequences. Two types of masking were used: reversing a VQ phone code sequence within the mask boundaries, or replacing a VQ phone code sequence with noise VQ phone codes.

Target phrases were manually selected such that the target phrase to be masked occurred either early or late within an utterance. The purpose of exploring the target phrase position was to observe any effects on the decoder ability to synthesize surrounding unmasked words. The forced-alignments were used to determine the start and end timestamps for the target phrases to be masked. Those timestamps were used for determining the location of the target phrase within the sequence of VQ phone codes. The VQ phone codes were modified by swapping the original VQ phone codes with a new sequence of phone codes from the mask. The swapped sequence corresponded to the duration of the target phrase and the surrounding VQ phone codes remained untouched. This process is demonstrated in Figure 7.3.

For the evaluation, two utterances were selected that were shared between a female and a male speaker. For each utterance, two target phrases were selected to mask at two different positions in the utterance (near the beginning and near the end). For the first utterance, the

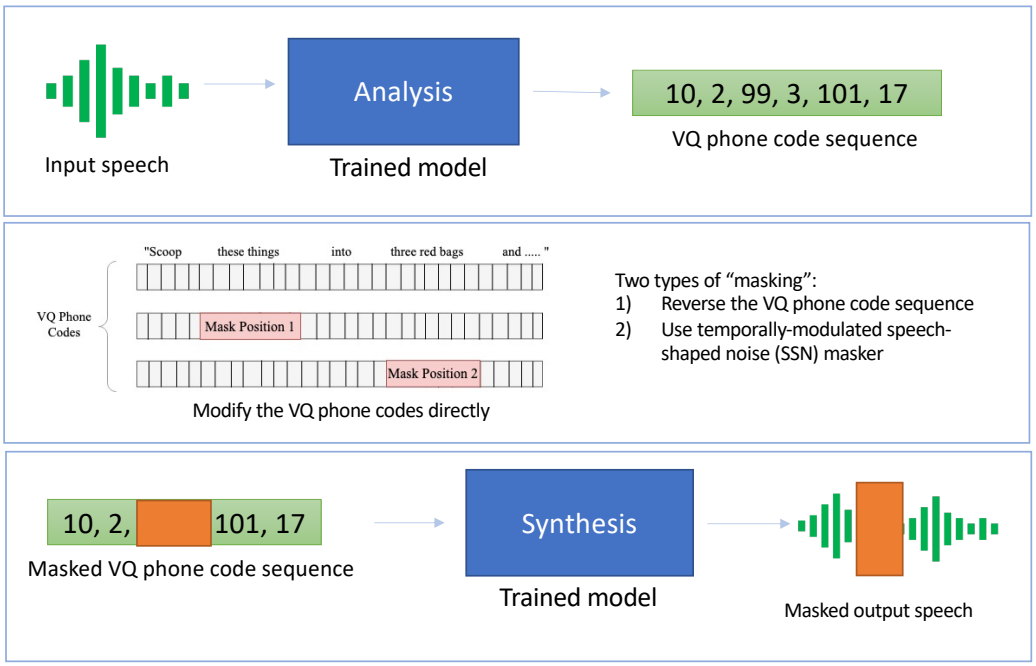


Figure 7.3: For a given utterance, two different masking positions were explored. Mask position #1 occurs early in the sentence and mask position #2 occurs later in the sentence. The masking process occurs at the VQ phone code level, so that the VQ phone codes are modified to perform masking. Montreal forced alignments were used for determining the word boundaries within the VQ phone code sequences. Two types of masking were used: reversing the VQ phone code sequence within the mask boundaries, or apply a temporally-modulated speech-shaped noise (SSN) masker (ICRA noise 9 from Cooke et al. [2013]).

two target phrases were "these things" (position1) and "three red bags" (position2). For the second utterance, the two target phrases were "sunlight strikes" (position1) and "raindrops in the air" (position2). In total, 16 examples were evaluated.

7.5.3 Results

The experiments were evaluated using a listening test with human subjects. All of the listeners were fluent in English and were recruited using the Prolific platform[5] while the test itself was administered through the Qualtrics platform[6]. A total of 20 participants were recruited and each were paid the equivalent of £ 7.50 per hour.

Participants were instructed that one or more words had been removed from the utterance, but were not told which ones. They were asked a yes/no question as to whether the speaker voice remained consistent throughout the utterance. The results are shown in Table 7.4 where the "Speaker Similarity" reflects the percentage of listeners who responded 'yes' to indicate that the speaker voice remained consistent within an utterance. Overall the SSN was better for maintaining speaker identity throughout the utterance compared to reversal. Masking the phrase at position2 resulted in more speaker consistency. It is possible that masking earlier in the utterances caused some performance degradation with the decoder because the decoder was

[5] https://www.prolific.co/
[6] https://www.qualtrics.com/uk/

based on auto-regressive speech generation from WaveRNN so errors early in the utterance may have had a compounding effect.

Listeners were also asked to take part in an A/B preference test for SSN versus reversal masking types. The preference test revealed a slight preference for SSN. Finally, the intelligibility was estimated from ASR-based word recognition and is reported as word error rate (WER)[7]. The IBM Watson Speech-to-Text API[8] was used for the ASR. A baseline WER was calculated on natural unmasked utterances and it was found to be 24.4%. This is higher than expected but likely due to pronunciations and the audio quality. The WER for masking is reported in Table 7.4. The WER is reported separately for the unmasked phrases and the masked phrases. To calculate the WER for masked phrases, the ASR transcript was examined to see if the words in the target phrase were present and if not then the WER is reported as 100% for the masked phrase only. To calculate the WER for unmasked phrases, the ASR transcript was compared with the original but with the target phrase removed from the original transcript. The words were converted to lowercase, punctuation was removed, and any ASR-based notation (such as %HESITATION) was also removed. Overall, the WER increased for unmasked phrases compared to the original speech and that is because some of the non-target phrases were unintelligible after masking. Example transcripts from ASR and the original texts are shown in Table 7.5. The WER for unmasked phrases was lowest when the target phrase was in position1 for both types of masking.

Table 7.4: Within-utterance speaker similarity and ASR-based WER for content masking, comparing two masking techniques and two different target phrase positions within an utterance.

	Speaker	ASR-Based WER	
Masks & Positions	Similarity↑	Masked↑	Unmasked↓
Reversal Position #1	63.7%	100%	47.7%
Reversal Position #2	77.5%	100%	69.2%
SSN Position #1	70.0%	100%	52.4%
SSN Position #2	76.2%	100%	61.5%

Table 7.5: Example transcripts from ASR (for utterances with masking) and the original texts

Reversal Position #1 – target phrase "these things"

Original	she can scoop these things into three red bags and we will go meet her wednesday at the train station
ASR	skip in the second three backpacks market may tell wednesday at the train station

SSN Position #1 – target phrase "sunlight strikes"

Original	when the sunlight strikes raindrops in the air they act as a prism and form a rainbow
ASR	thanks in advance the act as a prism and full knowing

[7] `https://pypi.org/project/jiwer/`
[8] `https://www.ibm.com/cloud/watson-speech-to-text`

7.5.4 Summary

The goal of speech privacy masking was to conceal targeted content without disrupting the intelligibility of surrounding words or causing changes to the speaker identity. It was found that masking at the beginning of an utterance (position1) resulted lower WER from ASR for the non-target words that were intended to remain intelligible. The within-utterance speaker similarity appears to be more consistent when the masking occurs at the end (position2). The results suggest that the decoder may "recover" slightly better from the SSN masking compared to the reversal masking.

7.6 Voice Transformation

Voice transformation involves the ability to model speaker identity in a way that is divorced from the utterance content. For disentangled representations, the quality of separation between phone and speaker can be observed by swapping VQ speaker codes while keeping the VQ phone codes constant. As it has been stated in Chapter 6, the dual-encoder VQ-VAE system variants did not learn a one-to-one mapping between speaker labels and VQ speaker codes. It is for that reason that this task will be framed as a "transformation" task rather than a "conversion" task. It is voice transformation because the input data will be transformed into a specific VQ speaker code, but not a specific target speaker. Neither the monolingual +**Adversarial** nor the multilingual +**Adversarial-L** models utilized all of the possible VQ speaker codes, from the available 256 for each model. In the multilingual model (SIWIS), 11 VQ speaker codes were utilized for 36 speakers. In the monolingual model (VCTK) there were 18 VQ speaker codes utilized for 110 speakers. Framing this task as voice transformation will also allow for voice mixing, wherein VQ speaker vectors could be combined, such as by taking an average between two vectors, in order to create a mixture of two VQ speaker codes. While the VQ code indices are discrete integers, each VQ code has a corresponding VQ vector that consists of continuous values so it is possible to calculate a centroid between two given VQ vectors.

7.6.1 Data Preparation

Similar to the reasoning for content privacy in Section 7.5, the training data was utilized in this task because it presents the easiest case where the learned representations are expected to be higher quality than those in test data (in terms of potential disentanglement as well as the speech synthesis quality). The purpose of a voice transformation task in this thesis is to demonstrate a capability rather than demonstrate that a model is robust and generalizes to out of domain speakers or content. Two different datasets were used in these experiments. For monolingual voice transformation with +**Adversarial**, VCTK training data from condition1 was used. For multilingual voice transformation with +**Adversarial-L**, SIWIS training data from condition1 was used. Data from Italian was omitted in the voice transformation experiments due to the strong data imbalance across languages. To prepare the data, the audio was passed through the VQ-VAE system variant as an inference step separate from model training. The VQ phone codes and VQ speaker codes were obtained as shown in Figure 7.4.

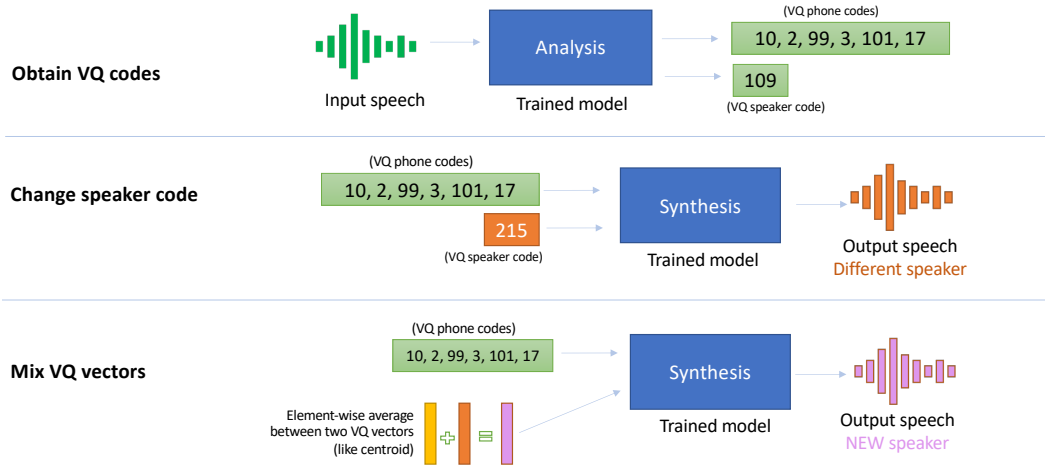


Figure 7.4: Diagram showing the voice transformation experiments. Input speech is passed to a trained encoder to perform inference and obtain sequences of VQ phone codes as well as a VQ speaker code. VQ speaker codes are swapped for a different speaker code from the codebook learned for that system. Two VQ speaker vectors are averaged to obtain a new speaker representation.

7.6.2 Experiments

There were two main experiments in the voice transformation task. One set of experiments was based on swapping VQ speaker codes to change the source voice. Another set of experiments was based on mixing VQ speaker vectors to create new voices. The VQ speaker vector mixing was possible due to the nature of the VQ codebook lookup table. Recall that all VQ codes map to VQ vectors. It is the vectors themselves that are passed on to the decoder (WaveRNN) for speech synthesis. Therefore it is possible to mix VQ vectors and then pass the resulting vector to the decoder. Figure 7.4 shows a comparison of both types of voice transformation.

7.6.2.1 Single Code Representation

For the multilingual model, one male and one female speaker were selected (**spk13**-male, **spk04**-female) from the SIWIS data and seen conditions. The VQ phone and speaker codes were extracted. The VQ speaker codes were replaced with each of the 11 multilingual VQ speaker codes from the codebook that was learned in training (described in Chapter 5). For the multilingual model, two utterances per speaker per language were used, making a total of 12 examples (3 languages, 2 speakers, 2 utterances). Recall that the +**Adversarial-L** system requires a one-hot language vector for the decoder. For the one-hot language vector the language from the source sentence was used. For the monolingual model and codebook, the same approach was followed by selecting a male and female speaker from the VCTK data and seen conditions (**p229**-female-English, **p302**-male-Canadian). For the monolingual model, two utterances per each speaker were used making a total of 4 examples (1 language, 2 speakers, 2 utterances).

7.6.2.2 Mixed Code Representations

For the mixed representations, the VQ vectors were combined in an effort to create new voices. This was one in a similar spirit as *one-shot* voice conversion [Wu et al., 2020]. Ideally, this could be done using various combinations of VQ speaker codes and weighting them. In this section, two VQ speaker vectors were mixed by calculating an unweighted mean between the vectors. In a vector space, the resulting representation is a new centroid that is equidistant between the paired vectors. VQ speaker vectors were randomly selected for each model and then mixed. Similar source utterances were selected for synthesis as described in 7.6.2.1. For the multilingual model, two utterances per speaker per language were used making a total of 12 examples (3 languages, 2 speakers, 2 utterances). For the monolingual model, two utterances per each speaker were used making a total of 4 examples (1 language, 2 speakers, 2 utterances).

7.6.3 Results

The voice transformation task was evaluated using a listening test with human subjects who self-reported as fluent in the respective language that was being tested. 20 participants per language were recruited using the same platform, payment, and test delivery as reported in 7.5.3 (Qualtrics and Prolific). For the listening test evaluation, 4 speaker VQ codes (3 single-representations, 1 mixed) were selected from each model. As stated before, two source sentences were used for synthesis (one male and one female). In the case of the multilingual model, this was per-language. Participants listened to all 8 samples in their language and marked naturalness on a scale of 1 to 5. Table 7.6 and Table 7.7 show the results from naturalness. In the multilingual model, German and French each had the overall best quality MOS scores for all three VQ speaker codes as well as the mixed voice whereas English had the worst. The high MOS scores for German indicate that the multilingual model has a dominant language in terms of quality. The monolingual results in Table 7.7 show consistency across each of the speaker VQ codes. It is not possible to compare MOS across systems or languages because they used different source speakers as well as different listeners, however some speaker codes appear to result in more natural speech within a language.

Table 7.6: Multilingual MOS naturalness scores from human judgements for voice transformation and voice mixing in English, French and German (SIWIS) data using VQ speaker codes from the +**Adversarial-L** system variant.

Speaker Code	English	French	German	Avg
Code 85	2.4	2.9	3.4	2.9
Code 192	2.6	3.0	3.1	2.9
Code 238	2.5	3.0	3.2	2.9
Code 131+248	2.4	3.1	3.3	2.9

Table 7.7: Monolingual MOS naturalness scores from human judgements for voice transformation and voice mixing in English (VCTK) using VQ speaker codes from the +**Adversarial** system.

Speaker Code	Avg
Code 67	2.3
Code 109	2.3
Code 242	2.5
Code 109+242	2.4

The listeners were also asked about speaker similarity in an effort to understand the consistency of the VQ speaker codes. Listeners were provided with matched and unmatched pairs in an A/B test, and were asked to decide if the A/B examples were from the same or different speaker. There were 16 total matched pairs and 24 unmatched pairs per language and dataset. This format worked well since there was not a prototypical target voice for the comparison. Comparing VQ code similarity allows for observation of trends across a particular language as well as across a particular VQ speaker code. Mixing VQ speaker vectors did not impact the overall quality of synthesis. Figure 7.5 shows that German language samples resulted in the highest similarity between the VQ speaker codes, followed by French, and then English. Some VQ speaker codes are judged to have high similarity in two or more languages, for example *192* and *131+248* for each of German and French. Figure 7.6 had overall worse similarity compared to the multilingual model, and likewise codes *67* and *242* appear to be similar.

Figure 7.5: Multilingual +**Adversarial-L**: matrix representation of voice transformation speaker similarity for results of A/B similarity tests. The German language samples resulted in the highest similarity between the VQ speaker codes, followed by French, and then English. Some VQ speaker codes are judged to have high similarity in two or more languages, for example *192* and *131+248* for each of German and French.

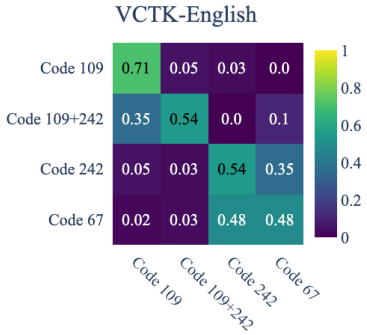


Figure 7.6: Monolingual +**Adversarial**: matrix representation of voice transformation speaker similarity for results of A/B similarity tests. Some of the codes sound similar to the listeners, such as *67* and *242*.

7.6.4 Summary

The voice transformation task revealed that it is possible to create new voices using VQ speaker vector mixing. Overall the German voice transformation utterances had higher quality and higher speaker similarity compared to the other languages. Some codes are similar within each model. For example in the multilingual model codes *192* and *131+248* sound very similar to listeners for German as well as to listeners of French. The ability to perform voice transformation with high MOS naturalness is an indication that there is some disentanglement between the phone content and speaker information in the VQ codebooks.

7.7 Language Code-Switching

The purpose of exploring linguistic code-switching is to find out if speech can be generated from VQ phone codes in completely different languages in the same utterance. Code-switching will be simulated in this task by concatenating together VQ phone codes from utterances in different languages and using only one VQ speaker code for synthesis. This task will also take advantage of the fact that the SIWIS data contains utterances from bilingual and trilingual speakers, so it is possible to ensure that natural source utterances in two different languages are spoken by one single speaker. However, recall from Chapter 6 that there is not a one-to-one mapping between speakers and VQ speaker codes. Only one VQ speaker code will be used for the code-switching utterances. The expected outcome is an utterance in two languages that sounds consistently like one speaker throughout the utterance.

7.7.1 Data Preparation

The data in this task was the SIWIS dataset because only the multilingual +**Adversarial-L** system was utilized. Files from the SIWIS training set were passed to the system as an inference step separate from model training to obtain the VQ phone codes and VQ speaker code. As described earlier in Section 7.6 and Section 7.5, the training data was used in order to simplify the problem and examine the scenario where performance on the task is expected to be the best since the model is not being required to generalize to unseen speakers or content. There

is no guarantee that the same VQ speaker code would be obtained for both utterances, even if the source speaker is the truly the same speaker. Only the VQ speaker code from the first utterance will be used. For this reason, all of the code-switching language pairs (English/French, English/German) are concatenated in two different language orderings (e.g., English-French, French-English). This process is described in Figure 7.7.

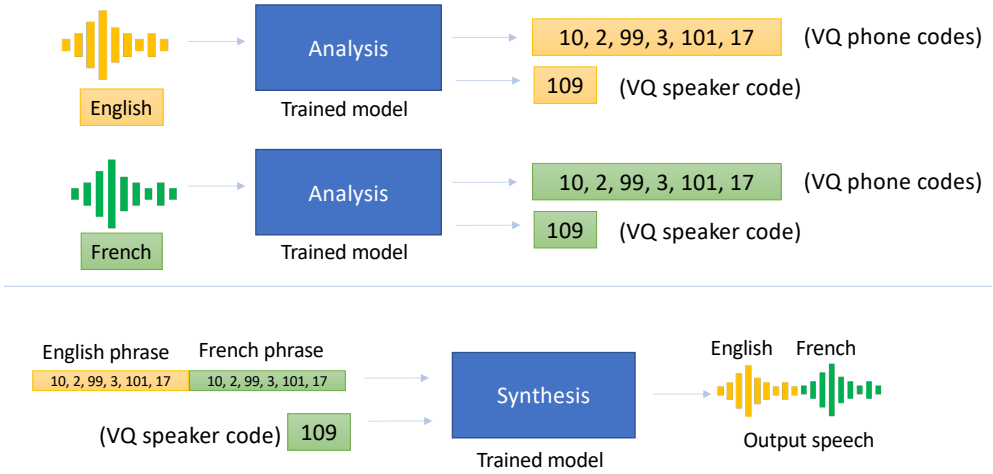


Figure 7.7: A demonstration of language code-switching by obtaining VQ phone codes for speech in each language and then concatenating the VQ phone code sequences before synthesis. The result is speech in two languages. Note in this example, the speaker with VQ code *109* is bilingual so the VQ speaker code remains the same.

7.7.2 Experiments

To simulate language code-switching, the VQ phone codes from entire files were concatenated together as demonstrated in Figure 7.7 because the label files that came with the SIWIS dataset had proven to contain unreliable time alignments for all of the languages. Two speaker were selected for each language, one male and one female. Six utterances were then selected for English-German and 6 utterances were selected for English-French. Both language pairs used male and female speakers. In addition, the language order was reversed so that the pairs were: English-German, German-English, English-French, and French-English. The order of languages was reversed in order to observe whether the WaveRNN decoder was particularly sensitive to language ordering, since the decoder could only accept a single one-hot language code. There were a total of 24 code-switched files (6 per language pair). For the one-hot language vector the language of the first utterance was used. Apart from concatenating the VQ phone codes (as demonstrated in Figure 7.7) there were no other modifications made.

7.7.3 Results

The main objective for this task was to find out if the multilingual model could preserve speaker similarity while also synthesizing the multilingual speech. Listeners were recruited in the same manner as described earlier in 7.6.3. Participants were required to self-identify as fluent in both of the languages belonging to a language pair. 20 participants per language pair were recruited,

for example the same participants did tasks in English-German as well as German-English.

In a modified A/B test, participants were presented with (A) code-switched speech from concatenated VQ phone codes, and (B) code-switched speech from concatenated audio files. The participants were asked to to decide if the speaker was the same between the two A/B samples. The results are reported in Table 7.8 and the values reflect the percentage of responses indicating that 'yes' the speaker is the same in sample A and sample B. The French-English pair had the lowest agreement. It is possible that the WaveRNN decoder was not able to recover from generating speech in the French language and switch to English midway through. This could have been caused by the data imbalance as French is the majority language in the SIWIS dataset.

The listeners were also presented with single code-switched utterances from only (A) that were based on concatenated VQ phone codes. The participants were asked to judge if the speaker voice was consistent throughout each utterance, or if the speaker had changed part-way through. All of the language-switching was set up using bilingual or trilingual speakers, and the utterances carefully selected to ensure that the speaker remained the same. So the expected result is that the speaker would sound the same regardless of the simulated language code-switching. Results in Table 7.8 reflect the percent of listeners who judged that the speaker had not changed. There was slightly more consistency for English-German pairs, compared to French.

Table 7.8: Speaker similarity for linguistic code-switching. A/B examples compare speaker similarity from VQ phones with the concatenated audio. A-only is within-utterance speaker consistency. German-English has the highest speaker consistency.

	Speaker Similarity	
Data	A/B↑	A only↑
English-French	57.9%	69.0%
French-English	30.8%	60.7%
English-German	67.5%	77.5%
German-English	75.0%	77.5%

7.7.4 Summary

The language code-switching task has demonstrated that the multilingual +**Adversarial-L** system can perform language switching, at least at the utterance level. The speaker similarity for French-English code-switching was inconsistent between the concatenated audio files and the concatenated VQ phone codes. The reason for this is not known but it could be related to the decoder ability to "recover" from French and switch to generating English. It could also be due to different VQ phone code distributions between French and English. Overall, the German-English pairs had high speaker consistency within the utterances.

7.8 Further Considerations

Some of the related work that was described earlier in this chapter had presented disentanglement in terms of contrasting prediction or probing tasks. For example Ebbers et al. [2021] had evaluated speaker and phone representations in a phone recognition task. The work presented in this chapter did not follow a similar procedure as Ebbers et al. [2021]. In this thesis, it is not possible to do a phone recognition task from VQ speaker codes/vectors because of the temporal pooling that was built into the VQ-VAE system variants, resulting in one speaker code per utterance. It is not possible to perform a speaker recognition task based on VQ phone codes/vectors because they would need to be recombined into a new representation or treated as a sequence. However the sequences depend on factors such as duration of utterance and speaking rate, which may be speaker dependent but is not a useful way to represent speaker information for real-world speech applications such as automatic speaker verification. Therefore contrastive probing tasks were omitted in this thesis.

Using VQ vectors instead of one-hot vectors of VQ codes may have improved the performance for phone recognition. Furthermore, the phone error rate substitution was the most significant type of error made by VQ-VAE system variants. It would be interesting to find out if the substitutions were reasonable or if they were related to factors such as speaker accent. It was shown in Chapter 6 that the speaker accent information is most likely being carried by the VQ speaker codes rather than the VQ phone codes. It is possible that the overall quantity of substitution errors would decrease if the analysis was repeated using a single-speaker dataset.

7.9 Conclusion

This chapter has presented five separate tasks that demonstrate the utility and function of learned VQ representations. The tasks have included: phone recognition, speaker diarization, content privacy masking, voice transformation, and language code-switching. The objective set forth in this chapter was to better understand if the learned representations had a use outside of the system that had performed disentanglement.

The phone recognition and speaker diarization tasks have been particularly useful for assessing the objective. From phone recognition, it was found that the dual-encoder system +**Speaker** performed better (on average) than the other dual-encoder VQ-VAE variants but not as well as the audio baseline. The addition of a VQ speaker codebook did not significantly degrade performance on this task compared to the original single-encoder **VQ-VAE**. It was also found that the VQ-VAE system variants made similar proportions of error types in phone recognition as the audio baseline. Overall, the VQ-VAE variants had less insertions and deletions than the baseline but more substitutions. The speaker diarization task demonstrated that it is possible to use VQ speaker codes for diarization. The +**Adversarial** system performed best on speaker diarization (on average) even compared to the strong x-vector baseline. The +**Global VQ** system only produced one single VQ speaker code for all speakers, and is evidence of "codebook collapse".

The content masking for privacy task revealed that it may be difficult for the WaveRNN decoder to recover from noise masking when the mask occurs early in the utterance. At the same time, this task demonstrated that it is possible to mask words in an utterance while the speech is in a compressed state (the VQ codebook and code sequences are considered

to be a compressed state). Likewise the WaveRNN decoder was also tested in the language code-switching task. There was some evidence to suggest that the decoder is not able to recover from the language shift of French to English midway through an utterance. Finally, the results from voice transformation indicate that there has been some disentanglement between the phone content and speaker information based on the MOS naturalness and speaker similarity judgements from human listeners.

These tasks taken altogether along with the analysis that was presented earlier in Chapter 6 show that there is some evidence of disentanglement between phone content and speaker information. Just as Chapter 6 had demonstrated there is no single measure of disentanglement, the tasks presented here in this chapter show that there is no single task that can prove disentanglement. The +**Adversarial** system was shown to have the best speech synthesis quality from Chapter 5 evaluation, as well as the best intrinsic disentanglement from Chapter 6 analysis. The same system was compared to self-supervised and semi-supervised techniques for phone recognition and speaker diarization. It was also the main system used for content privacy as well as voice transformation and language code-switching. While disentanglement may be relative across systems, it can be said that the +**Adversarial** system variant is the most promising based on evaluation and analysis.

Chapter 8

Conclusion

8.1 Summary of Findings

This thesis has presented experiments that explore how to learn and use speech representations in a variety of tasks ranging from speech authenticity and naturalness assessment to multilingual speech synthesis and voice conversion. The purpose of this thesis was to understand *disentanglement* in the context of speech. While the term itself has been borrowed from research in the image domain (where objects *can* be fully segmented or factorized) the term has been applied inconsistently in speech research especially at the time that this thesis was conceived and written. To continue the analogy of disentanglement for image processing, it may be straightforward to segment objects in an image using edges, such as finding the boundary between a red-coloured segment and a white-coloured segment. For the same analogy applied to speech disentanglement, the objective is to take a pink-colored section and retrieve a representation of red and a separate representation of white. Each of the chapters in this thesis has addressed a different angle of speech representations and disentanglement. Some of the original motivation for speech disentanglement in this work was inspired by early work in the field of speaker recognition which models factors of variability between different speakers and within utterances of a particular speaker. Unlike speaker recognition, which seeks to *separate* and *discard* unwanted or irrelevant informational factors (channel, noise, session variability, etc) the objective in this thesis has been to *separate* and *retain* representations of informational factors.

Chapter 3: Methods to Estimate Speech Authenticity and Quality. The experiments in this chapter explored representation learning for detecting replayed speech in an automatic speaker verification (ASV) scenario and the technique was entered into the 2019 ASVspoof challenge. The learned representations (a special type of x-vector) explicitly modeled environmental factors such as room size and reverberation time as well as spoofing attack factors such as replay device quality. The proposed technique combined abstract speech representations with low-level signal features for training a convolutional neural network to predict spoofed vs. bonafide decisions for speech utterances. The speech replay detection technique resulted in a method that generalized well to unseen conditions and demonstrated consistent performance between very mismatched train and evaluation datasets. The work on speech replay detection was further expanded and similar representation learning techniques were applied to automatic estimation of synthetic speech naturalness mean-opinion scores (MOS) from human listening

judgements. Representation learning can help with MOS estimation, especially for preserving the relative quality among speakers for a given TTS system but the problem of automatic scoring for synthetic speech remains challenging. An automatic tool for MOS estimation was developed and made publicly available, as well as utilized for synthetic speech scoring in Chapter 5.

Chapter 4: Methods to Disentangle Representations of Speaker Identity. A new approach was proposed and tested for separating speaker identity and emotion/style information from existing state-of-the-art speaker representations such as x-vectors and i-vectors using autoencoders. The approach that was presented was also defined as a type of *disentanglement* because it did not discard information but rather moved information of one type into one representation (such as speaker information) and moved information of another type into a another representation (such as style/emotion information). The disentangled representations were created and evaluated at several levels of compression by altering the dimensionality of the latent space in the disentangling autoencoder. Evaluation found that disentanglement of speaker identity and style/emotion could be done with high dimensional representations (400-dim to 512-dim) as well as low-dimensional representations (50-dim to 100-dim). The approach was based on existing speaker representations (x-vectors and i-vectors) so any resulting disentanglement would be dependent on initial conditions of those representations, including the number of MFCCs that were used or the size of the neural network. Therefore Chapter 4 proposed to explore an end-to-end solution for disentanglement in Chapter 5.

Chapter 5: Methods for End-to-End Disentanglement. An end-to-end architecture with several variations was introduced in Chapter 5 based on the VQ-VAE paradigm and operating on directly on the speech waveform as input and output. The objective of using and modifying VQ-VAE was to encourage information disentanglement in the latent space (VQ codebooks) using self-supervised or semi-supervised learning. 10 variants of the system were developed using a combination of additional encoders and VQ codebooks, as well as additional classifiers for multi-task learning. The 10 system variants were compared to two existing VQ-VAE baseline systems for speech synthesis quality. The top-performing system for speech synthesis quality was identified and further developed for multilingual speech synthesis. Listening tests for speech naturalness as well as objective measures (F0 reconstruction, MOS estimation, speaker similarity, and ASR word error rate) showed that it is possible to learn separate representations of speaker identity and speech content for multi-speaker English data as well as multi-speaker multi-language data for English, French, German, and Italian.

Chapter 6: Intrinsic Analysis of Disentangled Representations The analysis presented in Chapter 6 set forth the objective to better understand if the learned representations from the top-performing systems in Chapter 5 were modeling targeted speech features (such as speaker identity and content) as well as whether the representations model only the targeted features. The analysis consisted of understanding how the VQ codebooks were utilized and the frequency distribution of codes. Several language modeling techniques were adapted to analyze sequences of VQ codes for the phone content as well as VQ codes for speaker identity. The language modeling analysis revealed that the VQ phone codebooks did not learn representations that correspond directly to speech phonemes. However, the VQ F0 codebooks did learn some representations that correspond to unvoiced/silence as well as male and female speaker genders. It was found that accent information tends to be correlated with the VQ speaker codebooks rather than the VQ phone codebooks. Overall the analysis revealed the inherent difficulty of quantifying and qualifying the meaning of disentanglement and it was suggested that disen-

tanglement be considered as a relative term. In the case that disentanglement is relative, one system was identified as having better disentanglement than another, and the system with better disentanglement was also the top-performing system from speech synthesis evaluation that was done in Chapter 5. It was also revealed that fine-tuning one of the proposed VQ-VAE variants from English-only to multi-language data resulted in entirely new VQ codebooks being learned for speakers as well as phones.

Chapter 7: Extrinsic Analysis of Disentangled Representations. Several tasks were explored in this chapter that utilized the learned VQ representations and those tasks included: phone recognition, speaker diarization, content masking for privacy, voice transformation, and language-switching. The phone recognition task revealed that the proposed VQ-VAE systems performed worse than a state-of-the-art audio baseline (using MFCCs) but all of the VQ-VAE systems made similar proportions of errors as the audio baseline (i.e., substitutions were most common and deletions were least common). All of the tested VQ-VAE system variants made similar types of errors in similar proportions as the audio baseline. It was found that the addition of a speaker codebook did not negatively impact phone recognition performance, by comparing one of the dual-encoder VQ-VAE models (from Chapter 5) with the original VQ-VAE model. The speaker diarization task revealed that the VQ codes from the proposed speaker codebooks (from Chapter 5) can be utilized for speaker diarization and perform as well as a traditional strong baseline based on x-vectors. Content privacy masking demonstrated how VQ phone codes can be manipulated in a way that conceals content information (specifically words) while speech is compressed in a latent representation. The VQ-VAE decoder (WaveRNN) is able to generate speech that is intelligible and retains speaker consistency throughout the utterance despite having some content masked with noise. The voice transformation task revealed that the learned VQ speaker codes can be used to transform voices between male and female. In addition, the VQ speaker codes can be mixed together to create new voices. The multi-language code-switching task showed that it is possible to use the VQ phone codebooks to generate utterances that contain content from more than one language in the same utterance. This is a step forward for multilingual speech synthesis, especially because the VQ phone codebook that was utilized in the experiments was a single codebook representing four languages.

8.2 Evidence for Disentanglement

The three principles of disentanglement that this thesis contributes are as follows:

1. The learned representations are sufficiently rich to model targeted speech features.

2. Representations that model targeted speech features do not also model non-target speech features (e.g., a representation of speaker identity does not also model speech content).

3. Representations have utility outside of the system that disentangles information.

For each of the three principles, evidence was found for and against disentanglement through rigorous experimentation. Each of the principles will be described here as well as a summary of which experimental outcomes support each principle.

In support of the first principle, it was found that there was high quality speech synthesis in seen and unseen conditions, high quality speech for voice conversion tasks, successful phone

recognition performance, and successful speaker diarization performance. The content masking task that was introduced in Chapter 7 also provides support for the first principle. Evidence against disentanglement for the first principle were that no one-to-one mapping between VQ codes and phones or speaker was found. Furthermore, high-frequency phones were not found to correlate with high-frequency VQ phone codes.

In support of the second principle, it was found that there was low KL-divergence for matched content conditions on VQ phone code probabilities when comparing distributions across different speakers. Similarly, there was high KL-divergence for unmatched content conditions on VQ phone code probabilities across different speakers. Both of these findings suggest that the VQ phone codes correctly model content information and not speaker information, and therefore support the second principle of disentanglement. The probing task should be interpreted carefully because KL-divergence alone may not provide a sufficiently strong model of the presence or absence of information. Therefore it is possible that the VQ phone codes do model speaker information but it is not captured by the probing task. Additionally, there is evidence against disentanglement for the second principle such as the same phone being represented by different sequences of VQ phone codes. Some of the differences in the sequences are due to the length or duration of a phone, but phones are also often represented by entirely different VQ phone codes (rather than repeating sequences). The VQ F0 codes correlate with speaker gender which can be expected to some extent, but this shows that the VQ F0 codes are not disentangled from speaker attributes. Finally the speaker vectors in VQ-VAE systems are often very similar to each other, suggesting that the learned representations are not modeling speaker uniquely.

In support of the third principle, this was demonstrated by extrinsic tasks in Chapter 7, specifically: phone recognition, speaker diarization, and voice conversion. Further the speaker vectors were found to correlate with accent information in Chapter 6. While the utility of the learned representations was shown in these tasks, several of the tasks continued to utilize the trained WaveRNN decoder so it is difficult to claim that the representations have utility completely outside of the trained systems.

8.3 Limitations

While the work in this thesis provides new insights about disentanglement and exciting avenues of future work, this thesis is not without some limitations. There were advancements in the field of speech processing related to self-supervised learning that arose after the experiments in Chapter 5 had already been started based on the VQ-VAE architecture. In particular, a new type of autoencoder has since been published called a variable-rate autoencoder [Dieleman et al., 2021] and a new representation learning technique has been proposed in an unpublished manuscript based on contrastive predictive coding (CPC) [van den Oord et al., 2018]. Both of these advances were developed at Google/DeepMind with a multitude of other engineering resources available. Generally, interest in speech disentanglement has been increasing during the past few years. Even with these recent advancements, there is no single definition or formalism that defines speech disentanglement. The speech community would benefit from such a development. The work presented in this thesis shows why it is so difficult to define disentanglement for speech. The VQ-VAE method for disentanglement from Chapter 5 may potentially lose information (which would falsely present as disentanglement) or the information may be redistributed into

other parts of the multi-task system (which may falsely present as lack of disentanglement). There may be some aspects to speech disentanglement that are fundamentally limiting such as the separation of speaker identity and speaking style, or the separation of phones and accent.

8.4 Future Directions

Given that the multi-language speech synthesis and representation learning was successful for voice transformation and language-switching, it would be interesting to explore an approach that learns discrete latent units for applications to speech-to-speech direct translation using waveforms. This research direction would require a lot of clean, parallel audio in the source and target languages as well as additional modules that model the translation aspect. It may be possible to translate between source and target languages within a learned latent space, such as VQ codebooks or similar representations from other speech synthesis architectures.

8.5 Final Remarks

The view of disentanglement presented in this thesis relies on the ability to evaluate learned representations either intrinsically or as they can be applied in downstream tasks. The approach to disentanglement that was presented in this thesis shows that disentanglement is a very difficult problem and there are some aspects of it that can only be assessed successfully with downstream tasks such as voice transformation and phone recognition. The principles of disentanglement that were explored in this thesis can be used as a guide for further work on speech representation learning especially for cases where learned representations may be obtained externally to the application technology, such as learning disentangled phone/speaker representations and then utilizing the speaker representations for diarization. Speech disentanglement is an inherently difficult task but experiments in this work show that disentanglement has a high utility for a variety of downstream tasks.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., and Others (2016). Tensorflow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283.

Akagi, M. and Irie, Y. (2012). Privacy Protection for Speech Based on Concepts of Auditory Scene Analysis. *Proceedings of the 41st International Congress and Exposition on Noise Control Engineering (InterNoise 2012), New York City, USA*, page 485.

Akuzawa, K., Iwasawa, Y., and Matsuo, Y. (2018). Expressive Speech Synthesis Via Modeling Expressions with Variational Autoencoder. *Proceedings of Interspeech 2018, Hyderabad, India*.

Albanie, S., Nagrani, A., Vedaldi, A., and Zisserman, A. (2018). Emotion Recognition in Speech Using Cross-modal Transfer in the Wild. In *Proceedings of the 26th ACM International Conference on Multimedia*, pages 292–301.

Amiriparian, S., Gerczuk, M., Ottl, S., Cummins, N., Freitag, M., Pugachevskiy, S., Baird, A., and Schuller, B. (2017). Snore Sound Classification Using Image-Based Deep Spectrum Features. In *Proceedings of Interspeech 2017, Stockholm, Sweden*, pages 3512–3516.

An, G. (1996). The Effects of Adding Noise During Backpropagation Training on a Generalization Performance. *Neural Computation*, 8(3):643–674.

Anjos, A., Günther, M., de Freitas Pereira, T., Korshunov, P., Mohammadi, A., and Marcel, S. (2017). Continuously Reproducing Toolchains in Pattern Recognition and Machine Learning Experiments. In *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, USA*.

Anjos, A., Shafey, L. E., Wallace, R., Günther, M., Mccool, C., and Marcel, S. (2012). Bob: A Free Signal Processing and Machine Learning Toolbox for Researchers. In *20th ACM Conference on Multimedia Systems (ACMMM), Nara, Japan*.

Arık, S. Ö., Chen, J., Peng, K., Ping, W., and Zhou, Y. (2018). Neural Voice Cloning With a Few Samples. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, pages 10040–10050.

Aryal, S. and Gutierrez-Osuna, R. (2014). Can Voice Conversion be Used to Reduce Non-native Accents? In *Proceedings of 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7879–7883.

Asami, T., Masumura, R., Masataki, H., and Sakauchi, S. (2014). Read and Spontaneous Speech Classification Based on Variance of GMM Supervectors. In *Proceedings of Interspeech 2014, Singapore*, pages 2375–2379.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460.

Bengio, Y. (2013). Deep Learning of Representations: Looking Forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37. Springer.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

Bishop, C. M. (1995). Training With Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1):108–116.

Blasi, D. E., Wichmann, S., Hammarström, H., Stadler, P. F., and Christiansen, M. H. (2016). Sound–Meaning Association Biases Evidenced Across Thousands of Languages. *Proceedings of the National Academy of Sciences*, 113(39):10818–10823.

Blue, L., Vargas, L., and Traynor, P. (2018). Hello, Is It Me You're Looking For?: Differentiating Between Human and Electronic Speakers for Voice Interface Security. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 123–133.

Burton, D. (1987). Text-dependent Speaker Verification Using Vector Quantization Source Coding. *IEEE Transactions on Speech and Audio Processing*, 35(2):133–143.

Busso, C., Bulut, M., Lee, C.-C., Kazemzadeh, A., Mower, E., Kim, S., Chang, J. N., Lee, S., and Narayanan, S. S. (2008). IEMOCAP: Interactive Emotional Dyadic Motion Capture Database. In *Proceedings of Language Resources and Evaluation (LREC)*, pages 335–359.

Cai, W., Chen, J., and Li, M. (2018). Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System. In *2018 IEEE Odyssey - The Speaker and Language Recognition Workshop, Les Sables d'Olonne, France*, pages 74–81.

Campbell, J. P. (1997). Speaker Recognition: A Tutorial. *Proceedings of the IEEE*, 85(9):1437–1462.

Casebeer, J., Vale, V., Isik, U., Valin, J.-M., Giri, R., and Krishnaswamy, A. (2021). Enhancing into the Codec: Noise Robust Speech Coding with Vector-Quantized Autoencoders. In *Proceedings of 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 711–715.

Chakroborty, S. and Saha, G. (2009). Improved Text-independent Speaker Identification Using Fused MFCC & IMFCC Feature Sets Based on Gaussian Filter. *International Journal of Signal Processing*, 5(1):11–19.

Chartsias, A., Joyce, T., Papanastasiou, G., Semple, S., Williams, M., Newby, D. E., Dharmakumar, R., and Tsaftaris, S. A. (2019). Disentangled Representation Learning in Cardiac Image Analysis. *Medical Image Analysis*, 58:101535.

Chen, F., Adcock, J., and Krishnagiri, S. (2008). Audio Privacy: Reducing Speech Intelligibility While Preserving Environmental Sounds. In *Proceedings of the 16th ACM International Conference on Multimedia*, pages 733–736.

Chen, S. F. and Goodman, J. (1999). An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech & Language*, 13(4):359–394.

Chen, Z. and Lin, Y. (2020). Improving x-vector and PLDA for Text-dependent Speaker Verification. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 726–730.

Chettri, B., Mishra, S., Sturm, B. L., and Benetos, E. (2018a). A Study On Convolutional Neural Network Based End-to-end Replay Anti-Spoofing. *arXiv preprint arXiv:1805.09164*.

Chettri, B., Sturm, B. L., and Benetos, E. (2018b). Analysing Replay Spoofing Countermeasure Performance Under Varied Conditions. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6.

Chung, J. S., Nagrani, A., and Zisserman, A. (2018). VoxCeleb2: Deep Speaker Recognition. *Proceedings of Interspeech 2018, Hyderabad, India*, pages 1086–1090.

Chung, Y.-A., Tang, H., and Glass, J. (2020). Vector-Quantized Autoregressive Predictive Coding. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 3760–3764.

Clark, R. A. J., Richmond, K., and King, S. (2005). Multisyn Voices From ARCTIC Data for the Blizzard Challenge. In *Proceedings of Interspeech 2005, Lisbon, Portugal*, pages 101–104.

Cooke, M., Mayo, C., Valentini-Botinhao, C., Stylianou, Y., Sauert, B., and Tang, Y. (2013). Evaluating the Intelligibility Benefit of Speech Modifications in Known Noise Conditions. *Speech Communication*, 55(4):572–585.

Cooper, E., Chang, A., Levitan, Y., and Hirschberg, J. (2016). Data Selection and Adaptation for Naturalness in HMM-Based Speech Synthesis. In *Proceedings of Interspeech 2016, San Francisco, USA*, pages 357–361.

Cooper, E. and Hirschberg, J. (2018). Adaptation and Frontend Features to Improve Naturalness in Found-Data Dynthesis. In *Proceedings of Speech Prosody 2018, Poznań, Poland*, pages 794–798.

Cooper, E., Huang, W.-C., Toda, T., and Yamagishi, J. (2021). Generalization Ability of MOS Prediction Networks. *arXiv preprint arXiv:2110.02635*.

Cooper, E., Lai, C.-I., Yasuda, Y., Fang, F., Wang, X., Chen, N., and Yamagishi, J. (2020). Zero-Shot Multi-Speaker Text-to-Speech with State-of-the-art Neural Speaker Embeddings. In *Proceedings of 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6184–6188.

Cooper, E. and Yamagishi, J. (2021). How Do Voices from Past Speech Synthesis Challenges Compare Today? In *Proceedings of the 11th ISCA Speech Synthesis Workshop (SSW11), Budapest, Hungary*, pages 183–188.

Croux, C. and Dehon, C. (2010). Influence Functions of the Spearman and Kendall Correlation Measures. *Statistical Methods & Applications*, 19(4):497–515.

Cummins, N., Amiriparian, S., Hagerer, G., Batliner, A., Steidl, S., and Schuller, B. W. (2017). An Image-Based Deep Spectrum Feature Representation for the Recognition of Emotional Speech. In *Proceedings of the 25th ACM International Conference on Multimedia*, pages 478–484.

Curelaru, F. (2018). Evaluation of the Standard i-vectors Based Speaker Verification Systems on Limited Data. In *2018 International Conference on Communications (COMM)*, pages 101–106.

Dehak, N. and Chollet, G. (2006). Support Vector GMMs for Speaker Verification. In *2006 IEEE Odyssey - The Speaker and Language Recognition Workshop, San Juan, PR, USA*, pages 1–4.

Dehak, N., Dehak, R., Kenny, P., Brümmer, N., Ouellet, P., and Dumouchel, P. (2009). Support Vector Machines Versus Fast Scoring in the Low-Dimensional Total Variability Space for Speaker Verification. In *Proceedings of Interspeech 2009, Brighton, United Kingdom*, pages 1559–1562.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-End Factor Analysis for Speaker Verification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Dehak, R., Dehak, N., Kenny, P., and Dumouchel, P. (2007). Linear and Non Linear Kernel GMM Supervector Machines for Speaker Verification. In *Proceedings of Interspeech 2007, Antwerp, Belgium*, pages 302–305.

Delgado, H., Todisco, M., Sahidullah, M., Evans, N., Kinnunen, T., Lee, K. A., and Yamagishi, J. (2018). ASVspoof 2017 Version 2.0: Meta-data Analysis and Baseline Enhancements. In *2018 IEEE Odyssey - The Speaker and Language Recognition Workshop, Les Sables d'Olonne, France*, pages 296–303.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. (2020). Jukebox: A Generative Model for Music. *arXiv preprint arXiv:2005.00341*.

Dieleman, S., Nash, C., Engel, J., and Simonyan, K. (2021). Variable-Rate Discrete Representation Learning. *arXiv preprint arXiv:2103.06089*.

Ding, S. and Gutierrez-Osuna, R. (2019). Group Latent Embedding for Vector Quantized Variational Autoencoder in Non-Parallel Voice Conversion. In *Proceedings of Interspeech 2019, Graz, Austria*, pages 724–728.

Dingemanse, M., Blasi, D. E., Lupyan, G., Christiansen, M. H., and Monaghan, P. (2015). Arbitrariness, Iconicity, and Systematicity in Language. *Trends in Cognitive Sciences*, 19(10):603–615.

Doddington, G. R., Przybocki, M. A., Martin, A. F., and Reynolds, D. A. (2000). The NIST Speaker Recognition Evaluation – Overview, Methodology, Systems, Results, Perspective. *Speech Communication*, 31(2-3):225–254.

Dong, C., Dong, Y., Li, J., and Wang, H. (2008). Support Vector Machines Based Text Dependent Speaker Verification Using HMM Supervectors. In *2008 IEEE Odyssey - The Speaker and Language Recognition Workshop, Stellenbosch, South Africa*.

Donley, J., Ritz, C., and Kleijn, W. B. (2016). Improving Speech Privacy in Personal Sound Zones. In *Proceedings of 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 311–315.

Dreschler, W. A., Verschuure, H., Ludvigsen, C., and Westermann, S. (2001). Icranoises: Artificial Noise Signals with Speech-Like Spectral and Temporal Properties for Hearing Instrument Assessment. *Audiology*, 40(3):148–157.

Dutta, S., Tripp, B., and Taylor, G. W. (2018). Convolutional Neural Networks Regularized by Correlated Noise. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 375–382.

Ebbers, J., Kuhlmann, M., Cord-Landwehr, T., and Haeb-Umbach, R. (2021). Contrastive Predictive Coding Supported Factorized Variational Autoencoder for Unsupervised Learning of Disentangled Speech Representations. In *Proceedings of 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3860–3864.

Ebbers, J., Kuhlmann, M., and Haeb-Umbach, R. (2020). Adversarial Contrastive Predictive Coding for Unsupervised Learning of Disentangled Representations. *arXiv preprint arXiv:2005.12963*.

Faundez-Zanuy, M. (2004). On the Vulnerability of Biometric Security Systems. *IEEE Aerospace and Electronic Systems Magazine*, 19(6):3–8.

Ferrer, L., Burget, L., Plchot, O., and Scheffer, N. (2012). A Unified Approach for Audio Characterization and its Application to Speaker Recognition. In *2012 IEEE Odyssey - The Speaker and Language Recognition Workshop, Singapore*.

Fong, J., Williams, J., and King, S. (2021). Analysing Temporal Sensitivity of VQ-VAE Subphone Codebooks. In *Proceedings of the 11th ISCA Speech Synthesis Workshop (SSW11), Budapest, Hungary*, pages 227–231.

Font, R. and Cano, M. J. (2017). Experimental Analysis of Features for Replay Attack Detection - Results on the ASVspoof 2017 Challenge. In *Proceedings of Interspeech 2017, Stockholm, Sweden*, pages 7–11.

Fu, S.-W., Tsao, Y., Hwang, H.-T., and Wang, H.-M. (2018). Quality-Net: An End-to-End Non-Intrusive Speech Quality Assessment Model Based on BLSTM. In *Proceedings of Interspeech 2018, Hyderabad, India*, pages 1873–1877.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial Training of Neural Networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Gârbacea, C., van den Oord, A., Li, Y., Lim, F. S., Luebs, A., Vinyals, O., and Walters, T. C. (2019). Low Bit-rate Speech Coding with VQ-VAE and a WaveNet Decoder. In *Proceedings of*

2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 735–739.

Garofolo, J. S. (1993). TIMIT Acoustic Phonetic Continuous Speech Corpus. *Linguistic Data Consortium, 1993*.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423.

Gersho, A. and Gray, R. M. (1991). *Vector Quantization and Signal Compression*, volume 159. Springer Science & Business Media.

Gibiansky, A., Arik, S., Diamos, G., Miller, J., Peng, K., Ping, W., Raiman, J., and Zhou, Y. (2017). Deep Voice 2: Multi-Speaker Neural Text-to-Speech. In *Advances in Neural Information Processing Systems*, volume 30, pages 2962–2970.

Goldman, J.-P., Honnet, P.-E., Clark, R., Garner, P. N., Ivanova, M., Lazaridis, A., Liang, H., Macedo, T., Pfister, B., Ribeiro, M. S., Wehrli, E., and Yamagishi, J. (2016). The SIWIS Database: A Multilingual Speech Database With Acted Emphasis. In *Proceedings of Interspeech 2016, San Francisco, USA*, pages 1532–1535.

Govind, D. and Prasanna, S. M. (2013). Expressive Speech Synthesis: A Review. *International Journal of Speech Technology*, 16(2):237–260.

Gries, S. T. and Ellis, N. C. (2015). Statistical Measures for Usage-Based Linguistics. *Language Learning*, 65(S1):228–255.

Griffin, D. and Lim, J. (1984). Signal Estimation from Modified Short-time Fourier Transform. *IEEE Transactions on Speech and Audio Processing*, 32(2):236–243.

Grzybowska, J. and Kacprzak, S. (2016). Speaker Age Classification and Regression Using i-vectors. In *Proceedings of Interspeech 2016, San Francisco, USA*, pages 1402–1406.

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit. *Nature*, 405(6789):947.

Hansen, J. H. and Hasan, T. (2015). Speaker Recognition by Machines and Humans: A Tutorial Review. *IEEE Signal Processing Magazine*, 32(6):74–99.

Hayashi, T. and Watanabe, S. (2020). DiscreTalk: Text-to-Speech as a Machine Translation Problem. *arXiv preprint arXiv:2005.05525*.

Heigold, G., Moreno, I., Bengio, S., and Shazeer, N. (2016). End-to-end Text-dependent Speaker Verification. In *Proceedings of 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5115–5119.

Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. (2018). Towards a Definition of Disentangled Representations. *arXiv preprint arXiv:1812.02230*.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2016). Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *preprint https://openreview.net/forum?id=Sy2fzU9gl*.

Himawan, I., Madikeri, S., Motlicek, P., Cernak, M., Sridharan, S., and Fookes, C. (2019). Voice Presentation Attack Detection Using Convolutional Neural Networks. In *Handbook of Biometric Anti-Spoofing*, pages 391–415. Springer.

Hinterleitner, F. (2017). Influencing Factors on Perceptual Quality. In *Quality of Synthetic Speech*, pages 69–100. Springer, Singapore.

Hinterleitner, F., Manolaina, C., and Möller, S. (2014). Influence of a Voice on the Quality of Synthesized Speech. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 99–104.

Hinterleitner, F., Neitzel, G., Möller, S., and Norrenbrock, C. (2011). An Evaluation Protocol for the Subjective Assessment of Text-to-Speech in Audiobook Reading Tasks. *Proceedings of Blizzard Challenge 2011, Turin, Italy*.

Hinterleitner, F., Norrenbrock, C., and Möller, S. (2013a). Is Intelligibility Still the Main Problem? A Review of Perceptual Quality Dimensions of Synthetic Speech. In *Proceedings of the 8th ISCA Speech Synthesis Workshop (SSW8), Barcelona, Spain*, pages 147–151.

Hinterleitner, F., Norrenbrock, C., and Möller, S. (2013b). Perceptual Quality Dimensions of Text-to-Speech Systems in Audiobook Reading Tasks. *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2013*, pages 44–49.

Hirschberg, J. (2000). A Corpus-Based Approach to the Study of Speaking Style. In *Prosody: Theory and Experiment*, pages 335–350. Springer.

Hodari, Z., Watts, O., Ronanki, S., and King, S. (2018). Learning Interpretable Control Dimensions for Speech Synthesis by Using External Data. In *Proceedings of Interspeech 2018, Hyderabad, India*, pages 32–36.

Hsu, C.-C., Hwang, H.-T., Wu, Y.-C., Tsao, Y., and Wang, H.-M. (2016). Voice Conversion from Non-parallel Corpora Using Variational Auto-encoder. In *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–6.

Hu, T.-Y., Shrivastava, A., Tuzel, O., and Dhir, C. (2020). Unsupervised Style and Content Separation by Minimizing Mutual Information for Speech Synthesis. In *Proceedings of 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3267–3271.

Huang, W.-C., Cooper, E., Yamagishi, J., and Toda, T. (2021). LDNet: Unified Listener Dependent Modeling in MOS Prediction for Synthetic Speech. *arXiv preprint arXiv:2110.09103*.

Huang, W.-C., Hwang, H.-T., Peng, Y.-H., Tsao, Y., and Wang, H.-M. (2018). Voice Conversion Based on Cross-Domain Features Using Variational Auto Encoders. In *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 51–55.

Huang, W.-C., Luo, H., Hwang, H.-T., Lo, C.-C., Peng, Y.-H., Tsao, Y., and Wang, H.-M. (2020). Unsupervised Representation Disentanglement Using cross domain features and adversarial learning in Variational Autoencoder Based Voice Conversion. *IEEE/ACM Transactions on Emerging Topics in Computational Intelligence*, 4(4):468–479.

Huang, W.-C., Wu, Y.-C., Lo, C.-C., Tobing, P. L., Hayashi, T., Kobayashi, K., Toda, T., Tsao, Y., and Wang, H.-M. (2019). Investigation of F0 Conditioning and Fully Convolutional Networks in Variational Autoencoder Based Voice Conversion. *Proceedings of Interspeech 2019, Graz, Austria*, pages 709–713.

Hwang, M. and Huang, X. (1992). Subphonetic Modeling with Markov States-Senone. In *Proceedings of 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 33–36 vol.1.

Inaguma, H., Kiyono, S., Duh, K., Karita, S., Soplin, N. E. Y., Hayashi, T., and Watanabe, S. (2020). ESPnet-ST: All-in-One Speech Translation Toolkit. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 302–311.

ISO/IEC JTC1 SC37 Biometrics (2017). Information Technology – Biometric Presentation Attack Detection – Part 3: Testing and Reporting. *International Organization for Standardization: Geneva, Switzerland*.

Janicki, A., Alegre, F., and Evans, N. (2016). An Assessment of Automatic Speaker Verification Vulnerabilities to Replay Spoofing Attacks. *Security and Communication Networks*, 9(15):3030–3044.

Jati, A. and Georgiou, P. (2019). Neural Predictive Coding Using Convolutional Neural Networks Toward Unsupervised Learning of Speaker Characteristics. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(10):1577–1589.

Jauk, I. (2017). Unsupervised Learning for Expressive Speech Synthesis. *PhD Thesis. Universitat Politècnica de Catalunya*.

Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Moreno, I. L., et al. (2018). Transfer Learning from Speaker Verification to Multi-Speaker Text-to-Speech Synthesis. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, pages 4485–4495.

Joyce, J. M. (2011). *Kullback-Leibler Divergence*, pages 720–722. Springer Berlin Heidelberg.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A., Dieleman, S., and Kavukcuoglu, K. (2018). Efficient Neural Audio Synthesis. In *International Conference on Machine Learning, Stockholm, Sweden*, pages 2410–2419.

Kamble, M., Tak, H., and Patil, H. (2018). Effectiveness of Speech Demodulation-Based Features for Replay Detection. In *Proceedings of Interspeech 2018, Hyderabad, India*, pages 641–645.

Kamble, M. R. and Patil, H. A. (2019). Analysis of Reverberation Via Teager Energy Features for Replay Spoof Speech Detection. In *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2607–2611.

Kawahara, H., Masuda-Katsuse, I., and De Cheveigne, A. (1999). Restructuring Speech Representations Using a Pitch-Adaptive Time–Frequency Smoothing and an Instantaneous-Frequency-Based F0 Extraction: Possible Role of a Repetitive Structure in Sounds. *Speech Communication*, 27(3-4):187–207.

Kenny, P. (2005). Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms. *CRIM, Montreal,(report) CRIM-06/08-13*, 14(28-29):2.

Kenny, P., Ouellet, P., Dehak, N., Gupta, V., and Dumouchel, P. (2008). A Study of Inter-Speaker Variability in Speaker Verification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 16(5):980–988.

Kim, J., Kim, S., Kong, J., and Yoon, S. (2020). Glow-tts: A generative flow for text-to-speech via monotonic alignment search. In *Advances in Neural Information Processing Systems*, volume 33, pages 8067–8077.

Kim, S., Hori, T., and Watanabe, S. (2017). Joint CTC-Attention Based End-to-End Speech Recognition Using Multi-Task Learning. In *Proceedings of 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4835–4839.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR), San Diego, USA*.

Kinnunen, T., Juvela, L., Alku, P., and Yamagishi, J. (2017). Non-parallel Voice Conversion Using i-vector PLDA: Towards Unifying Speaker Verification and Transformation. In *Proceedings of 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5535–5539.

Kinnunen, T., Lee, K. A., Delgado, H., Evans, N., Todisco, M., Sahidullah, M., Yamagishi, J., and Reynolds, D. A. (2018). t-DCF: A Detection Cost Function for the Tandem Assessment of Spoofing Countermeasures and Automatic Speaker Verification. In *2018 IEEE Odyssey - The Speaker and Language Recognition Workshop, Les Sables d'Olonne, France*, pages 312–319.

Kinnunen, T. and Li, H. (2010). An Overview of Text-independent Speaker Recognition: From Features to Supervectors. *Speech Communication*, 52(1):12–40.

Knight, W. R. (1966). A Computer Method for Calculating Kendall's Tau With Ungrouped Data. *Journal of the American Statistical Association*, 61(314):436–439.

Kobayashi, K., Toda, T., and Nakamura, S. (2018). Intra-Gender Statistical Singing Voice Conversion with Direct Waveform Modification Using Log-Spectral Differential. *Speech Communication*, 99(4):211–220.

Kondo, K. and Sakurai, H. (2014). Gender-Dependent Babble Maskers Created From Multi-Speaker Speech for Speech Privacy Protection. In *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 251–254.

Kuhn, R., Junqua, J.-C., Nguyen, P., and Niedzielski, N. (2000). Rapid Speaker Adaptation in Eigenvoice Space. *IEEE Transactions on Speech and Audio Processing*, 8(6):695–707.

Kwon, O., Jang, I., Ahn, C., and Kang, H.-G. (2019). Emotional Speech Synthesis Based on Style Embedded Tacotron2 Framework. In *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pages 1–4.

Lai, C.-I., Abad, A., Richmond, K., Yamagishi, J., Dehak, N., and King, S. (2019). Attentive Filtering Networks for Audio Replay Attack Detection. In *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6316–6320.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *Nature*, 521(7553):436.

Lee, K. A., Sadjadi, S. O., Li, H., and Reynolds, D. A. (2020). Two Decades into Speaker Recognition Evaluation - Are We There Yet? *Computer Speech & Language*, 61(Art no. 101058).

Leng, Y., Tan, X., Zhao, S., Soong, F., Li, X.-Y., and Qin, T. (2021). MBNET: MOS Prediction for Synthesized Speech with Mean-Bias Network. In *Proceedings of 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 391–395.

Li, M. and Narayanan, S. (2014). Simplified Supervised i-vector Modeling with Application to Robust and Efficient Language Identification and Speaker Verification. *Computer Speech & Language*, 28(4):940–958.

Li, Y., Swersky, K., and Zemel, R. (2015). Generative Moment Matching Networks. In *International Conference on Machine Learning, Lille, France*, pages 1718–1727.

Liu, L.-J., Ling, Z.-H., Jiang, Y., Zhou, M., and Dai, L.-R. (2018). WaveNet Vocoder with Limited Training Data for Voice Conversion. In *Proceedings of Interspeech 2018, Hyderabad, India*, pages 1983–1987.

Liu, M., Wang, L., Dang, J., Nakagawa, S., Guan, H., and Li, X. (2019). Replay Attack Detection Using Magnitude and Phase Information with Attention-Based Adaptive Filters. In *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6201–6205.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., and Song, L. (2017). Sphereface: Deep Hypersphere Embedding for Face Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 212–220.

Liu, Z. and Mak, B. (2020). Multi-Lingual Multi-Speaker Text-to-Speech Synthesis for Voice Cloning with Online Speaker Enrollment. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 2932–2936.

Lo, C.-C., Fu, S.-W., Huang, W.-C., Wang, X., Yamagishi, J., Tsao, Y., and Wang, H.-M. (2019). MOSNet: Deep Learning Based Objective Assessment for Voice Conversion. *Proceedings of Interspeech 2019, Graz, Austria.*

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2019). Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *International Conference on Machine Learning, Long Beach, USA*, pages 4114–4124.

Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2020a). A Sober Look at the Unsupervised Learning of Disentangled Representations and their Evaluation. *Journal of Machine Learning Research*, 21:1–62.

Locatello, F., Poole, B., Rätsch, G., Schölkopf, B., Bachem, O., and Tschannen, M. (2020b). Weakly-Supervised Disentanglement Without Compromises. In *International Conference on Machine Learning, (online attendance)*, pages 6348–6359.

López, J. V. E., Tóth, L., Hoffmann, I., Kálmán, J., Pákáski, M., and Gosztolya, G. (2019). Assessing Alzheimer's Disease from Speech Using the i-vector Approach. In *International Conference on Speech and Compute (SPECOM), Istanbul, Turkey*, pages 289–298.

Lorenzo-Trueba, J., Yamagishi, J., Toda, T., Saito, D., Villavicencio, F., Kinnunen, T., and Ling, Z. (2018). The Voice Conversion Challenge 2018: Promoting Development of Parallel and Nonparallel Methods. In *2018 IEEE Odyssey - The Speaker and Language Recognition Workshop, Les Sables d'Olonne, France*, pages 195–202.

Lorincz, B., Stan, A., and Giurgiu, M. (2021). An Objective Evaluation of the Effects of Recording Conditions and Speaker Characteristics in Multi-Speaker Deep Neural Speech Synthesis. *arXiv preprint arXiv:2106.01812*.

Lu, H., King, S., and Watts, O. (2013). Combining a Vector Space Representation of Linguistic Context with a Deep Neural Network for Text-to-Speech Synthesis. In *Proceedings of 8th ISCA Speech Synthesis Workshop (SSW8), Barcelona, Spain*, pages 261–265.

Luo, Y.-J., Agres, K., and Herremans, D. (2019). Learning Disentangled Representations of Timbre and Pitch for Musical Instrument Sounds Using Gaussian Mixture Variational Autoencoders. *Proceedings of International Society for Music Information Retrieval Conference, Delft, The Netherlands*, pages 746–753.

Luong, H.-T., Takaki, S., Henter, G. E., and Yamagishi, J. (2017). Adapting and Controlling DNN-Based Speech Synthesis Using Input Codes. In *Proceedings of 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4905–4909.

Luong, H.-T., Takaki, S., Kim, S., and Yamagishi, J. (2016). A DNN-Based Text-to-Speech Synthesis System Using Speaker, Gender, and Age Codes. *The Journal of the Acoustical Society of America*, 140(4):2962–2962.

Martinez, D., Plchot, O., Burget, L., Glembek, O., and Matějka, P. (2011). Language Recognition in i-vectors Space. In *Proceedings of Interspeech 2011, Florence, Italy*, pages 861–864.

Matrouf, D., Bonastre, J.-F., and Fredouille, C. (2006). Effect of Speech Transformation on Impostor Acceptance. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages 933–936.

Mayo, C., Clark, R., and King, S. (2005). Multidimensional Scaling of Listener Responses to Synthetic Speech. In *Interspeech 2005-Eurospeech: 9th European Conference on Speech Communication and Technology*, pages 1725–1728. International Speech Communication Association.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., and Sonderegger, M. (2017). Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. In *Proceedings of Interspeech 2017, Stockholm, Sweden*, pages 498–502.

Mixdorff, H., Pfitzinger, H. R., and Grauwinkel, K. (2005). Towards Objective Measures for Comparing Speaking Styles. In *Proceedings of SPECOM*, pages 131–134.

Morise, M., Yokomori, F., and Ozawa, K. (2016). WORLD: a Vocoder-Based High-quality Speech Synthesis System for Real-time Applications. *IEICE Transactions on Information and Systems*, 99(7):1877–1884.

Moro-Velazquez, L., Villalba, J., and Dehak, N. (2020). Using x-vectors to Automatically Detect Parkinson's Disease from Speech. In *Proceedings of 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1155–1159.

Morris, A. C., Maier, V., and Green, P. (2004). From WER and RIL to MER and WIL: Improved Evaluation Measures for Connected Speech Recognition. In *Eighth International Conference on Spoken Language Processing (ICSLP), Jeju Island, Korea*, pages 2765–2768.

Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. (2011). Natural Language Processing: An Introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.

Nagarsheth, P., Khoury, E., Patil, K., and Garland, M. (2017). Replay Attack Detection Using DNN for Channel Discrimination. In *Proceedings of Interspeech 2017, Stockholm, Sweden*, pages 97–101.

Nagrani, A., Chung, J. S., and Zisserman, A. (2017). VoxCeleb: A Large-Scale Speaker Identification Dataset. *Telephony*, 3:33–039.

Nautsch, A., Wang, X., Evans, N., Kinnunen, T. H., Vestman, V., Todisco, M., Delgado, H., Sahidullah, M., Yamagishi, J., and Lee, K. A. (2021). ASVspoof 2019: Spoofing Countermeasures for the Detection of Synthesized, Converted and Replayed Speech. *IEEE/ACM Transactions on Biometrics, Behavior, and Identity Science*, 3(2):252–265.

Ng, A. Y. (2004). Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, page 78.

Noé, P.-G., Mohammadamini, M., Matrouf, D., Parcollet, T., Nautsch, A., and Bonastre, J.-F. (2021). Adversarial Disentanglement of Speaker Representation for Attribute-Driven Privacy Preservation. *Proceedings of Interspeech 2021, Brno, Czech Republic*, pages 1902–1906.

Oplustil-Gallegos, P., Williams, J., Rownicka, J., and King, S. (2020). An Unsupervised Method to Select a Speaker Subset from Large Multi-Speaker Speech Synthesis Datasets. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 1758–1762.

Ostendorf, M., Price, P. J., and Shattuck-Hufnagel, S. (1995). The Boston University Radio News Corpus. *Linguistic Data Consortium*, pages 1–19.

Paseddula, C. and Gangashetty, S. V. (2018). Input Fusion of MFCC and SCMC Features for Acoustic Scene Classification using DNN. In *2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS)*, pages 13–17.

Patton, B., Agiomyrgiannakis, Y., Terry, M., Wilson, K., Saurous, R. A., and Sculley, D. (2016). AutoMOS: Learning a Non-Intrusive Assessor of Naturalness-of-Speech. *arXiv preprint arXiv:1611.09207*.

Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, 2(6):559–572.

Peri, R., Li, H., Somandepalli, K., Jati, A., and Narayanan, S. (2020a). An Empirical Analysis of Information Encoded in Disentangled Neural Speaker Representations. In *2020 IEEE Odyssey - The Speaker and Language Recognition Workshop, Tokyo, Japan*, pages 194–201.

Peri, R., Pal, M., Jati, A., Somandepalli, K., and Narayanan, S. (2020b). Robust Speaker Recognition Using Unsupervised Adversarial Invariance. In *Proceedings of 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6614–6618.

Pitrelli, J. F., Bakis, R., Eide, E. M., Fernandez, R., Hamza, W., and Picheny, M. A. (2006). The IBM Expressive Text-to-Speech Synthesis System for American English. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1099–1108.

Polyak, A., Adi, Y., Copet, J., Kharitonov, E., Lakhotia, K., Hsu, W.-N., Mohamed, A., and Dupoux, E. (2021). Speech ReSynthesis from Discrete Disentangled Self-Supervised Representations. *Proceedings of Interspeech 2021, Brno, Czech Republic*, pages 3615–3619.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011). The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU), Waikoloa, Hawaii, USA*, pages 1–4.

Prechelt, L. (1998). Early Stopping - But When? In *Neural Networks: Tricks of the Trade*, pages 55–69. Springer.

Qian, K., Zhang, Y., Chang, S., Hasegawa-Johnson, M., and Cox, D. (2020). Unsupervised Speech Decomposition via Triple Information Bottleneck. In *International Conference on Machine Learning, (online attendance)*, pages 7836–7846.

Rabiner, L. (1977). On the Use of Autocorrelation Analysis for Pitch Detection. *IEEE Transactions on Speech and Audio Processing*, 25(1):24–33.

Raj, D., Snyder, D., Povey, D., and Khudanpur, S. (2019). Probing the Information Encoded in x-vectors. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Singapore*, pages 726–733.

Ravanelli, M., Zhong, J., Pascual, S., Swietojanski, P., Monteiro, J., Trmal, J., and Bengio, Y. (2020). Multi-Task Self-Supervised Learning for Robust Speech Recognition. In *Proceedings of 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6989–6993.

Renals, S., Hain, T., and Bourlard, H. (2007). Recognition and Understanding of Meetings: The AMI and AMIDA projects. In *IEEE Workshop on Automatic Speech Recognition Understanding (ASRU), Kyoto, Japan*, pages 238–247.

Reynolds, D. A. (1995). Speaker Identification and Verification Using Gaussian Mixture Speaker Models. *Speech Communication*, 17(1-2):91–108.

Reynolds, D. A. and Rose, R. C. (1995). Robust Text-independent Speaker Identification Using Gaussian Mixture Speaker Models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83.

Ridgeway, K. (2016). A Survey of Inductive Biases for Factorial Representation-Learning. *arXiv preprint arXiv:1612.05299*.

Rix, A. W., Beerends, J. G., Hollier, M. P., and Hekstra, A. P. (2001). Perceptual Evaluation of Speech Quality (PESQ)- A New Method for Speech Quality Assessment of Telephone Networks and Codecs. In *Proceedings of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 749–752.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241.

Rothauser, E., Chapman, W., Guttman, N., Nordby, K., Silbiger, H., Urbanek, G., and Weinstock, M. (1969). IEEE Recommended Practice for Speech Quality Measurements. *IEEE Transactions on Audio and Electroacoustics*, 17(3):225–246.

Rownicka, J., Bell, P., and Renals, S. (2018). Analyzing Deep CNN-Based Utterance Embeddings for Acoustic Model Adaptation. In *2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece*, pages 235–241.

Rownicka, J., Bell, P., and Renals, S. (2019). Embeddings for DNN Speaker Adaptive Training. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Singapore*, pages 479–486.

Sahidullah, M., Kinnunen, T., and Hanilçi, C. (2015). A Comparison of Features for Synthetic Speech Detection. In *Proceedings of Interspeech 2015, Dresden, Germany*, pages 2081–2091.

Satt, A., Rozenberg, S., and Hoory, R. (2017). Efficient Emotion Recognition From Speech Using Deep Learning on Spectrograms. In *Proceedings of Interspeech 2017, Stockholm, Sweden*, pages 1089–1093.

Schoeffler, M., Bartoschek, S., Stöter, F.-R., Roess, M., Westphal, S., Edler, B., and Herre, J. (2018). webMUSHRA — A Comprehensive Framework for Web-Based Listening Tests. *Journal of Open Research Software*, 6(1).

Schröder, M. (2009). Expressive Speech Synthesis: Past, Present, and Possible Futures. In Tao, J. and Tan, T., editors, *Affective Information Processing*, pages 111–126. Springer.

Schröder, M., Charfuelan, M., Pammi, S., and Steiner, I. (2011). Open Source Voice Creation Toolkit for the Mary TTS Platform. In *Proceedings of Interspeech 2011, Florence, Italy*, pages 3253–3256.

Sell, G., Snyder, D., McCree, A., Garcia-Romero, D., Villalba, J., Maciejewski, M., Manohar, V., Dehak, N., Povey, D., Watanabe, S., and Others (2018). Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge. In *Proceedings of Interspeech 2018, Hyderabad, India*, pages 2808–2812.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-ryan, R., and Others (2018). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. In *Proceedings of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783.

Shiota, S., Villavicencio, F., Yamagishi, J., Ono, N., Echizen, I., and Matsui, T. (2016). Voice Liveness Detection for Speaker Verification Based on a Tandem SingleDouble Channel Pop Noise Detector. In *Proceedings of Interspeech 2016, San Francisco, USA*, pages 259–263.

Shor, J., Jansen, A., Maor, R., Lang, O., Tuval, O., de Chaumont Quitry, F., Tagliasacchi, M., Shavitt, I., Emanuel, D., and Haviv, Y. (2020). Towards Learning a Universal Non-Semantic Representation of Speech. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 140–144.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR), San Diego, USA*.

Sisman, B., Yamagishi, J., King, S., and Li, H. (2020). An Overview of Voice Conversion and Its Challenges: From Statistical Modeling to Deep Learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:132–157.

Snyder, D., Garcia-Romero, D., Povey, D., and Khudanpur, S. (2017). Deep Neural Network Embeddings for Text-independent Speaker Verification. In *Proceedings of Interspeech 2017, Stockholm, Sweden*, pages 999–1003.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., and Khudanpur, S. (2018). x-vectors: Robust DNN Embeddings for Speaker Recognition. In *Proceedings of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333.

Snyder, D., Ghahremani, P., Povey, D., Garcia-Romero, D., Carmiel, Y., and Khudanpur, S. (2016). Deep Neural Network-Based Speaker Embeddings for End-to-end Speaker Verification. In *2016 IEEE Spoken Language Technology Workshop (SLT), San Juan, PR, USA*, pages 165–170.

Soong, F. K., Rosenberg, A. E., Juang, B.-H., and Rabiner, L. R. (1987). Report: A Vector Quantization Approach to Speaker Recognition. *AT&T Technical Journal*, 66(2):14–26.

Stamatatos, E., Fakotakis, N., and Kokkinakis, G. (2000). Text Genre Detection Using Common Word Frequencies. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*, pages 808–814.

Stanton, D., Wang, Y., and Skerry-Ryan, R. (2018). Predicting Expressive Speaking Style from Text in End-to-End Speech Synthesis. In *2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece*, pages 595–602.

Steiner, I. and Le Maguer, S. (2018). Creating New Language and Voice Components for the Updated MaryTTS Text-to-Speech Synthesis Platform. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan*, pages 3171–3175.

Tachibana, H., Uenoyama, K., and Aihara, S. (2018). Efficiently Trainable Text-to-Speech System Based on Deep Convolutional Networks with Guided Attention. In *Proceedings of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Tanaka, K., Kameoka, H., Kaneko, T., and Hojo, N. (2019). WaveCycleGan2: Time-domain Neural Post-filter for Speech Waveform Generation. *arXiv preprint arXiv:1904.02892*.

Toda, T., Chen, L.-H., Saito, D., Villavicencio, F., Wester, M., Wu, Z., and Yamagishi, J. (2016). The Voice Conversion Challenge 2016. In *Proceedings of Interspeech 2016, San Francisco, USA*, pages 1632–1636.

Todisco, M., Delgado, H., and Evans, N. (2016). A New Feature for Automatic Speaker Verification Anti-Spoofing: Constant Q Cepstral Coefficients. In *2016 IEEE Odyssey - The Speaker and Language Recognition Workshop, Bilbao, Spain*, pages 283–290.

Todisco, M., Delgado, H., and Evans, N. (2017). Constant Q Cepstral Coefficients: A Spoofing Countermeasure for Automatic Speaker Verification. *Computer Speech & Language*, 100(45):516–535.

Todisco, M., Wang, X., Vestman, V., Sahidullah, M., Delgado, H., Nautsch, A., Yamagishi, J., Evans, N., Kinnunen, T., and Lee, K. A. (2019). ASVspoof 2019: Future Horizons in Spoofed and Fake Audio Detection. In *Proceedings of Interspeech 2019, Graz, Austria*.

Tomashenko, N., Srivastava, B. M. L., Wang, X., Vincent, E., Nautsch, A., Yamagishi, J., Evans, N., Patino, J., Bonastre, J.-F., Noé, P.-G., and Todisco, M. (2020). Introducing the VoicePrivacy Initiative. In *Proceedings of Interspeech 2020, Shanghai, China*.

Tomashenko, N., Wang, X., Vincent, E., Patino, J., Srivastava, B. M. L., Noé, P.-G., Nautsch, A., Evans, N., Yamagishi, J., O'Brien, B., et al. (2021). The VoicePrivacy 2020 Challenge: Results and Findings. *arXiv preprint arXiv:2109.00648*.

Tu, T., Chen, Y.-J., Liu, A. H., and Lee, H.-Y. (2020). Semi-Supervised Learning for Multi-Speaker Text-to-Speech Synthesis Using Discrete Speech Representation. In *Proceedings of Interspeech 2020, Shanghai, China*.

Vair, C., Colibro, D., Castaldo, F., Dalmasso, E., and Laface, P. (2006). Channel Factors Compensation in Model and Feature Domain for Speaker Recognition. In *2006 IEEE Odyssey - The Speaker and Language Recognition Workshop, San Juan, PR, USA*, pages 1–6.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. In *Proceedings of the 9th ISCA Speech Synthesis Workshop (SSW9), Sunnyvale, USA*, page 125.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017). Neural Discrete Representation Learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 6309–6318.

Van Segbroeck, M., Travadi, R., Vaz, C., Kim, J., Black, M. P., Potamianos, A., and Narayanan, S. S. (2014). Classification of Cognitive Load from Speech Using an i-vector Framework. In *Proceedings of Interspeech 2014, Singapore*, pages 751–755.

Variani, E., Lei, X., McDermott, E., Moreno, I. L., and Gonzalez-dominguez, J. (2014). Deep Neural Networks for Small Footprint Text-dependent Speaker Verification. In *Proceedings of 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4052–4056.

Verma, P. and Das, P. K. (2015). i-vectors in Speech Processing Applications: A Survey. *International Journal of Speech Technology*, 18(4):529–546.

Vogt, R. and Sridharan, S. (2008). Explicit Modelling of Session Variability for Speaker Verification. *Computer Speech & Language*, 22(1):17–38.

Wagner, P., Beskow, J., Betz, S., Edlund, J., Gustafson, J., Eje Henter, G., Le Maguer, S., Malisz, Z., Szekely, E., Tannander, C., and Vose, J. (2019). Speech Synthesis Evaluation — State-of-the-Art Assessment and Suggestion for a Novel Research Program. In *Proceedings of the 10th ISCA Speech Synthesis Workshop (SSW10), Vienna, Austria*, pages 105–110.

Wan, L., Wang, Q., Papir, A., and Moreno, I. L. (2018). Generalized End-to-end Loss for Speaker Verification. In *Proceedings of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4879–4883.

Wan, V. and Renals, S. (2003). SVMSVM: Support Vector Machine Speaker Verification Methodology. In *Proceedings of 2003 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2.

Wang, T., Tao, J., Fu, R., Yi, J., Wen, Z., and Zhong, R. (2020a). Spoken Content and Voice Factorization for Few-Shot Speaker Adaptation. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 796–800.

Wang, X., Takaki, S., and Yamagishi, J. (2019a). Neural Source-filter-Based Waveform Model for Statistical Parametric Speech Synthesis. In *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5916–5920.

Wang, X., Takaki, S., Yamagishi, J., King, S., and Tokuda, K. (2019b). A Vector Quantized Variational Autoencoder (VQ-VAE) Autoregressive Neural F0 Model for Statistical Parametric Speech Synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:157–170.

Wang, X., Yamagishi, J., Todisco, M., Delgado, H., Nautsch, A., Evans, N., Sahidullah, M., Vestman, V., Kinnunen, T., Lee, K. A., and Others (2020b). ASVspoof 2019: A Large-Scale Public Database of Synthesized, Converted and Replayed Speech. *Computer Speech & Language*, 64(2020):101114.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., and Others (2017a). Tacotron: Towards End-to-end Speech Synthesis. *Proceedings of Interspeech 2017, Stockholm, Sweden*, pages 4006–4010.

Wang, Y., Skerry-Ryan, R., Xiao, Y., Stanton, D., Shor, J., Battenberg, E., Clark, R., and Saurous, R. A. (2017b). Uncovering Latent Style Factors for Expressive Speech Synthesis. *arXiv preprint arXiv:1711.00520*.

Wang, Y., Stanton, D., Zhang, Y., Ryan, R.-S., Battenberg, E., Shor, J., Xiao, Y., Jia, Y., Ren, F., and Saurous, R. A. (2018). Style Tokens: Unsupervised Style Modeling, Control and Transfer in End-to-End Speech Synthesis. In *International Conference on Machine Learning, Stockholm, Sweden*, pages 5180–5189.

Watts, O., Eje Henter, G., Fong, J., and Valentini-Botinhao, C. (2019). Where Do the Improvements Come From in Sequence-to-Sequence Neural TTS? In *Proceedings of the 10th ISCA Speech Synthesis Workshop (SSW10), Vienna, Austria*, pages 217–222.

Wester, M., Wu, Z., and Yamagishi, J. (2016). Analysis of the Voice Conversion Challenge 2016 Evaluation Results. In *Proceedings of Interspeech 2016, San Francisco, USA*, pages 1637–1641.

Williams, J. and King, S. (2019). Disentangling Style Factors from Speaker Representations. In *Proceedings of Interspeech 2019, Graz, Austria*, pages 3945–3949.

Williams, J. and Rownicka, J. (2019). Speech Replay Detection with x-vector Attack Embeddings and Spectral Features. In *Proceedings of Interspeech 2019, Graz, Austria*, pages 1053–1057.

Williams, J., Rownicka, J., Oplustil-Gallegos, P., and King, S. (2020). Comparison of Speech Representations for Automatic Quality Estimation in Multi-Speaker Text-to-Speech Synthesis. *2020 IEEE Odyssey - The Speaker and Language Recognition Workshop, Tokyo, Japan*, pages 222–229.

Witkowski, M., Kacprzak, S., Zelasko, P., and Kowalczyk, K. (2017). Audio Replay Attack Detection Using High-frequency Features. In *Proceedings of Interspeech 2017, Stockholm, Sweden*, pages 82–86.

Wu, D.-Y., Chen, Y.-H., and Lee, H.-Y. (2020). VQVC+: One-Shot Voice Conversion by Vector Quantization and U-Net Architecture. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 4691–4695.

Wu, D.-Y. and Lee, H.-Y. (2020). One-Shot Voice Conversion by Vector Quantization. In *Proceedings of 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7734–7738.

Wu, X., Sun, L., Kang, S., Liu, S., Wu, Z., Liu, X., and Meng, H. (2018). Feature Based Adaptation for Speaking Style Synthesis. In *Proceedings of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5304–5308.

Wu, Z., Evans, N., Kinnunen, T., Yamagishi, J., Alegre, F., and Li, H. (2015a). Spoofing and Countermeasures for Speaker Verification: A Survey. *Speech Communication*, 66(2015):130–153.

Wu, Z., Kinnunen, T., Evans, N., Yamagishi, J., Hanilçi, C., Sahidullah, M., and Sizov, A. (2015b). ASVspoof 2015: the First Automatic Speaker Verification Spoofing and Countermeasures Challenge. In *Proceedings of Interspeech 2015, Dresden, Germany*, pages 2037–2041.

Wu, Z., Watts, O., and King, S. (2016). Merlin: An Open Source Neural Network Speech Synthesis System. In *Proceedings of the 9th ISCA Speech Synthesis Workshop (SSW9), Sunnyvale, USA*, pages 202–207.

Wu, Z., Yamagishi, J., Kinnunen, T., Hanilçi, C., Sahidullah, M., Sizov, A., Evans, N., Todisco, M., and Delgado, H. (2017). ASVspoof: the Automatic Speaker Verification Spoofing and Countermeasures Challenge. *IEEE Journal of Selected Topics in Signal Processing*, 11(4):588–604.

Xue, S., Abdel-Hamid, O., Jiang, H., Dai, L., and Liu, Q. (2014). Fast Adaptation of Deep Neural Network Based on Discriminant Codes for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1713–1725.

Yamagishi, J. and Kobayashi, T. (2007). Average-Voice-Based Speech Synthesis Using HMM-Based Speaker Adaptation and Adaptive Training. *IEICE Transactions on Information and Systems*, 90(2):533–543.

Yamagishi, J., Ling, Z., and King, S. (2008). Robustness of HMM-Based Speech Synthesis. In *Proceedings of Interspeech 2008, Brisbane, Australia*, pages 581–584.

Yamagishi, J., Usabaev, B., King, S., Watts, O., Dines, J., Tian, J., Guan, Y., Hu, R., Oura, K., Wu, Y.-J., and Others (2010). Thousands of Voices for HMM-Based Speech Synthesis–Analysis and Application of TTS Systems Built on Various ASR Corpora. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 18(5):984–1004.

Yamagishi, J., Veaux, C., and Macdonald, K. (2019). *CSTR VCTK Corpus: English Multi-Speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92)*. University of Edinburgh. The Centre for Speech Technology Research (CSTR).

Yuan, J., Brenier, J. M., and Jurafsky, D. (2005). Pitch Accent Prediction: Effects of Genre and Speaker. In *Proceedings of Eurospeech (2005)*, pages 1409–1412.

Zen, H., Agiomyrgiannakis, Y., Egberts, N., Henderson, F., and Szczepaniak, P. (2016). Fast, Compact, and High Quality LSTM-RNN Based Statistical Parametric Speech Synthesizers for Mobile Devices. *Proceedings of Interspeech 2016, San Francisco, USA*, pages 2273–2277.

Zen, H., Dang, V., Clark, R., Zhang, Y., Weiss, R. J., Jia, Y., Chen, Z., and Wu, Y. (2019). LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech. In *Proceedings of Interspeech 2019, Graz, Austria*, pages 1526–1530.

Zen, H., Senior, A., and Schuster, M. (2013). Statistical Parametric Speech Synthesis Using Deep Neural Networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7962–7966.

Zhang, G., Qin, Y., and Lee, T. (2020). Learning Syllable-level Discrete Prosodic Representation for Expressive Speech Generation. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 3426–3430.

Zhao, Y., Huang, W.-C., Tian, X., Yamagishi, J., Das, R. K., Kinnunen, T., Ling, Z., and Toda, T. (2020a). Voice Conversion Challenge 2020: Intra-lingual Semi-parallel and Cross-lingual Voice Conversion. *arXiv preprint arXiv:2008.12527*.

Zhao, Y., Li, H., Lai, C.-I., Williams, J., Cooper, E., and Yamagishi, J. (2020b). Improved Prosody from Learned F0 Codebook Representations for VQ-VAE Speech Waveform Reconstruction. In *Proceedings of Interspeech 2020, Shanghai, China*, pages 4417–4421.

Zhao, Z., Bao, Z., Zhao, Y., Zhang, Z., Cummins, N., Ren, Z., and Schuller, B. (2019). Exploring Deep Spectrum Representations via Attention-Based Recurrent and Convolutional Neural Networks for Speech Emotion Recognition. *IEEE Access*, 7:97515–97525.

Zhou, C., Horgan, M., Kumar, V., Vasco, C., and Darcy, D. (2018). Voice Conversion with Conditional SampleRNN. In *Proceedings of Interspeech 2018, Hyderabad, India*, pages 1973–1977.

Zipf, G. (1970). The Psycho-Biology of Language. An Introduction to Dynamic Philology. *Foundations of Language*, 6(4).