Unsupervised Learning of Disentangled Representations for Speech with Neural Variational Inference Models

by

Wei-Ning Hsu

B.S., National Taiwan University (2014)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author .....
Department of Electrical Engineering and Computer Science
May 21, 2018

Certified by .....
James R. Glass
Senior Research Scientist
Computer Science and Artificial Intelligence Laboratory
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Unsupervised Learning of Disentangled Representations for Speech with Neural Variational Inference Models

by

Wei-Ning Hsu

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2018, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Despite recent successes in machine learning, artificial intelligence is still far from matching human intelligence in many ways. Two important aspects are transferability and amount of supervision required. Take speech recognition for example: while humans can easily adapt to a new accent without explicit supervision (i.e., ground truth transcripts for speech of a new accent), current machine learning techniques still struggle with such a scenario. We argue that an essential component of human learning is unsupervised or weakly supervised representation learning, which transforms input signals to low dimensional representations that facilitate subsequent structured learning and knowledge acquisition.

In this thesis, we develop unsupervised representation learning frameworks for speech data. We start with investigating an existing variational autoencoder (VAE) model for learning latent representations, and derive novel latent space operations for speech transformation. The transformation method is applied to unsupervised domain adaptation problems, which addresses the transferability issues of supervised machine learning framework. We then extend the VAE models, and propose a novel factorized hierarchical variational autoencoder (FHVAE), which better models a generative process of sequential data, and learns not only disentangled, but also interpretable latent representations without any supervision. By leveraging the interpretability, we demonstrate that such representations can be applied to a wide range of tasks, including but not limited to: voice conversion, denoising, speaker verification, speaker invariant phonetic feature extraction, and noise invariant phonetic feature extraction. In the last part of this thesis, we examine scalability issues regarding the original FHVAE training algorithm in terms of runtime, memory, and optimization stability. Based on our analysis, we propose a hierarchical sampling algorithm for training, which enables training of FHVAE models on arbitrarily large datasets.

Thesis Supervisor: James R. Glass
Title: Senior Research Scientist

Computer Science and Artificial Intelligence Laboratory

Acknowledgments

I was extremely lucky to be admitted to MIT and provided the opportunities to meet all these amazing people here. Doing research is never easy without the support from the others. There are too many people I would like to thank to for being part of this journey, witnessing my struggle as well as my growth. First and foremost, I would like to thank my advisor, Jim Glass. He gave me great freedom to explore whatever I found interesting, and also guided me at a higher level so I did not lose my direction. I still remebered our conversation during our trip to Shanghai, where Jim mentioned that mentoring students is not just about producing research papers, but also about helping the student to grow in every other aspect. I was very lucky to have Jim as my thesis advisor, and was grateful to his education.

I would also like to thank my lab mates in the Spoken Language System group, who are not only very talented, but also very easy to get along with. As an international student, language was a huge barrier for me to become social, which made me somewhat nervous when first coming to the group. However, I found our group was extremely welcoming and helpful. I soon felt like being part of it. Particulary, I would also like to thank Yu Zhang. He has been my main collaborator since I joined the group, and got me up to speed from knowing nothing about neural networks. He is very knowledgeable and I enjoyed our collaboration a lot.

To my roommates, friends in ROCSA, band members in JAM-soul, thank you for enriching my life. I have always believed that better life can lead to better work. My part of life outside of doing research is just as important as my research.

I would like to thank my family, they are the every reason that I can be here today. My grandparents, My parents, Chi-Hung (Thomas) and Tse-Fen (Connie), and my brother Shao-Ning, have unconditionally supported me through my entire life. Instead of ever asking me to follow the order and complete any task mechanically, my parents have always encouraged me to think more and make decisions myself, which is basically what the mindset of a researcher is supposed to be. My gratitude to my parents' love and education is beyond words.

To Chien-Yu (Charlotte), thank you for keeping me company all these years and putting up my with emotion when things were not going well. You know I am not good at expressing myself, but I am glad to have you with me.

This work was sponsored by the TUSA fellowship and PingAn.

Bibliographic Note

Portions of this thesis are based on peer-reviewed publications. Chapter 2 was published in Hsu et al. (2017a). Chapter 3 was published in Hsu et al. (2017c). Chapter 4 and Chapter 5 was published in Hsu et al. (2017b) and Hsu and Glass (2018a). Chapter 6 was published in Hsu and Glass (2018b).

Part of the code in this thesis is available at `https://github.com/wnhsu`.

Contents

1 Introduction **23**

1.1 On the Importance of Representation Learning 23

1.2 Contributions 25

2 Learning Representations with Variational Autoencoders **27**

2.1 Variational Autoencoders for Speech 28

2.1.1 Introduction to Variational Autoencoders 28

2.1.2 A Convolutional Model Architecture 29

2.2 Interpreting Learned Representations 31

2.2.1 Latent Attribute Representations and Derivation 31

2.2.2 Arithmetic Operations to Modify Speech Attributes 32

2.3 Data, Experiments, and Discussion 32

2.3.1 Data and Preprocessing 32

2.3.2 Experiment Setups 33

2.3.3 Sampling from the Learned Generative Process 33

2.3.4 Decoding Latent Phoneme Representation 35

2.3.5 Speech Transformation and Voice Conversion 35

2.3.6 Speech Interpolation 38

3 Unsupervised Adaptation via VAE-Based Data Augmentation **41**

3.1 A Sequence-to-Sequence Recurrent VAE 42

3.2 VAE-Based Data Augmentation 43

3.2.1 Latent Nuisance Representations 43

3.2.2 Data Augmentation Pipeline Overview 44
3.2.3 Type I: Nuisance Factor Replacement 44
3.2.4 Type II: Latent Nuisance Subspace Perturbation 45
3.3 Setup 48
3.3.1 Dataset 48
3.3.2 VAE Setup and Training 49
3.3.3 ASR Setup and Training 49
3.4 Results and Discussion 50
3.4.1 Baselines 51
3.4.2 Replacing Nuisance Attributes 52
3.4.3 Correctness of Soft Latent Nuisance Subspace Perturbation . . 53
3.4.4 Effect of Perturbation Ratios 54
3.4.5 Effect of Augmented Dataset Size 54
3.4.6 Comparing with DDA on Aurora-4 55
3.5 Conclusions 56

4 Learning Disentangled and Interpretable Representations 57
4.1 Factorized Hierarchical Variational Autoencoder 58
4.1.1 A Factorized Hierarchical Generative Process 58
4.1.2 An Inference Model 61
4.1.3 An Unsupervised Discriminative Objective 63
4.1.4 Inferring S-Vectors During Testing 64
4.1.5 Sequence-to-Sequence Autoencoder Model Architecture 64
4.2 Experimental Setup 66
4.2.1 Datasets and Surface Representations 66
4.2.2 FHVAE Model and Training Configurations 66
4.3 Comparison of FHVAE Model Architectures 67
4.4 Visualizing Latent Space Factorization 68
4.4.1 Re-combining Latent Segment and Sequence Variables 68
4.4.2 Walking in the Latent Space 70

4.5 Conclusions 71

5 Applications of Disentangled and Interpretable Representations 79

5.1 Speech Transformation 79

5.1.1 Denoising 80

5.1.2 Voice Conversion 81

5.2 Speaker Verification 82

5.3 Extracting Domain Invariant Features 84

5.3.1 Robustness to Speaker Variation 86

5.3.2 Robustness to Noise and Channel Variation 87

5.4 Study of FHVAE Architecture for ASR Feature Extraction 88

5.4.1 Baseline 89

5.4.2 Comparing Model Architectures 90

5.4.3 Effect of FHVAE Discriminative Training 90

5.4.4 Choice of Sequence Label 91

5.4.5 Use of S-Vector 91

5.4.6 Verifying Results on CHiME4 92

6 Scalable Factorized Hierarchical Variational Autoencoder Training 93

6.1 Limitations of the Original FHVAE training 94

6.1.1 Original FHVAE Training 94

6.1.2 Scalability Issues 96

6.2 Training with Hierarchical Sampling 98

6.3 Experimental Setup 99

6.3.1 Datasets 99

6.3.2 Training and Model Configurations 100

6.4 Results and and Discussion 102

6.4.1 Time and Memory Complexity 102

6.4.2 Evaluating Disentanglement Performance 102

6.5 Conclusions 106

7 Conclusion and Future Work **109**

7.1 Summary of Contributions 109

7.2 Future Work 111

A Derivation of Sequence Variational Lower Bound **115**

B Derivation of the Inferred S-Vector **119**

List of Figures

2-1 Illustration of the convolutional VAE architecture. 30

2-2 Random samples drawn from models trained with syllable-level and word-level dataset. The segments in (a) are 200ms, and the segment in (b) is 1s. 34

2-3 Comparison of sum of off-diagonal covariance scales for each dimension for the syllable and word-level dataset. 34

2-4 Comparison between VAE, AE and Fbank on averaging representations of /ae/, /th/, and /n/ from left to right. Each segment is 200ms long. 35

2-5 Cosine similarities of latent attribute representations. 36

2-6 Modify the phone from /aa/ (top) to /ae/ (bottom). Each segment is 200ms long. 37

2-7 Modify from a female (top) to a male (bottom). Each segment is 200ms long. We can observe that the vertical spacing between horizontal stripes are denser in the examples on the bottom, indicating that the pitch is lower in those examples. However the conversion is not perfect, because harmonics are not uniformly spread. 37

2-8 Interpolation in the latent space using VAE and AE. Each segment is 200ms long. 39

3-1 Illustration of the Sequence-to-Sequence LSTM VAE architecture. . . 43

3-2 Flowchart of generating transformed labeled data. 45

3-3 Two examples of replacing the nuisance attributes. 46

3-4 Eigenvalues of PCA analysis on latent nuisance representations in descending order. 47

3-5 An example of perturbing latent nuisance attributes. In this example, the transformed utterance contains more background noise and is of a higher pitch. 48

4-1 Graphical explanation of sequence-level and segment-level attributes distribution for two different utterances. Each dot denotes the value of a particular attribute of a segment. Distributions of some attributes of a segment, such as the phonetic content, do not vary between utterances, as shown on the lower left of the figure. On the other hand, distributions of other attributes of a segment, such as the fundamental frequency, have sequence-dependent distributions, as shown on the lower right of the figure. We want to encode the first type of attributes into one set of latent variables ($\boldsymbol{z}_1$), and the second type of attributes into the other set of latent variables ($\boldsymbol{z}_2$). 59

4-2 Graphical illustration of the proposed generative model. Grey nodes denote the observed variables, and white nodes are the hidden variables. 59

4-3 Graphical illustration of the proposed inference model. Grey nodes denote the observed variables, and white nodes are the hidden variables. 61

4-4 Sequence-to-sequence factorized hierarchical variational autoencoder. Dashed lines indicate the sampling process using the reparameterization trick (Kingma and Welling, 2013). The encoders for $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are pink and amber, respectively, while the decoder for $\boldsymbol{x}$ is blue. Darker colors denote the recurrent neural networks, while lighter colors denote the fully-connected layers predicting the mean and log variance. . . . 65

4-5 Three examples from different speakers. Within each example, from left to right are 1) the original segment, 2) FC reconstructed segment, and 3) LSTM reconstructed segment. The leftmost images show expanded views of the higher frequency harmonic structure (horizontal dark bands) of the spectrogram suggesting that the LSTM reconstruction is superior to the FC model. 68

4-6 (left) Examples generated by varying different latent variables of a FHVAE model trained with $\alpha = 10$ on TIMIT dataset. The green block 'A' contains four reconstructed examples. The red block 'B' contains ten original examples on the first row and the corresponding reconstructed examples on the second row. The entry on the i-th row and the j-th column in the blue block 'C' is the reconstructed example using the latent segment variable $\boldsymbol{z}_1$ of the i-th row from the block 'A' and the latent sequence variable $\boldsymbol{z}_2$ of the j-th column from the block 'B.' (right) An illustration of harmonics and formants in filter bank images. 69

4-7 Examples generated by varying z_1 and z_2 of an FHVAE model trained with $\alpha = 0$ on Aurora-4 dataset. The green block 'A' and the red block 'B' contain the same eight examples from the test set. In block 'B,' original examples are shown in the first row and the corresponding reconstructed examples are shown in the second row. The entry on the i-th row and the j-th column in the blue block 'C' is the reconstructed example using the latent segment variable $\boldsymbol{z}_1$ of the i-th row from the block 'A' and the latent sequence variable $\boldsymbol{z}_2$ of the j-th column from the block 'B.' 72

4-8 Examples generated by varying z_1 and z_2 of an FHVAE model trained with $\alpha = 10$ on Aurora-4 dataset. The green block 'A' and the red block 'B' contain the same eight examples from the test set. In block 'B,' original examples are shown in the first row and the corresponding reconstructed examples are shown in the second row. The entry on the i-th row and the j-th column in the blue block 'C' is the reconstructed example using the latent segment variable $\boldsymbol{z}_1$ of the i-th row from the block 'A' and the latent sequence variable $\boldsymbol{z}_2$ of the j-th column from the block 'B.' 73

4-9 Traversing two different dimensions in the space of latent **segment** variables with five seed segments from the TIMIT test set using an FHVAE model trained on TIMIT with $\alpha = 0$. 74

4-10 Traversing another two different dimensions in the space of latent **segment** variables with five seed segments from the TIMIT test set using an FHVAE model trained on TIMIT with $\alpha = 0$. 75

4-11 Traversing two different dimensions in the space of latent **sequence** variables $\boldsymbol{z}_2$ with five seed segments from the TIMIT test set using an FHVAE model trained on TIMIT with $\alpha = 0$. 76

4-12 Traversing another two different dimensions in the space of latent **sequence** variables $\boldsymbol{z}_2$ with five seed segments from the TIMIT test set using an FHVAE model trained on TIMIT with $\alpha = 0$. 77

5-1 Graphical illustration of speech transformation by manipulating latent sequence variable of the target sequence. 80

5-2 FHVAE ($\alpha = 0$) decoding results of three combinations of *latent segment variables* z_1 and *latent sequence variables* z_2 from two utterances in Aurora-4: a clean one (top-left) and a noisy one (bottom-left). FHVAEs learn to encode local attributes, such as linguistic content, into z_1, and encode global attributes, such as noise level, into z_2. Therefore, by replacing z_2 of a noisy utterance with z_2 of a clean utterance, an FHVAE decodes a denoised utterance (middle-right) that preserves the linguistic content. Reconstruction results of the clean and noisy utterances are also shown on the right. Audio samples are available at `https://youtu.be/naJZITvCfI4`. 81

5-3 FHVAE ($\alpha = 0$) decoding results of three combinations of *latent segment variables* z_1 and *latent sequence variables* z_2 from one clean utterance (top-left) and one utterance with car noise (bottom-left) in Aurora-4. By replacing z_2 of a noisy utterance with z_2 of a clean utterance, an FHVAE decodes a denoised utterance (middle-right) that preserves the linguistic content. Audio samples are available at `https://youtu.be/pOP2DVZWRjM`. 82

5-4 FHVAE ($\alpha = 0$) decoding results of three combinations of *latent segment variables* z_1 and *latent sequence variables* z_2 from one male-speaker utterance (top-left) and one female-speaker utterance (bottom-left) in Aurora-4. By replacing z_2 of a male-speaker utterance with z_2 of a female-speaker utterance, an FHVAE decodes a voice-converted utterance (middle-right) that preserves the linguistic content. Audio samples are available at `https://youtu.be/VMX3IZYWYdg`. 83

5-5 FHVAE ($\alpha = 0$) decoding results of three combinations of *latent segment variables* z_1 and *latent sequence variables* z_2 from one female-speaker utterance (top-left) and one male-speaker utterance (bottom-left) in Aurora-4. By replacing z_2 of a female-speaker utterance with z_2 of a male-speaker utterance, an FHVAE decodes a voice-converted utterance (middle-right) that preserves the linguistic content. Audio samples are available at `https://youtu.be/Rurj2ByNRs8`. 84

6-1 Histogram of $\log \sum_{i=1}^{M} p(\bar{\boldsymbol{z}}_2^{(i,n)} | \bar{\boldsymbol{\mu}}_2^{(j)})$ with respect to different $M \in \{10^1, 10^2, 10^3, 10^4, 10^5\}$. Distributions shift by roughly a constant when M increases by 10 times, implying the denominator scales proportionally to M. 97

6-2 The proposed FHVAE architecture consists of two encoders (orange and green) and one decoder (blue). $\boldsymbol{x} = [x_1, \cdots, x_{20}]$ is a segment of 20 frames. Dotted lines in the encoders denote sampling from parametric distributions. 101

6-3 Scatter plots of t-SNE projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ with models trained on TIMIT. Each point represents one segment. Different colors are used to code segments of different labels with respect to the generating factor shown at the title of each plot. 104

6-4 Scatter plots of t-SNE projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ with models trained on Aurora-4. Each point represents one segment. Different colors are used to code segments of different labels with respect to the generating factor shown at the title of each plot. 105

6-5 Scatter plots of t-SNE projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ with models trained on AMI. Each point represents one segment. Different colors are used to code segments of different labels with respect to the generating factor shown at the title of each plot. 106

6-6 Scatter plots of t-SNE projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ with models trained on LibriSpeech. Each point represents one segment. Different colors are used to code segments of different labels with respect to the generating factor shown at the title of each plot. 107

6-7 Results of decoding re-combined latent variables. A segment in the $\boldsymbol{x}^C$ block is generated conditioned on the latent segment variable of a segment in the block $\boldsymbol{x}^A$ of the same column, and conditioned on the latent sequence variable of a red-box highlighted segment in the block $\boldsymbol{x}^B$ of the same row. 108

List of Tables

2.1 Inference model architecture. *Conv* refers to convolutional layers, *Reshape* convert a 3-D tensor of shape $(1, T/4, 256)$ to a vector, *Fc* refers to fully connected layers, and *Gauss* refers to the Gaussian parametric layer predicting mean and log variance of $q(\boldsymbol{z}|\boldsymbol{x})$ 30

2.2 Generative model architecture. *Fc* refers to fully connected layers, *Reshape* converts a vector to a 3-D tensor of shape $(1, T/4, 256)$, *T-Conv* refers to transposed convolutional layers, and *Gauss* refers to the Gaussian parametric layer predicting mean and log variance of $p(\boldsymbol{x}|\boldsymbol{z})$ 30

2.3 Average posteriors over 10 instances of source, target, and fixed attributes before and after modification. 38

3.1 CHiME-4 development set word error rate of acoustic models trained on different augmented sets. 52

3.2 Aurora-4 test_eval92 set word error rate of acoustic models trained on different augmented sets. 56

4.1 TIMIT test set segment variational lower bound results on different model architectures. 67

5.1 Comparison of speaker verification equal error rate (EER) on the TIMIT test set 85

5.2 TIMIT test phone error rate of acoustic models trained on different features and sets 86

5.3 Aurora-4 test word error rate of acoustic models trained on different features and sets 87
5.4 Aurora-4 test_eval92 set word error rate of acoustic models trained on different features. 89
5.5 CHiME-4 development set word error rate of acoustic models trained on different features. 92

6.1 Family of distributions adopted for FHVAE generative and inference models. 94
6.2 Processing time of the optimization step with different sequence batch size K. 102

Chapter 1

Introduction

1.1 On the Importance of Representation Learning

To measure how close artificial intelligence is to human intelligence, one often compares how well machines can do on a certain set of tasks with human performance. Thanks to the recent advances in machine learning, and especially deep learning, today's state-of-the-art machine learning models can achieve human-level performance on quite a few challenging recognition tasks, such as speech recognition (Erdogan et al., 2016; Xiong et al., 2016; Saon et al., 2017) and image recognition (Simonyan and Zisserman, 2014; He et al., 2016). With such achievements, can we say that the gap between artificial intelligence and human intelligence is closed? The answer is definitely "No!"

These state-of-the-art models are built within a supervised learning framework, which often requires a significant amount of labeled data for training. Compared to human learning, these models are extremely data inefficient, and vulnerable to domain change. Take automatic speech recognition (ASR) systems for example: commercial systems are often trained on tens of thousands of hours of annotated data (Li et al., 2017; Chiu et al., 2017). In addition, if we take a model trained on only clean speech, and evaluate its performance on noisy data, the performance often degrades significantly (Sun et al., 2017; Meng et al., 2017; Hsu et al., 2017c; Hsu and Glass, 2018a). Apparently, there is considerable differences between how humans learn and

the current supervised ASR learning framework.

Here we highlight two differences, which are transferability and level of supervision. Transferability refers to the ability of leveraging the knowledge acquired from learning one task to facilitate the learning of a new task. On the other hand, for the scenario of recognition or classification tasks, level of supervision indicates how much information about the prediction target is provided during the learning process. Take machine translation for example: an analogous task for humans is second language acquisition. Instead of learning by reading millions of pairs of parallel sentences in two different symbolic systems, which is how current machine translation systems are trained (Wu et al., 2016), what people do to acquire a second language is to build connections between a symbol in the new symbolic system with some previously learned concept. The abstract space of concepts serves as a bridge to connect different symbolic systems, and translation between languages can therefore be achieved.

The above process can also be regarded as representation learning, where elements in a new symbolic system are represented using learned concepts. In other words, suppose the set of learned concepts forms an abstract space, then the process can also be viewed as encoding a symbol to a point in that space. With that being achieved, knowledge previously learned about the abstract concept space can be transferred to the new symbolic system. For example, a learned relationship between two concepts can apply to the two symbols associated with those two concepts.

An important aspect of human learning is to create abstractions of what we perceive, and to build our knowledge upon those abstractions. We believe that to make machines learn like humans do, it is essential to design frameworks and algorithms for representation learning by utilizing unsupervised or weakly supervised data, such as paired raw visual and raw audio data. In this thesis, we work toward this goal by focusing on representation learning from one of the most information-rich and natural raw sensory input: speech. Speech waveforms have complex distributions that exhibit high variance due to factors that include linguistic content, speaking style, dialect, speaker identity, emotional state, environment, channel effects, etc. Understanding the influence of these factors on the speech signal is an important problem, which can

be used for a wide variety of applications, including, but not limited to adaptation and data augmentation for speech recognition (Jaitly and Hinton, 2013; Cui et al., 2015), voice conversion (Kain and Macon, 1998; Stylianou, 2009; Toda et al., 2006), and speech compression (Wong et al., 1983). However, most previous research has focused on handcrafting features to capture these factors, unsupervised learning of these factors is rather unexplored or else is task-oriented.

1.2 Contributions

In this thesis, we propose an unsupervised representation learning framework for speech data. To start with, in Chapter 2, we adopt a simple yet expressive neural network-based latent variable model to discover latent generating factors of speech, and use those generating factors as a representation of speech. A simple and effective method is proposed for connecting physical attributes with learned latent representations. We apply the learned representations derived from this method for unsupervised voice conversion, and data augmentation for unsupervised domain adaptation in Chapter 3.

In the second part of this thesis, we propose a novel factorized hierarchical latent variable model in Chapter 4, which better captures the nature of the hierarchical generative process of sequential data. The proposed model is capable of learning disentangled and interpretable representations of speech data, by encoding sequence-level generating factors, such as speaker and channel, and segment-level generating factors, such as phonetic content, into different sets of latent variables. We then apply the learned disentangled representations to a number of speech processing tasks in Chapter 5, including voice conversion, denoising, robust speech recognition, and speaker verification. On all these tasks, we demonstrate very strong results compared to the baselines that adopt traditional features, showing the importance of representation learning and the transferability of knowledge from unsupervised learning tasks to supervised learning tasks.

The last part of this thesis addresses the scalability issues of the original training

algorithms for the proposed factorized hierarchical latent variable model. In Chapter 6, a hierarchical sampling algorithm for training is proposed to solve memory, runtime, and optimization issue, and is verified on a wide spectrum of datasets, ranging from three hours to 1,000 hours. Lastly, we conclude this thesis and discuss about future work in Chapter 7.

Chapter 2

Learning Representations with Variational Autoencoders

Recently, there has been significant interest in deep probabilistic generative models, such as Variational Autoencoders (VAEs) (Kingma and Welling, 2013; Mnih and Gregor, 2014) and Generative Adversarial Nets (GANs) (Goodfellow et al., 2014). These models are extremely expressive latent variable models, which often utilize neural networks to capture the complex nonlinear relationship between variables. However, due to such complexity, an exact posterior inference of latent variables is often intractable. Variational Bayes is a widely applied technique that addresses the intractability issue by introducing an alternative inference model, which aims to approximate the true posterior distribution. Variational autoencoders leverage this method to carry out parametric and amortized variational inference in an encoder-decoder framework.

In this chapter, we adopt the VAE framework and propose a convolutional architecture to model the probabilistic generative process of speech for learning latent representations. We present simple arithmetic operations in the latent space to demonstrate that such operations can decompose the latent representation into subspaces modeling different physical generating factors, such as speaker identity and linguistic content. By manipulating the latent representation, we also demonstrate an ability to perturb some aspect of the surface speech segment, for example the speaker identity,

while keeping the remaining attributes fixed (e.g., linguistic content). To quantify the behavior of the latent representation modifications, an experiment is conducted to measure our ability to modify speaker characteristics without changing linguistic content, and vice versa. In addition, we perform an analysis to evaluate the model's ability to generate speech segments of different durations. Part of the content in this chapter was published in Hsu et al. (2017a).

2.1 Variational Autoencoders for Speech

2.1.1 Introduction to Variational Autoencoders

Variational autoencoders (Kingma and Welling, 2013) define a probabilistic generative process between observation $\boldsymbol{x}$ and latent variable $\boldsymbol{z}$ as follows: (1) a latent variable $\boldsymbol{z}$ is drawn from the prior distribution $p(\boldsymbol{z})$. (2) an observed variable $\boldsymbol{x}$ is then drawn from a conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$. The prior $p(\boldsymbol{z})$ and the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ are assumed to be in some parametric probability distribution family, whose parameters are collectively denoted by $\boldsymbol{\theta}$. In an unsupervised setting, we are only given a dataset $\boldsymbol{x} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$ of N i.i.d. samples. The true value of $\boldsymbol{\theta}$, as well as the latent variable $\boldsymbol{z}$ associated with each observation $\boldsymbol{x}$ in this process are unknown.

We are often interested in knowing the marginal likelihood of the data $p(\boldsymbol{x})$, or the posterior $p(\boldsymbol{z}|\boldsymbol{x})$; however, both require computing the intractable integral $\int p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})d\boldsymbol{z}$. To address this problem, the VAE framework introduces an amortized inference model $q(\boldsymbol{z}|\boldsymbol{x})$, which is in some parametric probabilistic distribution family and approximates the true posterior $p(\boldsymbol{z}|\boldsymbol{x})$. The parameters of $q(\boldsymbol{z}|\boldsymbol{x})$ are denoted by $\boldsymbol{\phi}$. We can therefore rewrite the marginal likelihood as:

$$\begin{aligned}
\log p(\boldsymbol{x}) &= D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) \\
&\geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) \\
&= -D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) + \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})],
\end{aligned} \tag{2.1}$$

where $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ is the variational lower bound of the marginal likelihood that we want to optimize with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

In the VAE framework we consider here, both the recognition model $q(\boldsymbol{z}|\boldsymbol{x})$ and the generative model $p(\boldsymbol{x}|\boldsymbol{z})$ are parameterized using diagonal Gaussian distributions, of which the mean and the covariance are computed with a neural network. The prior is assumed to be a centered isotropic multivariate Gaussian $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \boldsymbol{I})$, that has no free parameters.

In practice, the expectation in the second term of Eq. 2.1 is approximated with the Monte Carlo estimation, by first drawing L samples from $\boldsymbol{z}^l \sim q(\boldsymbol{z}|\boldsymbol{x})$, and then computing $\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] \simeq \frac{1}{L}\sum_{l=1}^{L} \log p(\boldsymbol{x}|\boldsymbol{z}^l)$. To yield a differentiable network after sampling, the reparameterization trick (Kingma and Welling, 2013) is used. Suppose $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu_z}, \boldsymbol{\sigma_z^2 I})$, after reparameterizing we have $\boldsymbol{z} = \boldsymbol{\mu_z} + \boldsymbol{\sigma_z} \odot \boldsymbol{\epsilon}$, where $\odot$ denotes an element-wise product, and vector $\boldsymbol{\epsilon}$ is sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and treated as an additional input.

2.1.2 A Convolutional Model Architecture

Our goal is to learn latent representations of speech segments to model the generation process. We let the observed data $\boldsymbol{x}$ be a sequence of frames of fixed length. The learned latent variable $\boldsymbol{z}$ is therefore supposed to encode the factors that result in the variability of speech segments, such as the content being spoken, speaker identity, and channel effect.

As mentioned in the previous section, a VAE is composed of an inference model and a generative model. The inference model takes a speech segment as input and predicts the mean $\boldsymbol{\mu_z}$ and the log-variance $\log \boldsymbol{\sigma_z^2}$ that parameterize the posterior distribution $q(\boldsymbol{z}|\boldsymbol{x})$. A speech segment is treated as a T-by-F tensor, similar to an image of height T and width F; however, unlike images, speech segments are only translational invariant in the time axis. Therefore, similar to Harwath and Glass (2017), 1-by-F filters are applied at the first convolutional layer, and w-by-1 filters at following layers. As suggested in Radford et al. (2015), instead of pooling, we use a stride size > 1 for down-sampling along the time axis. The output from the

last convolutional layer is flattened and fed into fully connected layers before going to the Gaussian parameter layer modeling the latent variable $\boldsymbol{z}$. See Table 2.1 for a summary, and Figure 2-1 for graphical illustration.

Table 2.1: Inference model architecture. *Conv* refers to convolutional layers, *Reshape* convert a 3-D tensor of shape $(1, T/4, 256)$ to a vector, *Fc* refers to fully connected layers, and *Gauss* refers to the Gaussian parametric layer predicting mean and log variance of $q(\boldsymbol{z}|\boldsymbol{x})$

	Conv1	Conv2	Conv3	Reshape	Fc1	Gauss
#filters/units	64	128	256	-	512	128
filter size	1xF	3x1	3x1	-	-	-
stride	(1,1)	(2,1)	(2,1)	-	-	-

The generative network takes sampled $\boldsymbol{z}$ as input, and predicts the mean $\boldsymbol{\mu_x}$ as well as the log-variance $\log \boldsymbol{\sigma_x^2}$ of the observed data. Here we use symmetric architectures to the corresponding recognition network. See Table 2.2 for a summary. Figure 2-1 illustrates the encoder-decoder architecture.

Table 2.2: Generative model architecture. *Fc* refers to fully connected layers, *Reshape* converts a vector to a 3-D tensor of shape $(1, T/4, 256)$, *T-Conv* refers to transposed convolutional layers, and *Gauss* refers to the Gaussian parametric layer predicting mean and log variance of $p(\boldsymbol{x}|\boldsymbol{z})$

	Fc1	Fc2	Reshape	T-Conv1	T-Conv2	Gauss
#filters/units	512	256 * (T/4)	-	128	64	1
filter size	-	-	-	3x1	3x1	1xF
stride	-	-	-	(2,1)	(2,1)	(1,1)

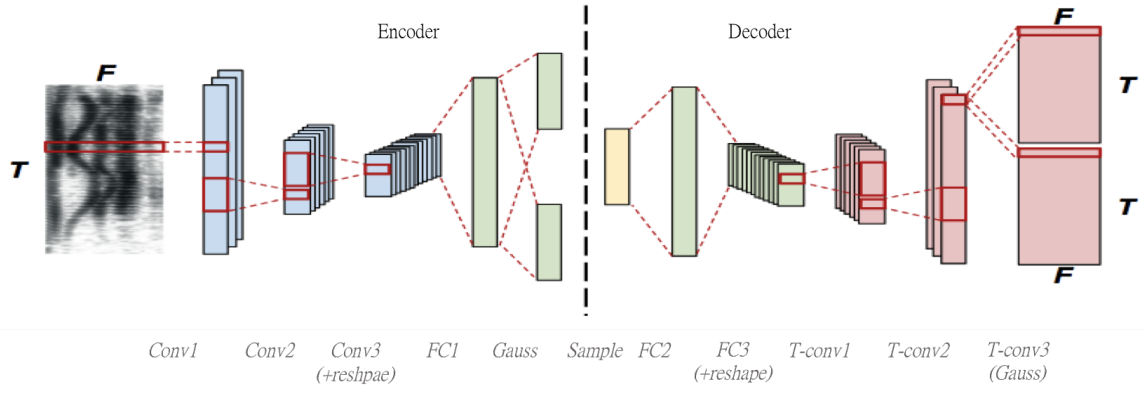


Figure 2-1: Illustration of the convolutional VAE architecture.

Different choices for the activation function were investigated. No activation is applied to Gaussian parameter layers, since the mean and the log-variance are unbounded for both $\boldsymbol{x}$ and $\boldsymbol{z}$. For other layers, we use tanh, which leads to a higher variational lower bound compared to using rectified linear units. Batch normalization is applied to every layer except for the Gaussian parameter layer.

2.2 Interpreting Learned Representations

In this section, we discuss methods for interpreting learned latent representations. Specifically, we propose the idea of latent attribute representations that capture the signature representation of a specific physical attribute, and derive simple arithmetic operations for modifying attributes of speech segments. We will use a to denote the *attribute*, such as a phoneme, and r to denote the *value of some attribute*, such as /ae/.

2.2.1 Latent Attribute Representations and Derivation

The first assumption we make is that conditioning on some attribute a being r, such as the *phone* being */ae/*, the prior distribution of $\boldsymbol{z}$ is also a Gaussian; in other words, $p(\boldsymbol{z}; r) = N(\boldsymbol{z}; \boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$. We therefore define $\boldsymbol{\mu}_r$ as the *latent attribute representation* for r. Let $\boldsymbol{X}_r = \{\boldsymbol{x}_r^{(i)}\}_{i=1}^{N_r}$ be a subset of $\boldsymbol{x}$ where the attribute a of each instance is r. We can then estimate $\boldsymbol{\mu}_r$ as follows:

$$\begin{aligned}
\boldsymbol{\mu}_r = \mathbb{E}_{p(\boldsymbol{z};r)}[\boldsymbol{z}] &= \int_{\boldsymbol{z}} \boldsymbol{z} p(\boldsymbol{z}; r) d\boldsymbol{z} \\
&= \int_{\boldsymbol{z}} \int_{\boldsymbol{x}} \boldsymbol{z} p(\boldsymbol{z}|\boldsymbol{x}; r) p(\boldsymbol{x}; r) d\boldsymbol{x} d\boldsymbol{z} \\
&\approx \int_{\boldsymbol{x}} p(\boldsymbol{x}; r) \int_{\boldsymbol{z}} \boldsymbol{z} q(\boldsymbol{z}|\boldsymbol{x}; r) d\boldsymbol{z} d\boldsymbol{x} \\
&\approx \frac{1}{N_r} \sum_{i=1}^{N_r} \int_z \boldsymbol{z} q(\boldsymbol{z}|\boldsymbol{x}_r^{(i)}) \\
&= \frac{1}{N_r} \sum_{i=1}^{N_r} \tilde{\boldsymbol{\mu}}_r^{(i)},
\end{aligned} \tag{2.2}$$

where $\tilde{\boldsymbol{\mu}}_r^{(i)}$ is the variational posterior mean of $\boldsymbol{x}_r^{(i)}$, predicted by the inference model. This results in averaging the posterior mean of each instance in $\boldsymbol{X}_r$.

2.2.2 Arithmetic Operations to Modify Speech Attributes

Here we make the second assumption: let there be K independent attributes that affect the realization of speech, each attribute a_k is then modeled using a subspace Z_{a_k}, where $Z = \cup_{k=1}^{K} Z_{a_k}$ and $Z_{a_k} \perp Z_{a_{k'}}$ if $k \neq k'$. Hence, the latent representation can be decomposed into K orthogonal latent attribute representations $\boldsymbol{z}_{a_1}, \boldsymbol{z}_{a_2}, \cdots, \boldsymbol{z}_{a_k}$, where $\boldsymbol{z}_{a_k} \in Z_{a_k}$ and $\boldsymbol{z} = \sum_{k=1}^{K} \boldsymbol{z}_{a_k}$. Combining the aforementioned assumption of the conditioned prior of $\boldsymbol{z}$, we can next derive the latent space arithmetic operations to modify the speech attributes.

Suppose we want to modify the attribute a_k, for example the speaker identity, of a speech segment $\boldsymbol{x}^{(i)}$, from being speaker r_s to being speaker r_t. Given the latent attribute representations $\boldsymbol{\mu}_{r_s}$ and $\boldsymbol{\mu}_{r_t}$ for speaker r_s and r_t respectively, the *latent speaker shift* $\boldsymbol{v}_{r_s \to r_t}$ is computed as: $\boldsymbol{v}_{r_s \to r_t} = \boldsymbol{\mu}_{r_t} - \boldsymbol{\mu}_{r_s}$. We can then modify a speech segment $\boldsymbol{x}^{(i)}$ of speaker r_s as follows:

$$\boldsymbol{z}^{(i)} \sim q(\boldsymbol{z}|\boldsymbol{x}^{(i)}) \tag{2.3}$$

$$\boldsymbol{z}_{mod}^{(i)} = \boldsymbol{z}^{(i)} + \boldsymbol{v}_{r_s \to r_t} \tag{2.4}$$

$$\boldsymbol{x}_{mod}^{(i)} \sim p(\boldsymbol{x}|\boldsymbol{z}_{mod}^{(i)}), \tag{2.5}$$

which does not modify latent attribute representations other than $\boldsymbol{z}_{a_k}^{(i)}$, because $\boldsymbol{v}_{r_s \to r_t} \perp \boldsymbol{z}_{a_{k'}}^{(i)}$ for $k' \neq k$.

2.3 Data, Experiments, and Discussion

2.3.1 Data and Preprocessing

The TIMIT acoustic-phonetic corpus (Garofolo et al., 1993; Zue et al., 1990) contains broadband recordings of phonetically-balanced read speech. A total of 6300

utterances (5.4 hours) are presented with 10 sentences from each of 630 speakers, of which approximately 70% are male and 30% are female. Each utterance comes with manually time-aligned phonetic and word transcriptions, as well as a 16-bit, 16kHz speech waveform file. We follow Kaldi's TIMIT recipe to split train/dev/test sets and exclude dialect sentences (SA), with 462/50/24 non-overlapping speakers in each set respectively. Phonetic transcriptions are based on 58 phones, excluding silence phones.

We consider two types of frame representations: magnitude spectrum in dB (Spec) and filter banks (FBank). For both features, we first apply a 25ms Hanning window with 10ms shift, and then compute the short time Fourier transform coefficients with flooring at -20dB. For FBank features, 80 Mel-scale filter banks that match human perceptual sensitivity are applied, which preserves more detail at lower frequency regions. We investigate two different segment lengths: 200ms and 1s, which correspond to 20 frames and 100 frames, and are referred to as *syllable-level* and *word-level* datasets, respectively.

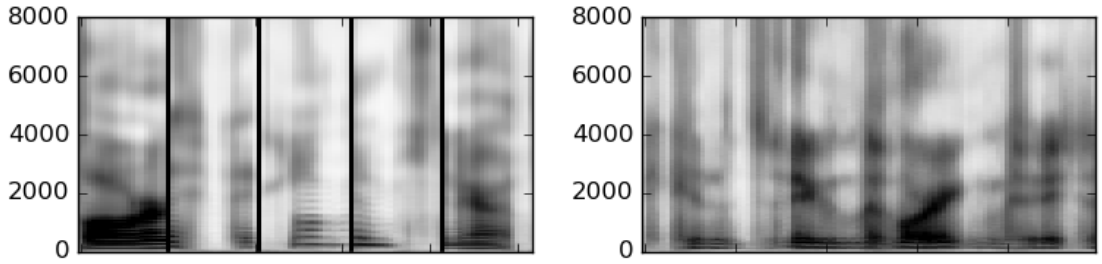
2.3.2 Experiment Setups

All models were trained with stochastic gradient descent using a mini-batch size of 128 without clipping to minimize the negative variational lower bound plus an $L2$-regularization with weight 10^{-4}. The Adam (Kingma and Ba, 2014) optimizer is used with $\beta_1 = 0.95$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and initial learning rate of 10^{-3}. Training is terminated if the lower bound on the development set does not improve for 10 epochs. To compare with VAE, we also train an autoencoder (AE) with the same proposed model architecture except for the Gaussian latent variable layer, which is replaced with a fully-connected layer of 128 hidden units.

2.3.3 Sampling from the Learned Generative Process

One of the advantages of VAEs is that the prior $p_\theta(z)$ is assumed to be a centered isotropic Gaussian, which enables us to sample latent vectors and reconstruct speech-

like segments. Here, we investigate the syllable-level and word-level datasets.



(a) Syllable-level (b) Word-level

Figure 2-2: Random samples drawn from models trained with syllable-level and word-level dataset. The segments in (a) are 200ms, and the segment in (b) is 1s.

Figure 2-2 (a) shows five random samples from the syllable-level model, which look and sound reasonable; however, we observe that random samples drawn from the word-level model are less natural because of excessive closures (vertical stripes), as shown in Figure 2-2 (b). The failure of drawing random samples implies that there is discrepancy between the assumed prior and the true prior. We hypothesize that because per-dimension KL-divergence values are computed, and correlations among dimensions are not penalized, the covariance matrix of the true prior may not be diagonal. We estimate the covariance matrix of the true prior by sampling the latent representations of the entire test set and compute the full covariance matrix. Figure 2-3 compares the syllable model and the word model on the sum of off-diagonal covariance scale for each dimension. We can observe that the word-level model has higher correlations between different dimensions than the syllable-level model.

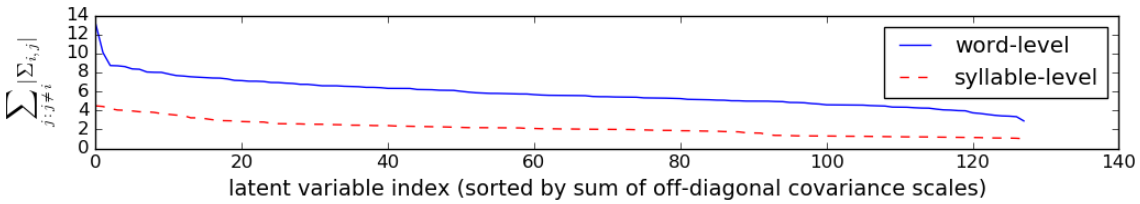
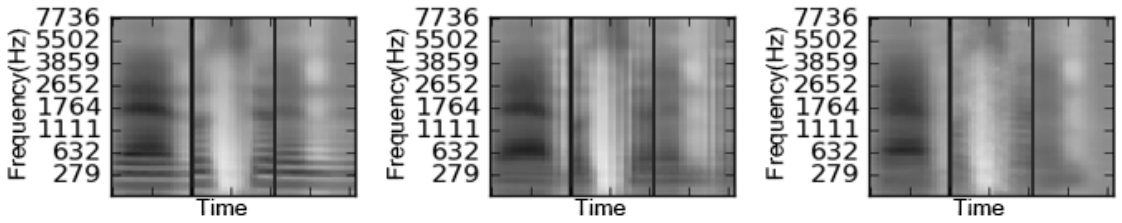


Figure 2-3: Comparison of sum of off-diagonal covariance scales for each dimension for the syllable and word-level dataset.

2.3.4 Decoding Latent Phoneme Representation

In Figure 2-4, we show the results of reconstructing from latent attribute representations of three phones, /ae/, /th/, and /n/, using VAE and AE respectively. As a baseline, we also show the results of averaging filter bank features. The VAE preserves more harmonic structure and clearer spectral envelope, while the AE and the FBank are more blurred. It is worth noting that AE also shows unnatural frequent vertical stripe artifacts.



(a) VAE (b) AE (c) Fbank

Figure 2-4: Comparison between VAE, AE and Fbank on averaging representations of /ae/, /th/, and /n/ from left to right. Each segment is 200ms long.

To assess the orthogonality-between-attributes assumption, we sampled six speakers, three males and three females, denoted by *{m,f}_spk[i]*, and ten phones, including vowels, stops, fricatives, and nasals, to compute three latent speaker representations and ten latent phone representations. Figure 2-5 plots the cosine similarities between these representations. From the figure, we can observe that off-diagonal blocks have low cosine similarities, which indicates that latent speaker representations and latent phone representations reside in orthogonal latent subspaces. Second, different latent phone representations also cluster according to the phonetic characteristics, which suggests the latent phone subspace may be further divided.

2.3.5 Speech Transformation and Voice Conversion

We next explored modifying the phone and speaker attributes using the derived operations in Section 2.2.2. Figure 2-6 shows an example of drawing 10 instances of the phone /aa/ (top) and transforming them to /ae/ (bottom) using the latent attribute

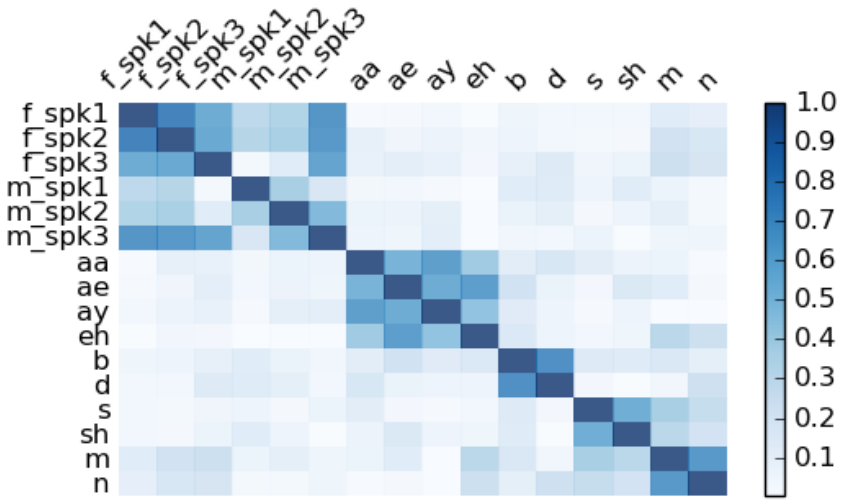


Figure 2-5: Cosine similarities of latent attribute representations.

shift $\boldsymbol{v}_{aa \to ae}$. We can observe that the second formant F_2, marked with red boxes,[1] of each instance goes up after modification, because it is being changed from a back vowel to a front vowel. On the other hand, the harmonics of each instance, which are closely related to the speaker identity, maintain roughly the same.

Figure 2-7 illustrates modifying 10 instances from a female speaker *falk0* to a male speaker *madc0* with the latent attribute shift $\boldsymbol{v}_{falk0 \to madc0}$. The harmonics (horizontal stripes) decrease after modification, while the spectrum envelope remains the same, indicating that the phonetic content is not changed.

In an attempt to quantify our latent attribute perturbation, we trained convolutional phonetic and speaker classifiers so that we could measure the difference of the posterior of each attribute before and after modification. The 58-class phone classifier achieves a test accuracy of 72.2%, while the 462-class speaker classifier achieves a test accuracy of 44.2%.

The shifts in posterior distributions of the phone and speaker classifications on the modified data are shown in Table 2.3. The upper half of the table contains results for speech segments that were transformed from /aa/ to /ae/. The first row shows that the average /aa/ posterior was 34% while the average correct speaker posterior was 51%. The second row shows that after modification to an /ae/, the average phone

[1]Best viewed in color.

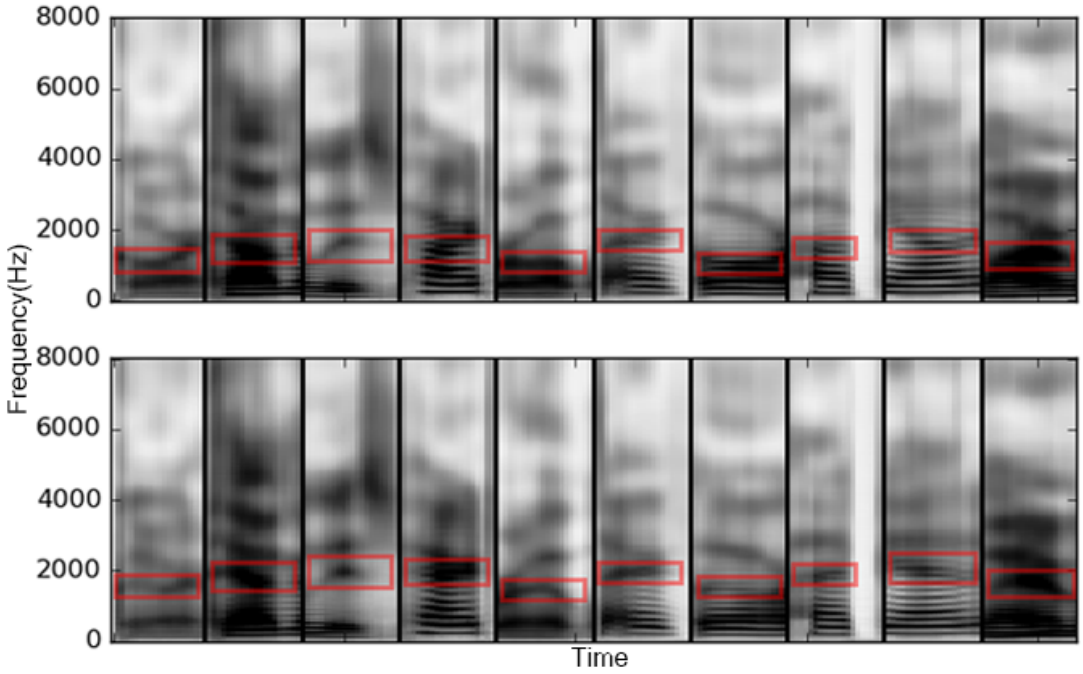


Figure 2-6: Modify the phone from /aa/ (top) to /ae/ (bottom). Each segment is 200ms long.

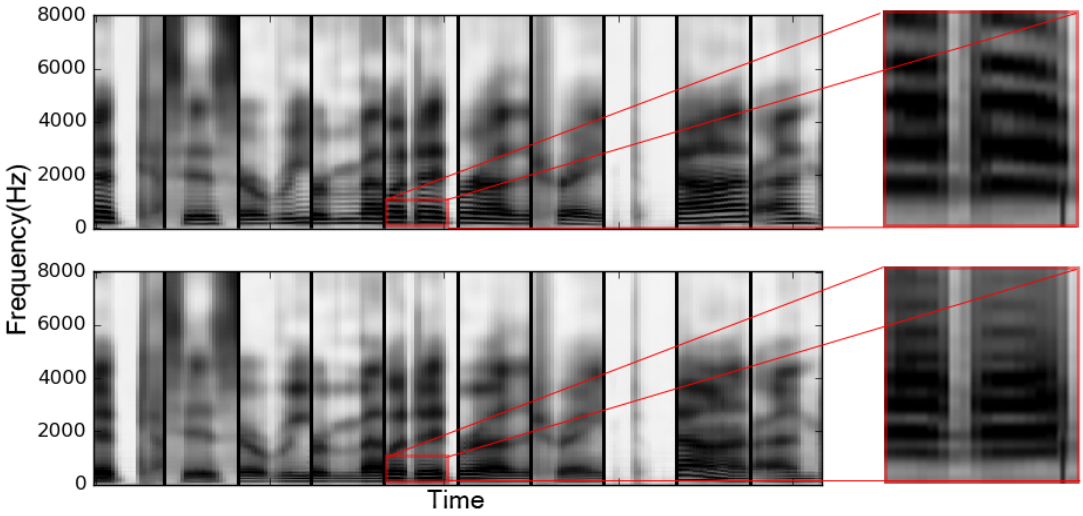


Figure 2-7: Modify from a female (top) to a male (bottom). Each segment is 200ms long. We can observe that the vertical spacing between horizontal stripes are denser in the examples on the bottom, indicating that the pitch is lower in those examples. However the conversion is not perfect, because harmonics are not uniformly spread.

posteriors shift dramatically to be 30% /ae/, while slightly degrading the average correct speaker posterior.

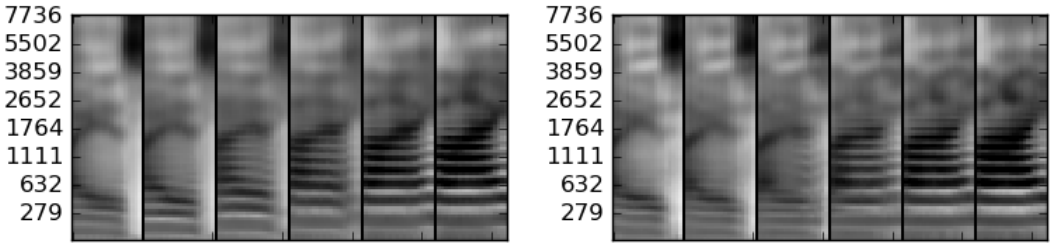
The lower part of the table shows the results of speech segments that had speaker identity modified from speaker 'falk0' to 'madc0'. The third row shows an average speaker posterior of 44% for 'falk0' in the unmodified samples, while the average correct phone posterior was 55%. After modification we see that the average speaker posterior has shifted to be 29% 'madc0' while slightly degrading the average correct phone posterior.

		/aa/	/ae/	ori. spk.
Modify Phone	before	34.06%	0.45%	50.78%
	after	0.24%	29.73%	41.66%
		falk0	madc0	ori. phone
Modify Speaker	before	44.48%	0.02%	54.61%
	after	3.11%	28.71%	48.71%

Table 2.3: Average posteriors over 10 instances of source, target, and fixed attributes before and after modification.

2.3.6 Speech Interpolation

Finally, we explore the operation of interpolation in the latent space between speech segments. Since $p(\boldsymbol{z})$ is log-concave, the interpolated $\boldsymbol{z}_{int} = \alpha \boldsymbol{z}_a + (1-\alpha)\boldsymbol{z}_b$, where $\alpha \in [0,1]$, would have $p(\boldsymbol{z}_{int}) \geq \min(p(\boldsymbol{z}_a), p(\boldsymbol{z}_b))$. Therefore it should also generate reasonable speech-like segments. Figure 2-8 shows the transition between a male /ey/ to a female /ay/ using VAE and AE respectively. For VAE, we can clearly observe the pitch shifting and the formant contour transforming; however for AE it is more akin to interpolation in the raw feature space, where the magnitude of one segment goes down as the other goes up.



(a) VAE (b) AE

Figure 2-8: Interpolation in the latent space using VAE and AE. Each segment is 200ms long.

Chapter 3

Unsupervised Adaptation via VAE-Based Data Augmentation

Domain mismatch between training and testing can lead to significant degradation in performance in many machine learning scenarios. Unfortunately, this is not a rare situation for automatic speech recognition deployments in real-world applications. Research on robust speech recognition can be regarded as trying to overcome this domain mismatch issue.

Some robust ASR research focuses on enhancing speech, by applying beam-forming techniques (Anguera et al., 2007; Erdogan et al., 2016), estimating noise masks (Narayanan and Wang, 2013; Isik et al., 2016), or training denoising models (Maas et al., 2012; Feng et al., 2014), etc. Other research extracts robust acoustic features (Kingsbury et al., 1998; Stern and Morgan, 2012; Vinyals and Ravuri, 2011; Sainath et al., 2012; Fredes et al., 2017) that are intended to be invariant for ASR even in adverse environments. Another line of research investigates modeling techniques, including, but not limited to, model adaptation (Gales, 1998; Yu et al., 2013), multi-condition training (Seltzer et al., 2013), and adversarial training (Sun et al., 2017; Meng et al., 2017). Over the past decade, neural network-based acoustic models have come to dominate the ASR field. To utilize the full capacity of neural network-based acoustic models, it is often a good strategy to train a model with as much, and as diverse, a dataset as possible (Li et al., 2012).

In this chapter, we consider a highly adverse scenario, where both source and target domain speech are available, but word transcripts are only available for the source domain data. We present novel augmentation-based methods that transform speech, but, do not require altering existing transcripts. Specifically, we first train an unsupervised sequence-to-sequence recurrent variational autoencoder (VAE) on both source and target domain data to learn a latent representation of speech. We then transform "nuisance" attributes of speech, such as speaker identities and noise types, that do not contain linguistic information, and are thus irrelevant to ASR, by modifying the latent representation with the operations proposed in the previous chapter, in order to create additional labeled training data whose distribution is more similar to the target domain. The proposed method is evaluated on the CHiME-4 and Aurora-4 dataset, reducing the absolute word error rate (WER) by as much as 35% and 40% compared to the non-adapted baseline. This chapter was published in Hsu et al. (2017c).

3.1 A Sequence-to-Sequence Recurrent VAE

Similar to the previous chapter, we use Mel-scale filter bank coefficients (FBank) of 80 dimensions as the frame-level representation of speech, which are extracted every 10ms using a 25ms Hamming window. Observed data $\boldsymbol{x} = \{x_1, \cdots, x_{20}\}$ are speech segments of 20 frames, roughly at the scale of a syllable. VAEs are applied to learn the generative process of syllable-level speech segments. Likewise, both the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ and the approximate posterior distribution $q(\boldsymbol{z}|\boldsymbol{x})$ are assumed to be diagonal Gaussian distributions, whose mean and log variance are parameterized by neural networks with $\boldsymbol{z}$ and $\boldsymbol{x}$ as inputs respectively. The prior $p(\boldsymbol{z})$ is considered a centered isotropic multivariate Gaussian of 64 dimensions.

To better model the temporal relationship within speech segments, we apply a sequence-to-sequence long short-term memory (Seq2Seq-LSTM) architecture as illustrated in Figure 3-1. The encoder is a two layer LSTM with 512 hidden units, which inputs the speech segment frame by frame. The outputs from both layers are then

concatenated and fed into a fully connected Gaussian parameter layer that predicts the mean and the log variance of the latent variable $\boldsymbol{z}$. The reparameterization trick is applied for differentiable sampling.

The decoder is also a two layer LSTM with 512 hidden units that takes the sampled latent variable as the input and generates a sequence of outputs. Each output is used as the input to a fully-connected Gaussian parameter layer shared across different time steps that predicts the mean and the log variance for one frame of $\boldsymbol{x}$. The entire model can be seen as a stochastic sequence-to-sequence autoencoder that encodes a frame sequence stochastically to the latent space, and then decodes statistically from a sampled latent variable to a sequence of frames.

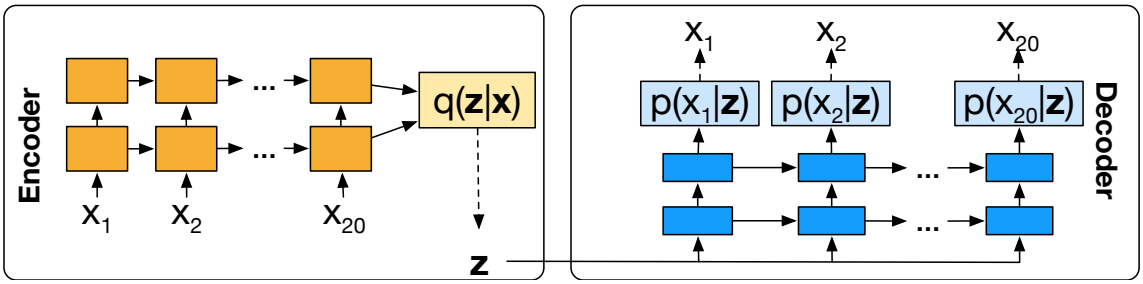


Figure 3-1: Illustration of the Sequence-to-Sequence LSTM VAE architecture.

3.2 VAE-Based Data Augmentation

Here we extend the idea of latent attribute representations in Section 2.2.1, in order to discover the latent subspace that models non-linguistic generating factors. We then propose two augmentation methods that transform a speech utterance without altering its associated transcripts by perturbing the latent space.

3.2.1 Latent Nuisance Representations

We define *nuisance factors* to be those that affect the surface form of a speech utterance but not the linguistic content, such as speaker identity, channel, and background noise, etc. These factors, unlike phonetic attributes, are generally consistent among segments within an utterance, which implies that we can assume the labels for these

factors are the same for all the segments within an utterance. Therefore, we can estimate one *latent nuisance representation* for each utterance by averaging the posterior mean of latent variables for each segment in that utterance, according to Eq. 2.2. We denote the latent nuisance representation of the j-th utterance with $\boldsymbol{\mu}_{utt_j}$.

3.2.2 Data Augmentation Pipeline Overview

By transforming nuisance factors of the labeled source data, we can generate labeled training data (i.e. with transcripts) for automatic speech recognition systems. The newly generated data can still use the original transcript for training but differ in some aspect, such as speaker quality and background noise, from the original speech. Figure 3-2 shows the flowchart of generating transformed labeled data. Let $(\{\boldsymbol{x}_{utt_j}^{(i)}\}_{i=1}^{N_j}, tra_{utt_j})$ be the source utterance of N_j segments with the transcript tra_{utt_j} that we want to modify. We first encode and sample each segment $\boldsymbol{x}_{utt_j}^{(i)}$ to generate $\boldsymbol{z}_{utt_j}^{(i)}$ using a trained VAE encoder. Then the same transformation operation is applied to each latent variable in $\{\boldsymbol{z}_{utt_j}^{(i)}\}_{i=1}^{N_j}$ to produce $\{\tilde{\boldsymbol{z}}_{utt_j}^{(i)}\}_{i=1}^{N_j}$. Finally, we decode $\{\tilde{\boldsymbol{z}}_{utt_j}^{(i)}\}_{i=1}^{N_j}$ using the same trained VAE decoder to obtain the modified utterance $\{\tilde{\boldsymbol{x}}_{utt_j}^{(i)}\}_{i=1}^{N_j}$ that shares the same transcript tra_{utt_j} with the original utterance. In other words, we create new labeled training data: $(\{\tilde{\boldsymbol{x}}_{utt_j}^{(i)}\}_{i=1}^{N_j}, tra_{utt_j})$. We next introduce two types of modification operations in Section 3.2.3 and 3.2.4 respectively.

3.2.3 Type I: Nuisance Factor Replacement

The first type of modification operation we consider is to replace the nuisance factors of one utterance with those of another utterance. We assume VAEs use orthogonal subspaces to model linguistic and nuisance factors, and apply the operation derived in Section 2.2.2. Let $\{\boldsymbol{z}_{utt_{src}}^{(i)}\}_{i=1}^{N_j}$ be the encoded latent variables of the source utterance segments we want to modify from, $\boldsymbol{\mu}_{utt_{src}}$ be the latent nuisance representation from the source utterance, and $\boldsymbol{\mu}_{utt_{tar}}$ be the latent nuisance representation from the target

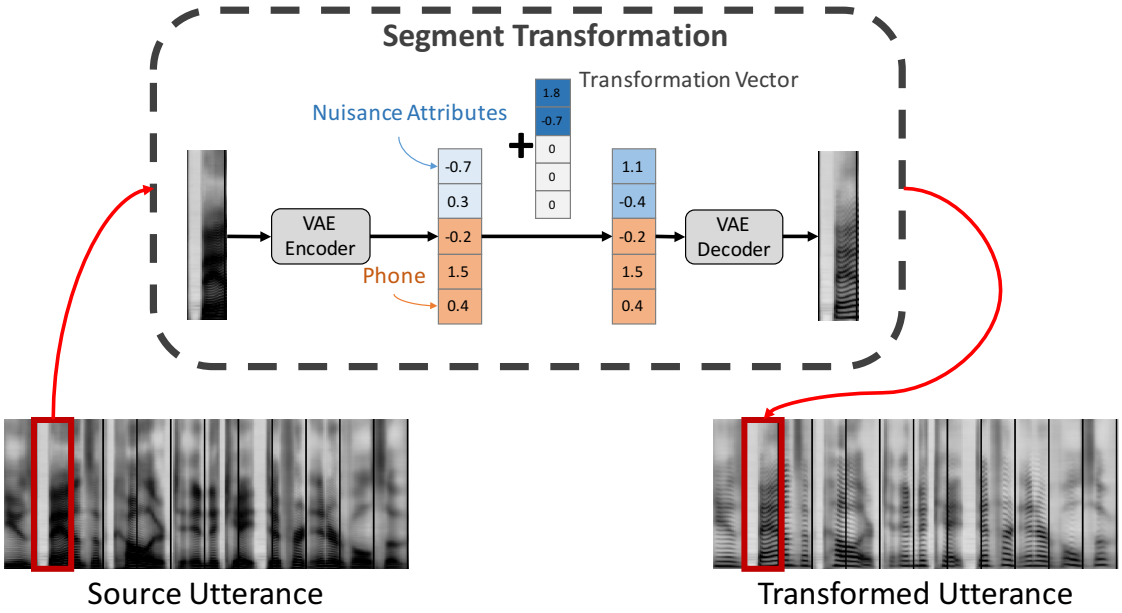


Figure 3-2: Flowchart of generating transformed labeled data.

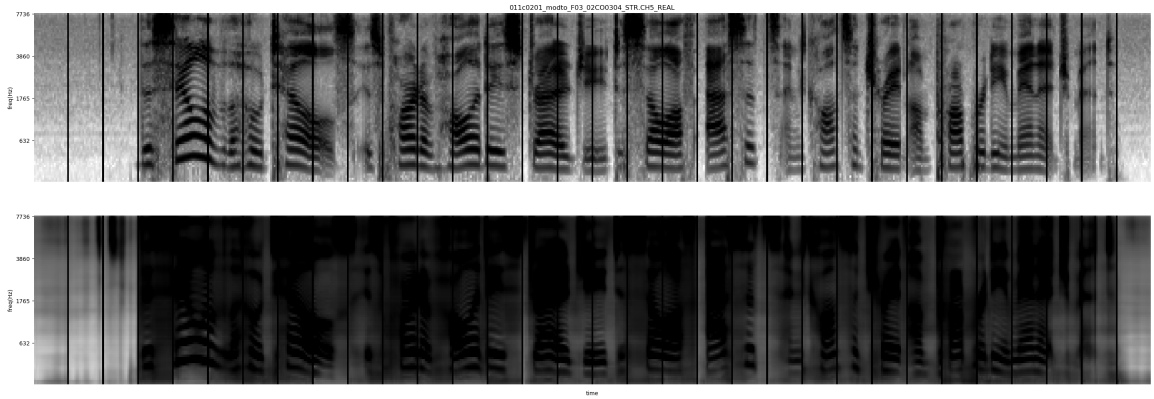
utterance. Then we modify $\boldsymbol{z}^{(i)}_{utt_{src}}$ as follows:

$$\tilde{\boldsymbol{z}}^{(i)}_{utt_{src}} = \boldsymbol{z}^{(i)}_{utt_{src}} - \boldsymbol{\mu}_{utt_{src}} + \boldsymbol{\mu}_{utt_{tar}}.$$

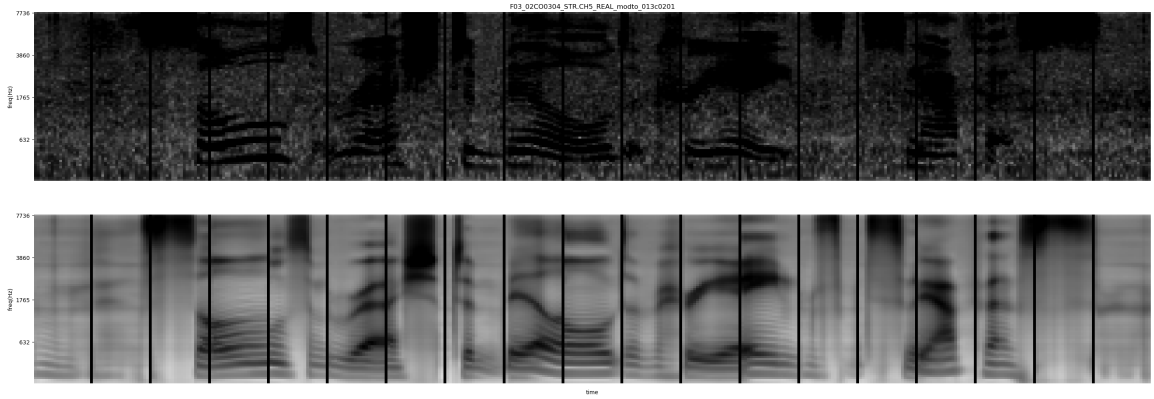
Figure 3-3 shows two examples of modifying the nuisance factors, where the first row is the original utterance and the second row is the modified utterance. In Figure 3-3(a), a clean utterance is modified by replacing its nuisance factors with those from a noisy utterance. Conversely, Figure 3-3(b) illustrates an example of modifying a noisy utterance by replacing its nuisance factors with those estimated from a clean utterance. In the figure, segments within an utterance are separated by vertical black lines. From both examples we can observe that while the spacing between harmonics and the level of noise changes, the linguistic content does not seem to change after replacing the nuisance attributes.

3.2.4 Type II: Latent Nuisance Subspace Perturbation

The fundamental assumption for such a transformation operation is that VAEs learn to use orthogonal subspaces to model linguistic factors and nuisance factors respec-



(a) modifying a clean utterance to be like a noisy utterance



(b) modifying a noisy utterance to be like a clean utterance

Figure 3-3: Two examples of replacing the nuisance attributes.

tively. Hence, we are able to modify the nuisance factors without changing the original linguistic attribute by only modifying factors in the latent nuisance subspace, and keeping factors in the latent linguistic subspace intact. While the operation in Section 3.2.3 bypasses the search for the latent nuisance subspace, we can alternatively discover this subspace, and then sample or perturb the factors in it to change the nuisance factors of an utterance.

Determining the latent nuisance subspace with PCA

Given a dataset of M utterances, we can compute M latent nuisance representations $\{\boldsymbol{\mu}_{utt_j}\}_{j=1}^{M}$, with one for each utterance. The latent nuisance subspace is composed of a set of bases, which captures the variations among these latent nuisance representations. We apply principle component analysis (PCA) on the M latent nuisance

representations to obtain a list of eigenvectors $\{\boldsymbol{e}_d\}_{d=1}^{D}$, sorted in descending order by their associated eigenvalues $\{\sigma_d^2\}_{d=1}^{D}$, where D is the dimension of the latent variable $\boldsymbol{z}$. Each eigenvalue interprets the variance of latent nuisance representations along the direction of its associated eigenvector. We plot the eigenvalues in Figure 3-4 in descending order, where we can observe that most of the variation is captured by the first few dimensions.

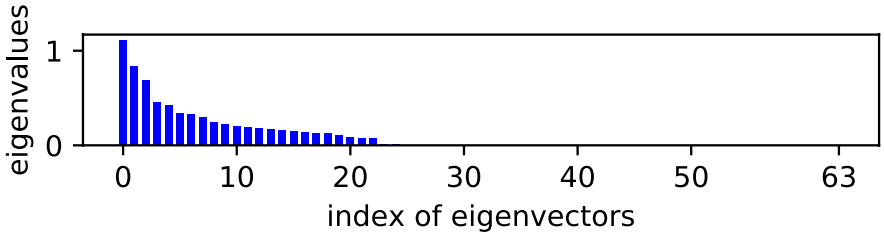


Figure 3-4: Eigenvalues of PCA analysis on latent nuisance representations in descending order.

Soft latent nuisance subspace perturbation

An intuitive way to determine and perturb the latent nuisance subspace is to select the first few eigenvectors and only perturb in those directions. We refer to this as *hard latent nuisance subspace perturbation*, since it demands a hard decision on the rank of the subspace. Alternatively, we propose an approach called *soft latent nuisance subspace perturbation*, which generates a *perturbation vector* $\boldsymbol{p}$ as follows:

$$\boldsymbol{p} = \gamma \sum_{d=1}^{D} \psi_d \sigma_d \boldsymbol{e}_d, \quad \psi_d \sim \mathcal{N}(0,1),$$

where ψ_d is drawn from a normal distribution, σ_d and $\boldsymbol{e}_d$ are the square root of the d-th largest eigenvalue and its associated eigenvector, and γ is a hyper-parameter, referred to as the *perturbation ratio*. It can be observed that the expected scale we perturb along an eigenvector $\boldsymbol{e}_d$ is proportional to the standard deviation of latent nuisance representations along that eigenvector, which is the square root of its eigenvalue σ_d^2. This approach thus automatically adapts to different distributions of eigenvalues,

regardless how many dimensions a VAE learns to use to model the nuisance attributes.

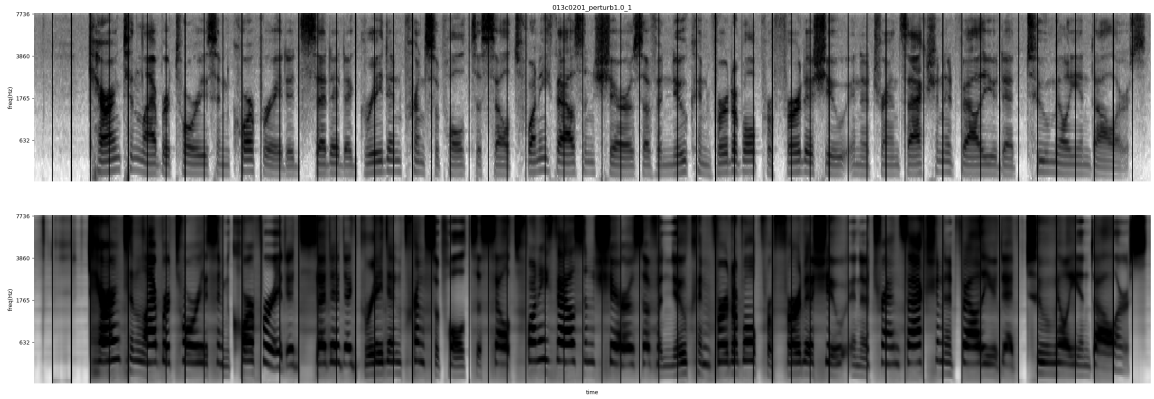


Figure 3-5: An example of perturbing latent nuisance attributes. In this example, the transformed utterance contains more background noise and is of a higher pitch.

Let $\{\boldsymbol{z}_{utt_{src}}^{(i)}\}_{i=1}^{N_j}$ be the encoded latent variables of the source utterance segments we want to perturb. We modify each latent variable as follows:

$$\tilde{\boldsymbol{z}}_{utt_{src}}^{(i)} = \boldsymbol{z}_{utt_{src}}^{(i)} + \boldsymbol{p},$$

which adds the same perturbation vector $\boldsymbol{p}$ to each segment in an utterance such that the nuisance factor change is consistent for all segments within an utterance. Figure 3-5 shows an example of perturbing the latent nuisance attributes with $\gamma = 1.0$, where the first row is the original utterance and the second row is the perturbed utterance.

3.3 Setup

3.3.1 Dataset

Our dataset is based on the CHiME-4 challenge (Vincent et al., 2016), which targets distant-talking ASR and whose setup is based on the speaker-independent medium (5K) vocabulary subset of the Wall Street Journal (WSJ0) corpus (Garofalo et al., 2007). The training set of the CHiME-4 dataset consists of 1,600 utterances recorded in four noisy environments from four speakers, and 7,138 simulated noisy utterances based on the clean utterances in the WSJ0 SI-84 training set. We use the original

7,138 clean utterances as the labeled source-domain data, and the 1,600 single channel real noisy utterances as the unlabeled target-domain data for unsupervised domain adaptation. Performance is evaluated on both the real noisy utterances and the original clean utterances in the development partition of the CHiME-4 dataset in terms of the word error rate (WER).

In addition, we also repeat our experiments on Aurora-4 (Pearce, 2002) to compare with the results reported in Sun et al. (2017). Aurora-4 is a broadband corpus designed for noisy speech recognition tasks based on WSJ0 as well. Two microphone types, clean/channel are included, and six noise types are artificially added to both microphone types, which results in four conditions: clean(A), channel(B), noisy(C), and channel+noisy(D). We use the clean training set as the labeled source-domain data, and the multi-condition development set as the unlabeled target-domain data. The multi-condition test_eval92 set is used for evaluation.

3.3.2 VAE Setup and Training

All the original clean utterances and the real noisy utterances are mixed and split into training and development sets with the ratio of 90-10 for training the Seq2Seq LSTM VAE. The VAE is trained with stochastic gradient descent using a mini-batch size of 128 without clipping to minimize the negative variational lower bound plus an $L2$-regularization with weight 10^{-4}. The Adam (Kingma and Ba, 2014) optimizer is used with $\beta_1 = 0.95$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and initial learning rate of 10^{-3}. Training is terminated if the lower bound on the development set does not improve for 50 epochs.

3.3.3 ASR Setup and Training

Kaldi (Povey et al., 2011) is used for feature extraction, decoding, forced alignment, and training of an initial HMM-GMM model on the original clean utterances. The recipe provided by the CHiME-4 challenge (`run_gmm.sh`) and the Kaldi Aurora-4 recipe are adapted by only changing the training data being used. The Computational

Network Toolkit (CNTK) (Yu et al., 2014) is used for neural network-based acoustic model training. For all CHiME-4 experiments, the same LSTM acoustic model (Sak et al., 2014) with the architecture proposed in Zhang et al. (2016) is applied, which has 1,024 memory cells and a 512-node projection layer for each LSTM layer, and 3 LSTM layers in total. Following the training setup in Hsu et al. (2016b), LSTM acoustic models are trained with a cross-entropy criterion, using truncated backpropagation-through-time (BPTT) (Williams and Peng, 1990) for optimization. Each BPTT segment contains 20 frames, and each mini-batch contains 80 utterances, since we find empirically parallelizing 80 utterances has similar performance to 40 utterances (Hsu et al., 2016a).

For all Aurora-4 experiments, the same 6 layer fully-connected deep neural network (DNN) acoustic model with 2,048 hidden units at each layer is applied, which is the same architecture as the one in Sun et al. (2017), except that the number of hidden units are doubled. The input to the DNN is a context window of 11 frames, with five frames of left context and five frames of right context. Each frame is represented using filter bank features with delta and delta-delta coefficients as proposed in Sun et al. (2017). DNN acoustic models are trained with the cross-entropy criterion, with a mini-batch size of 256. For both LSTM and DNN training, a momentum of 0.9 is used starting from the second epoch (Zhang et al., 2015). Ten percent of the training data is held out as a validation set to control the learning rate. The learning rate is halved when no gain is observed after an epoch.

We assume for both nuisance attribute replacement and latent nuisance subspace perturbation, the time alignment of senones does not change. Therefore, the same forced alignment is used to train the acoustic models.

3.4 Results and Discussion

In this section, we verify the effectiveness of the proposed VAE-based data augmentation methods for unsupervised domain adaptation. On each dataset, the same acoustic model architectures and training procedures, as well as the same language

models are used for all the experiments. For the CHiME-4 dataset, besides reporting the WER on the clean and the noisy development sets respectively, we also show the WER for the noisy set by the four recording locations: bus (BUS), cafe (CAF), pedestrian area (PED), and street junction (STR). All the CHiME-4 results are listed in Table 3.1. For the Aurora-4 dataset, we report the average WER as well as the WER in four conditions in Table 3.2. Different sets of experiments are separated by double horizontal lines and indexed by the *Exp. ID* on the first column. The second column, *Aug. Method*, explains the augmentation method and the hyper-parameter being used. The ratio of the new training set to the original clean training set is listed on the third column, referred to as the *Fold*.

3.4.1 Baselines

We first establish baselines by training models on two sets. The first set, *Orig.*, refers to the original clean training set that does not involve VAE. The second set, *Recon.*, refers to the reconstructed clean training set that is generated by using the VAE to first encode and then decode. Note that this does not involve the modification methods mentioned in Sections 3.2.3 and 3.2.4.

The results are listed in Table 3.1, *Exp. ID 1*. The fourth column shows the results on the matched domain (clean), and the fifth column shows the results on the mismatched domain (noisy). It can be observed that the performance degrades significantly when the models are tested on the mismatched domain. The WER increases from 19.04% to 87.08% for *Orig.*, and from 19.61% to 90.72% for *Recon.* respectively. In addition, since the reconstruction from the VAE is not perfect, part of the information may be lost during this process. Hence, the model trained on *Recon.* is slightly worse than the one trained on *Orig.* for all testing conditions. Lastly, the relative WERs of the four locations are consistent on both training sets. BUS appears to be the most difficult one, while PED is the easiest one among the four locations.

Setting			WER (%)		WER (%) by Noise Type			
Exp. ID	Aug. Method	Fold	Clean	Noisy	BUS	CAF	PED	STR
1	Orig.	1	**19.04**	**87.80**	96.16	92.35	78.46	84.24
	Recon.	1	19.61	90.72	98.95	93.45	81.52	88.97
2	Repl. Clean	1	**20.03**	67.12	71.99	76.84	55.32	64.33
	Repl. Noisy	1	26.31	**57.66**	62.12	69.25	46.89	52.38
3	Pert., $\gamma = 1.0$	1	20.01	**53.06**	55.66	66.12	41.94	48.50
	Uni-Pert., $\gamma = 1.0$	1	**19.70**	65.07	69.27	75.28	53.65	62.06
	Rev-Pert., $\gamma = 1.0$	1	19.75	87.98	95.13	90.58	76.71	89.50
4	Pert., $\gamma = 0.5$	1	**19.55**	65.61	67.87	77.37	54.54	62.66
	Pert., $\gamma = 1.0$	1	20.01	**53.06**	55.66	66.12	41.94	48.50
	Pert., $\gamma = 1.5$	1	19.99	53.59	57.09	64.91	42.23	50.11
	Pert., $\gamma = 2.0$	1	20.39	58.10	64.35	69.12	45.39	53.55
5	Orig. + Repl. Noisy	2	19.88	55.72	60.72	66.46	45.08	50.63
	Repl. Noisy	2	25.26	55.59	59.24	67.85	44.65	50.63
	Pert., $\gamma = 1.0$	2	**19.82**	**52.49**	55.52	65.04	41.17	48.24

Table 3.1: CHiME-4 development set word error rate of acoustic models trained on different augmented sets.

3.4.2 Replacing Nuisance Attributes

We evaluate the effectiveness of augmenting data by replacing the nuisance attributes as mentioned in Section 3.2.3. Let $\mathcal{U}_{src} = \{\boldsymbol{\mu}_{utt_j}\}_{j=1}^{M_{src}}$ be the set of latent nuisance representations of the source domain utterances, and $\mathcal{U}_{tar} = \{\boldsymbol{\mu}_{utt_j}\}_{j=M_{src}}^{M}$ be the set of latent nuisance representations of the target domain utterances. M_{src} is the number of source domain utterances, and $M_{tar} = M - M_{src}$ is the number of target domain utterances. We create the augmented set *Repl. Clean* by replacing the latent nuisance representation of each source domain utterance with one drawn from $\mathcal{U}_{src}$. The *Repl. Noisy* is generated similarly but is replaced with one drawn from $\mathcal{U}_{tar}$.

The results are shown in Table 3.1, *Exp. ID 2*. For both augmented methods, we observe at least 20% absolute WER reduction on the target domain compared to the baselines (*Exp. ID 1*). We observe an additional 10% absolute WER reduction when replacing the latent nuisance representations with those taken from the target domain instead of the source domain. We also observe that *Repl. Noisy* shows 6% worse WER on the source domain than *Repl. Clean*. The relative strength of *Repl.*

Clean and *Repl. Noisy* on different domains verifies the effectiveness of our proposed method at shifting the distribution from one domain to another.

3.4.3 Correctness of Soft Latent Nuisance Subspace Perturbation

We first examine the correctness of our proposed soft latent nuisance subspace perturbation by proposing two alternative perturbation methods. To eliminate the effect of the perturbation scale on the performance, we consider two alternative methods subject to the constraint that the expected squared Euclidean norm of the perturbation vector $\boldsymbol{p}$ is the same as the proposed method.

Recall that $\boldsymbol{p} = \gamma \sum_{d=1}^{D} \psi_d \sigma_d \boldsymbol{e}_d$ for our proposed method. We then have:

$$\mathbb{E}\left[||\boldsymbol{p}||_2^2\right] = \gamma^2 \sum_{d=1}^{D} \sigma_d^2 \, \mathbb{E}\left[\psi_d^2\right] = \gamma^2 \sum_{d=1}^{D} \sigma_d^2.$$

We first consider *uniform perturbation* to be a method that perturbs each direction with the same expected scale, controlled by the same perturbation ratio hyperparameter γ. The perturbation vector $\boldsymbol{p}_{uni}$ derived from this method can be formulated as:

$$\boldsymbol{p}_{uni} = \gamma \sum_{d=1}^{D} \psi_d \sigma_{uni} \boldsymbol{e}_d, \tag{3.1}$$

where $\sigma_{uni} = \sqrt{\sum_{d=1}^{D} \sigma_d^2 / D}$. In addition, we design another method that reverses the expected perturbation scales from the proposed soft latent nuisance subspace perturbation method, named the *reverse soft latent nuisance subspace perturbing*. This method perturbs a direction with the scale inversely correlated with the eigenvalue computed from the PCA analysis. The perturbation vector $\boldsymbol{p}_{rev}$ can be formulated as:

$$\boldsymbol{p}_{rev} = \gamma \sum_{d=1}^{D} \psi_d \sigma_{D-d} \boldsymbol{e}_d. \tag{3.2}$$

Both methods have the same expected squared Euclidean norm of the perturbation vector as the original proposed method when using the same perturbation ratio pa-

rameter γ.

We show the results of soft latent nuisance subspace perturbation (Pert.), uniform perturbation (Uni-Pert.), and reverse soft latent nuisance subspace perturbation (Rev-Pert.) in Table 3.1, *Exp. ID 3*, which all use the same perturbation ratio $\gamma = 1.0$. We can clearly observe the superiority of the proposed method among the three methods applied on the target domain. Pert. reduces the absolute WER by almost 35% from the baseline and outperforms Uni-Pert. by 12%, while Rev-Pert. achieves almost no improvement. This experiment verifies the importance of determining an appropriate way to perturb the latent space and the correctness of our method.

3.4.4 Effect of Perturbation Ratios

We next examine the effect on the hyper-parameter γ by choosing four scales: 0.5, 1.0, 1.5, 2.0, and list the results in Table 3.1, *Exp. ID 4*. We observe different WER trends for different perturbation ratios for the two testing conditions. First, regarding the target domain WER, we notice that $\gamma = 1.0$ reaches the best performance among the four scales. The smaller the perturbation ratio is, the more similar to the original clean data the augmented perturbed data would be. Hence, when we decrease the perturbation ratio, the performance would asymptotically approach those of the original clean data. On the other hand, as we increase the perturbation ratio, the chance of the perturbed utterances becoming linguistically different increases. This may hurt the performance and cancel out the benefit of having more diverse data by perturbing the nuisance attributes. As for the source domain WER, we observe degradation when increasing the perturbation ratio, because the perturbed data distribution becomes less similar to the original clean data distribution.

3.4.5 Effect of Augmented Dataset Size

In this section, we study the effect of the size by combining different sets of augmented data or the original data. Specifically, three cases are considered: (1) combining *Repl. Noisy* with the original data, (2) combining *Repl. Noisy* with another copy of *Repl.*

Noisy, and (3) combining *Pert.*, $\gamma = 1.0$ with another copy of *Pert.*, $\gamma = 1.0$.

The results are listed in Table 3.1, *Exp. ID 5*. In the first two cases, both source and target domain WERs are improved from the one-fold *Repl. Noisy*. While adding another copy of *Repl. Noisy* shows slightly better (0.13%) WER in the target domain of the first two cases, adding the original clean data significantly reduces (6.43%) WER in the source domain. This suggests that the second case addresses the issue of *Repl. Noisy* on shifting the data distribution entirely to the target domain. In the third case, a slight but consistent 0.19% and 0.57% WER reductions from the one-fold *Pert.*, $\gamma = 1.0$ in the source and target domain are observed. In summary, all three cases show improvement by increasing the size.

3.4.6 Comparing with DDA on Aurora-4

In this section, we repeat the experiments on the Aurora-4 dataset and compare with deep domain adaptation (Sun et al., 2017). Table 5.4 listed our Aurora-4 results and the reference results. DNN-PP (Du et al., 2014) is a method compared in Sun et al. (2017) that requires parallel clean-noisy data for training a speech enhancement model as a preprocessor.

Baseline results are established in Table 5.4, *Exp. ID 1*, where the models are trained with the original clean features (*Orig.*), and the VAE-reconstructed clean features (*Recon.*), respectively. Here we can observe significant degradation on mismatched domains (B, C, and D) from the matched domain (A). Results of nuisance attribute replacement and soft latent nuisance subspace perturbation with different perturbation ratios are shown in Table 5.4, *Exp. ID 2* and *Exp. ID 3*. Both augmentation methods achieve roughly 30% absolute WER reduction, and the soft latent nuisance subspace perturbation reaches the best performance when using a perturbation ratio $\gamma = 2.0$. By increasing the dataset size, we observe further WER reduction from two-fold to 16-fold.

Since the detailed training recipe is not provided in Sun et al. (2017), we could not reproduce exactly the same baseline results. However, despite the fact that our baseline is worse than that in Sun et al. (2017) by 17.76% absolute WER, our best

Setting			WER (%)	WER (%) by Condition			
Exp. ID	Aug. Method/Baselines	Fold	Avg.	A	B	C	D
0	Clean-DNN-HMM (Sun et al., 2017)	-	36.22	3.36	29.74	21.02	50.73
	DDA-DNN-HMM (Sun et al., 2017)	-	22.53	3.24	14.52	17.82	34.55
	DNN-PP (Du et al., 2014)	-	18.7	5.1	12.0	10.5	29.0
1	Orig.	1	53.98	3.38	50.56	42.67	67.70
	Recon.	1	66.29	4.58	65.44	51.02	79.97
2	Repl. Noisy	1	22.53	4.80	16.31	14.72	32.99
3	Pert., $\gamma = 0.5$	1	35.37	4.11	27.73	33.51	48.52
	Pert., $\gamma = 1.0$	1	24.82	4.35	17.11	22.38	36.36
	Pert., $\gamma = 1.5$	1	21.98	4.24	15.08	16.87	32.69
	Pert., $\gamma = 2.0$	1	20.68	4.45	14.33	14.74	30.72
	Pert., $\gamma = 2.5$	1	20.99	4.99	15.35	15.54	30.22
	Pert., $\gamma = 3.0$	1	21.18	5.29	15.47	15.71	30.45
	Pert., $\gamma = 3.5$	1	21.33	5.45	16.13	14.70	30.29
	Pert., $\gamma = 4.0$	1	22.00	6.43	17.15	15.00	30.62
4	Pert., $\gamma = 2.0$	2	20.06	4.13	13.85	14.96	29.77
	Pert., $\gamma = 2.0$	4	19.42	4.09	13.34	14.14	28.92
	Pert., $\gamma = 2.0$	8	18.86	4.28	12.89	13.51	28.16
	Pert., $\gamma = 2.0$	16	**18.76**	4.04	12.84	13.54	28.01

Table 3.2: Aurora-4 test_eval92 set word error rate of acoustic models trained on different augmented sets.

system (18.76%) still achieves better performance than the result of DDA (22.53%) that was reported in Sun et al. (2017), and matches the results of DNN-PP (Du et al., 2014) without the need for parallel data.

3.5 Conclusions

In this chapter, we presented two VAE-based data augmentation methods for unsupervised domain adaptation for robust ASR. In particular, we studied the latent representations obtained from VAEs, which enabled us to transform nuisance attributes of speech through modifying the latent variables. Our proposed methods were evaluated on two datasets, and achieved about 35% absolute WER reduction on both sets.

Chapter 4

Learning Disentangled and Interpretable Representations

Despite successes with VAEs, understanding the underlying factors represented by latent variables is a major challenge. Some research focuses on the supervised or semi-supervised setting using VAEs (Kingma et al., 2014; Hu et al., 2017). Another line of research attempts to develop weakly supervised or unsupervised methods to learn disentangled representations, such as DC-IGN (Kulkarni et al., 2015), InfoGAN (Chen et al., 2016), and β-VAE (Higgins et al., 2016). In the previous chapters, we proposed post-training analysis to associate physical attributes with learned representations. This is a step toward unsupervised learning of interpretable representations; however, such an approach is not satisfactory as we still need labeled data for analysis.

While there has been much research investigating learning from static data (e.g., images), there is relatively little research on learning from sequential data (Fabius and van Amersfoort, 2014; Chung et al., 2015b, 2016; Fraccaro et al., 2016; Edwards and Storkey, 2016; Johnson et al., 2016; Serban et al., 2017). Moreover, to the best of our knowledge, there has not been any attempt to learn disentangled and interpretable representations without supervision from sequential data. The information encoded in sequential data, such as speech, video, and text, is naturally multi-scaled; in speech for example, information about the channel, speaker, and linguistic content is encoded in the statistics at the session, utterance, and segment levels, respectively. By leveraging

this source of constraint, we can learn disentangled and interpretable factors in an unsupervised manner.

In this chapter, we propose a novel factorized hierarchical variational autoencoder (FHVAE), which learns disentangled and interpretable latent representations from sequential data without supervision by explicitly modeling the multi-scaled information with a factorized hierarchical graphical model. The inference model is designed such that the model can be optimized at the segment level, instead of at the sequence level, which may cause scalability issues when sequences become too long. A sequence-to-sequence neural network architecture is applied to better capture temporal relationships. We evaluate the proposed model on two speech datasets. Qualitatively, the model demonstrates an ability to factorize sequence-level and segment-level attributes into different sets of latent variables. Quantitatively, the model achieves 2.38% and 1.34% equal error rate on unsupervised and supervised speaker verification tasks respectively, which outperforms an i-vector baseline. On speech recognition tasks, it reduces the word error rate in mismatched train/test scenarios by up to 35%. This chapter includes partial content published in Hsu et al. (2017b).

4.1 Factorized Hierarchical Variational Autoencoder

4.1.1 A Factorized Hierarchical Generative Process

Generation of sequential data, such as speech, often involves multiple independent factors operating at different time scales. For instance, the speaker identity affects fundamental frequency (F0) and volume at the sequence level, while phonetic content only affects spectral contour and durations of formants at the segmental level. This multi-scale behavior results in the fact that some attributes, such as F0 and volume, tend to have a smaller amount of variation within an utterance, compared to between utterances; however, other attributes, such as phonetic content, tend to have a similar amount of variation within and between utterances.

We refer to the first type of attributes as *sequence-level attributes*, and the other as

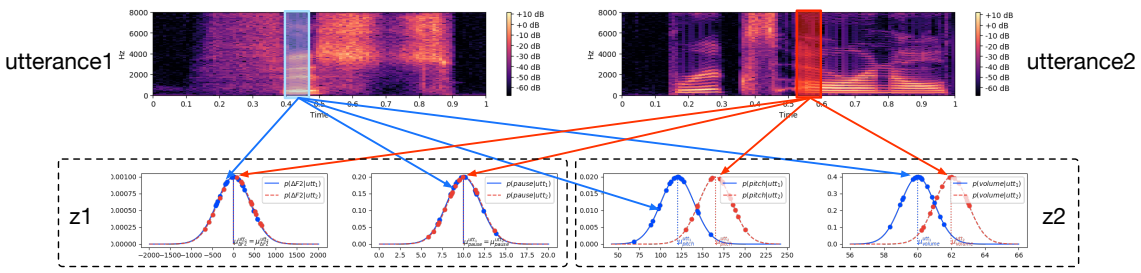


Figure 4-1: Graphical explanation of sequence-level and segment-level attributes distribution for two different utterances. Each dot denotes the value of a particular attribute of a segment. Distributions of some attributes of a segment, such as the phonetic content, do not vary between utterances, as shown on the lower left of the figure. On the other hand, distributions of other attributes of a segment, such as the fundamental frequency, have sequence-dependent distributions, as shown on the lower right of the figure. We want to encode the first type of attributes into one set of latent variables ($\boldsymbol{z}_1$), and the second type of attributes into the other set of latent variables ($\boldsymbol{z}_2$).

segment-level attributes. We aim to achieve disentanglement and interpretability by encoding the two types of attributes into *latent sequence variables* and *latent segment variables* respectively, where the former is regularized by a sequence-dependent prior and the latter by a sequence-independent prior. Figure 4-1 illustrates our assumptions on the distribution of the sequence-level and segment-level attributes of different utterances.

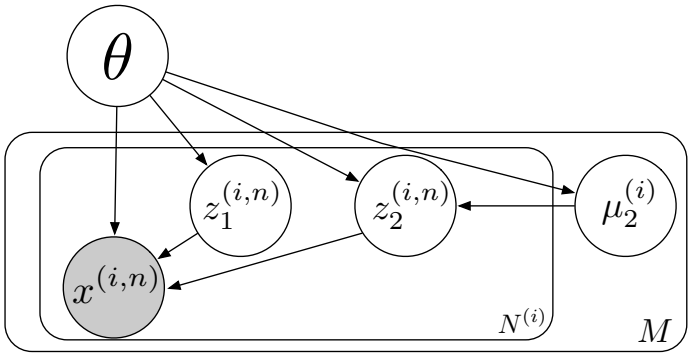


Figure 4-2: Graphical illustration of the proposed generative model. Grey nodes denote the observed variables, and white nodes are the hidden variables.

We now formulate a generative process for sequential data and propose our Factorized Hierarchical Variational Autoencoder (FHVAE). Consider some dataset $\mathcal{D} =$

$\{\boldsymbol{X}^{(i)}\}_{i=1}^{M}$ consisting of M i.i.d. sequences, where $\boldsymbol{X}^{(i)} = \{\boldsymbol{x}^{(i,n)}\}_{n=1}^{N^{(i)}}$ is a sequence of $N^{(i)}$ observed variables. $N^{(i)}$ is referred to as the *number of segments* for the i-th sequence, and $\boldsymbol{x}^{(i,n)}$ is referred to as the n-th *segment* of the i-th sequence. Note that here a "segment" refers to a variable of smaller temporal scale compared to the "sequence", which is in fact a sub-sequence. We will drop the index i whenever it is clear that we are referring to terms associated with a single sequence. We assume that each sequence $\boldsymbol{X}$ is generated from some random process involving the latent variables $\boldsymbol{Z}_1 = \{\boldsymbol{z}_1^{(n)}\}_{n=1}^{N}$, $\boldsymbol{Z}_2 = \{\boldsymbol{z}_2^{(n)}\}_{n=1}^{N}$, and $\boldsymbol{\mu}_2$. The following generation process as illustrated in Figure 4-2 is considered:

(1) an *s-vector* $\boldsymbol{\mu}_2$ is drawn from a prior distribution $p(\boldsymbol{\mu}_2)$;

(2) N i.i.d. *latent sequence variables* $\{\boldsymbol{z}_2^{(n)}\}_{n=1}^{N}$ are drawn from a sequence-dependent prior distribution $p(\boldsymbol{z}_2|\boldsymbol{\mu}_2)$;

(3) N i.i.d. *latent segment variables* $\{\boldsymbol{z}_1^{(n)}\}_{n=1}^{N}$ are drawn from a sequence-independent prior distribution $p(\boldsymbol{z}_1)$;

(4) N i.i.d. observed variables $\{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$ are drawn from a conditional distribution $p(\boldsymbol{x}|\boldsymbol{z}_1, \boldsymbol{z}_2)$, where $\boldsymbol{x}^{(n)}$ is conditioned on $\boldsymbol{z}_1^{(n)}$ and $\boldsymbol{z}_2^{(n)}$.

The joint probability for a sequence is formulated in Eq. 4.1:

$$p(\boldsymbol{x}, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{\mu}_2) = p(\boldsymbol{\mu}_2) \prod_{n=1}^{N} p(\boldsymbol{x}^{(n)}|\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)}) p(\boldsymbol{z}_1^{(n)}) p(\boldsymbol{z}_2^{(n)}|\boldsymbol{\mu}_2). \tag{4.1}$$

Specifically, we formulate each of the RHS term as follows:

$$p(\boldsymbol{x}|\boldsymbol{z}_1, \boldsymbol{z}_2) = \mathcal{N}(\boldsymbol{x}|f_{\mu_x}(\boldsymbol{z}_1, \boldsymbol{z}_2), diag(f_{\sigma_x^2}(\boldsymbol{z}_1, \boldsymbol{z}_2))) \tag{4.2}$$

$$p(\boldsymbol{z}_1) = \mathcal{N}(\boldsymbol{z}_1|\boldsymbol{0}, \sigma_{\boldsymbol{z}_1}^2 \boldsymbol{I}), \tag{4.3}$$

$$p(\boldsymbol{z}_2|\boldsymbol{\mu}_2) = \mathcal{N}(\boldsymbol{z}_2|\boldsymbol{\mu}_2, \sigma_{\boldsymbol{z}_2}^2 \boldsymbol{I}), \tag{4.4}$$

$$p(\boldsymbol{\mu}_2) = \mathcal{N}(\boldsymbol{\mu}_2|\boldsymbol{0}, \sigma_{\boldsymbol{\mu}_2}^2 \boldsymbol{I}), \tag{4.5}$$

where the priors over the s-vectors $\boldsymbol{\mu}_2$ and the latent segment variables $\boldsymbol{z}_1$ are centered

isotropic multivariate Gaussian distributions. The prior over the latent sequence variable $\boldsymbol{z}_2$ conditioned on $\boldsymbol{\mu}_2$ is an isotropic multivariate Gaussian centered at $\boldsymbol{\mu}_2$. The conditional distribution of the observed variable $\boldsymbol{x}$ is the multivariate Gaussian with a diagonal covariance matrix, whose mean and diagonal variance are parameterized by neural networks $f_{\mu_x}(\cdot,\cdot)$ and $f_{\sigma_x^2}(\cdot,\cdot)$ with input $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. We use θ to denote the set of parameters in the generative model.

This generative model is factorized in a way such that the latent sequence variables $\boldsymbol{z}_2$ within a sequence are forced to be close to $\boldsymbol{\mu}_2$ as well as to each other in Euclidean distance, and therefore are encouraged to encode sequence-level attributes that may have larger variance across sequences, but smaller variance within sequences. The constraint on the latent segment variables $\boldsymbol{z}_1$ is imposed globally, and therefore encourages encoding of residual attributes whose variation is not distinguishable between and within sequences.

4.1.2 An Inference Model

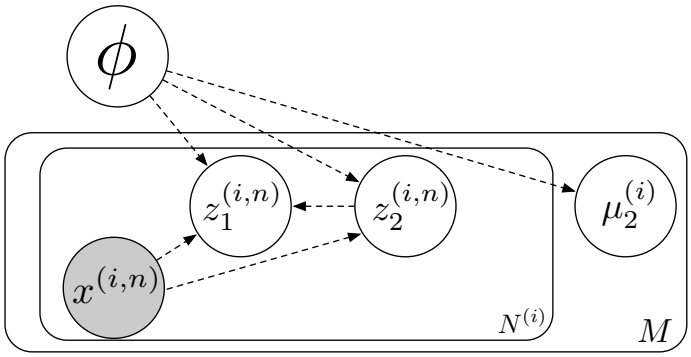


Figure 4-3: Graphical illustration of the proposed inference model. Grey nodes denote the observed variables, and white nodes are the hidden variables.

In the variational autoencoder framework, since the exact posterior inference is intractable, an inference model, $q(\boldsymbol{z}_1^{(i)}, \boldsymbol{z}_2^{(i)}, \boldsymbol{\mu}_2^{(i)}|\boldsymbol{X}^{(i)})$, that approximates the true posterior, $p(\boldsymbol{z}_1^{(i)}, \boldsymbol{z}_2^{(i)}, \boldsymbol{\mu}_2^{(i)}|\boldsymbol{X}^{(i)})$, for variational inference (Jordan et al., 1999) is introduced.

We consider the following inference model as Figure 4-3:

$$q(\boldsymbol{z}_1^{(i)}, \boldsymbol{z}_2^{(i)}, \boldsymbol{\mu}_2^{(i)} | \boldsymbol{X}^{(i)}) = q(\boldsymbol{\mu}_2^{(i)}) \prod_{n=1}^{N^{(i)}} q(\boldsymbol{z}_1^{(i,n)} | \boldsymbol{x}^{(i,n)}, \boldsymbol{z}_2^{(i,n)}) q(\boldsymbol{z}_2^{(i,n)} | \boldsymbol{x}^{(i,n)}) \tag{4.6}$$

$$q(\boldsymbol{\mu}_2^{(i)}) = \mathcal{N}(\boldsymbol{\mu}_2^{(i)} | h_{\mu_{\boldsymbol{\mu}_2}}(i), \sigma_{\tilde{\boldsymbol{\mu}}_2}^2 \boldsymbol{I}) \tag{4.7}$$

$$q(\boldsymbol{z}_2 | \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}_2 | g_{\mu_{\boldsymbol{z}_2}}(\boldsymbol{x}), diag(g_{\sigma^2_{\boldsymbol{z}_2}}(\boldsymbol{x}))) \tag{4.8}$$

$$q(\boldsymbol{z}_1 | \boldsymbol{x}, \boldsymbol{z}_2) = \mathcal{N}(\boldsymbol{z}_1 | g_{\mu_{\boldsymbol{z}_1}}(\boldsymbol{x}, \boldsymbol{z}_2), diag(g_{\sigma^2_{\boldsymbol{z}_1}}(\boldsymbol{x}, \boldsymbol{z}_2))), \tag{4.9}$$

where the posteriors over $\boldsymbol{\mu}_2$, $\boldsymbol{z}_1$, and $\boldsymbol{z}_2$ are all multivariate diagonal Gaussian distributions. Note that the mean of the posterior distribution of $\boldsymbol{\mu}_2$ is not directly inferred from $\boldsymbol{X}$, but instead is regarded as part of the inference model parameters, with one for each utterance, which would be optimized during training. Therefore, $h_{\mu_{\boldsymbol{\mu}_2}}(\cdot)$ can be seen as a lookup table, and we use $\tilde{\boldsymbol{\mu}}_2^{(i)} = h_{\mu_{\boldsymbol{\mu}_2}}(i)$ to denote the posterior mean of $\boldsymbol{\mu}_2$ for the i-th sequence; we fix the posterior covariance matrix of $\boldsymbol{\mu}_2$ for all sequences. Similar to the generative model, $g_{\mu_{\boldsymbol{z}_2}}(\cdot)$, $g_{\sigma^2_{\boldsymbol{z}_2}}(\cdot)$, $g_{\mu_{\boldsymbol{z}_1}}(\cdot,\cdot)$, and $g_{\sigma^2_{\boldsymbol{z}_1}}(\cdot,\cdot)$ are also neural networks whose parameters along with $h_{\mu_{\boldsymbol{\mu}_2}}(\cdot)$ are denoted collectively by ϕ. The variational lower bound for this inference model on the marginal likelihood of a sequence $\boldsymbol{X}$ is derived as follows:

$$\begin{aligned}
\mathcal{L}(\theta, \phi; \boldsymbol{X}) =& \sum_{n=1}^{N} \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)} | \tilde{\boldsymbol{\mu}}_2) + \log p(\tilde{\boldsymbol{\mu}}_2) + const \\
\mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)} | \tilde{\boldsymbol{\mu}}_2) =& \mathbb{E}_{q(\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)} | \boldsymbol{x}^{(n)})} \big[\log p(\boldsymbol{x}^{(n)} | \boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)}) \big] \\
& - \mathbb{E}_{q(\boldsymbol{z}_2^{(n)} | \boldsymbol{x}^{(n)})} \big[D_{KL}(q(\boldsymbol{z}_1^{(n)} | \boldsymbol{x}^{(n)}, \boldsymbol{z}_2^{(n)}) || p(\boldsymbol{z}_1^{(n)})) \big] \\
& - D_{KL}(q(\boldsymbol{z}_2^{(n)} | \boldsymbol{x}^{(n)}) || p(\boldsymbol{z}_2^{(n)} | \tilde{\boldsymbol{\mu}}_2)).
\end{aligned}$$

The detailed derivation can be found in Appendix A. Because the approximated posterior of $\boldsymbol{\mu}_2$ does not depend on the sequence $\boldsymbol{X}$, the *sequence variational lower bound* $\mathcal{L}(\theta, \phi; \boldsymbol{X})$ can be decomposed into the sum of $\mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)} | \tilde{\boldsymbol{\mu}}_2)$, the *conditional segment variational lower bounds*, over segments, plus the log prior probability of $\tilde{\boldsymbol{\mu}}_2$ and a constant. Therefore, instead of sampling a batch at the sequence level to

maximize the sequence variational lower bound, we can sample a batch at the segment level to maximize the *segment variational lower bound*:

$$\mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)}) = \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)}|\tilde{\boldsymbol{\mu}}_2) + \frac{1}{N} \log p(\tilde{\boldsymbol{\mu}}_2) + const. \tag{4.10}$$

This approach provides better scalability when the sequences are extremely long, such that computing an entire sequence for a batched update is too computationally expensive.

Here we only introduce two scales of attributes; however, one can easily extend this model to more scales by simply introducing $\boldsymbol{\mu}_k$ for $k = 2, 3, \cdots$[1] that constrains the prior distribution of latent variables at more scales, such as having a session-dependent prior or dataset-dependent prior.

4.1.3 An Unsupervised Discriminative Objective

The idea of having sequence-specific priors for each sequence is to encourage the model to encode the sequence-level attributes and the segment-level attributes into different sets of latent variables. However, when $\boldsymbol{\mu}_2$ is the same for all sequences, the prior probability is maximized, and the KL-divergence of the inferred posterior of $\boldsymbol{z}_2$ is measured from the same conditional prior for all sequences. This would result in trivial s-vectors $\boldsymbol{\mu}_2$, and therefore $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ would not be factorized to encode sequence and segment attributes respectively.

To encourage $\boldsymbol{z}_2$ to encode sequence-level attributes, we use $\boldsymbol{z}_2^{(i,n)}$, which is inferred from $\boldsymbol{x}^{(i,n)}$, to infer the sequence index i of $\boldsymbol{x}^{(i,n)}$. We formulate the discriminative objective as:

$$\begin{aligned}\log p(i|\boldsymbol{z}_2^{(i,n)}) &= \log p(\boldsymbol{z}_2^{(i,n)}|i) - \log \sum_{j=1}^{M} p(\boldsymbol{z}_2^{(i,n)}|j) \quad (p(i) \text{ is assumed uniform}) \\ &:= \log p(\boldsymbol{z}_2^{(i,n)}|\tilde{\boldsymbol{\mu}}_2^{(i)}) - \log \big(\sum_{j=1}^{M} p(\boldsymbol{z}_2^{(i,n)}|\tilde{\boldsymbol{\mu}}_2^{(j)}) \big),\end{aligned}$$

[1]The index starts from 2 because we do not introduce the hierarchy to $\boldsymbol{z}_1$.

Combining the discriminative objective using a weighting parameter α with the segment variational lower bound, the objective function to maximize then becomes:

$$\mathcal{L}^{dis}(\theta, \phi; \boldsymbol{x}^{(i,n)}) = \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(i,n)}) + \alpha \log p(i|\boldsymbol{z}_2^{(i,n)}), \tag{4.11}$$

which we refer to as the *discriminative segment variational lower bound*.

4.1.4 Inferring S-Vectors During Testing

During testing, we may want to use the s-vector $\boldsymbol{\mu}_2$ of an unseen sequence $\boldsymbol{X} = \{\boldsymbol{x}^{(n)}\}_{n=1}^N$ as the sequence-level attribute representation for tasks such as speaker verification. Since the exact maximum a posterior estimation of $\boldsymbol{\mu}_2$ is intractable, we approximate the estimation using the conditional segment variational lower bound as follows:

$$\begin{aligned}
\boldsymbol{\mu}_2^* &= \operatorname*{argmax}_{\boldsymbol{\mu}_2} \log p(\boldsymbol{\mu}_2|\boldsymbol{X}) = \operatorname*{argmax}_{\boldsymbol{\mu}_2} \log p(\boldsymbol{X}, \boldsymbol{\mu}_2) \\
&= \operatorname*{argmax}_{\boldsymbol{\mu}_2} \big(\sum_{n=1}^N \log p(\boldsymbol{x}^{(n)}|\boldsymbol{\mu}_2) \big) + \log p(\boldsymbol{\mu}_2) \\
&\approx \operatorname*{argmax}_{\boldsymbol{\mu}_2} \sum_{n=1}^N \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)}|\boldsymbol{\mu}_2) + \log p(\boldsymbol{\mu}_2).
\end{aligned} \tag{4.12}$$

The closed form solution of $\boldsymbol{\mu}_2^*$ can be derived by differentiating Eq. 4.12 w.r.t. $\boldsymbol{\mu}_2$ (see Appendix B):

$$\boldsymbol{\mu}_2^* = \frac{\sum_{n=1}^N g_{\mu_{\boldsymbol{z}_2}}(\boldsymbol{x}^{(n)})}{N + \sigma_{\boldsymbol{z}_2}^2 / \sigma_{\boldsymbol{\mu}_2}^2}. \tag{4.13}$$

4.1.5 Sequence-to-Sequence Autoencoder Model Architecture

In this section, we introduce the detailed neural network architectures for our proposed FHVAE. Let a segment $\boldsymbol{x} = x_{1:T}$ be a sub-sequence of $\boldsymbol{X}$ that contains T time steps, and x_t denotes the t-th time step of $\boldsymbol{x}$. We use recurrent network architectures for encoders that capture the temporal relationship among time steps, and generate

a summarized fixed-dimension vector after consuming an entire sub-sequence. Likewise, we adopt a recurrent network architecture that generates a frame step by step conditioned on the latent variables $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. The complete network can be seen as a stochastic sequence-to-sequence autoencoder that encodes $x_{1:T}$ stochastically into $\boldsymbol{z}_1, \boldsymbol{z}_2$, and stochastically decodes from them back to $x_{1:T}$.

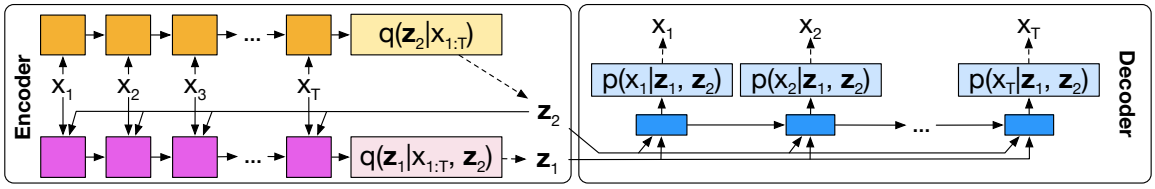


Figure 4-4: Sequence-to-sequence factorized hierarchical variational autoencoder. Dashed lines indicate the sampling process using the reparameterization trick (Kingma and Welling, 2013). The encoders for $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are pink and amber, respectively, while the decoder for $\boldsymbol{x}$ is blue. Darker colors denote the recurrent neural networks, while lighter colors denote the fully-connected layers predicting the mean and log variance.

Figure 4-4 shows our proposed Seq2Seq-FHVAE architecture.[2] Here we show the detailed formulation:

$$
\begin{aligned}
(\boldsymbol{h}_{\boldsymbol{z}_2,t}, \boldsymbol{c}_{\boldsymbol{z}_2,t}) &= \mathrm{LSTM}(x_{t-1}, \boldsymbol{h}_{\boldsymbol{z}_2,t-1}, \boldsymbol{c}_{\boldsymbol{z}_2,t-1}; \phi_{\mathrm{LSTM},\boldsymbol{z}_2}) \\
q(\boldsymbol{z}_2|x_{1:T}) &= \mathcal{N}(\boldsymbol{z}_2|\, \mathrm{MLP}(\boldsymbol{h}_{\boldsymbol{z}_2,T}; \phi_{\mathrm{MLP}_{\boldsymbol{\mu}},\boldsymbol{z}_2}), \mathrm{diag}(\exp(\mathrm{MLP}(\boldsymbol{h}_{\boldsymbol{z}_2,T}; \phi_{\mathrm{MLP}_{\boldsymbol{\sigma}^2},\boldsymbol{z}_2})))) \\
(\boldsymbol{h}_{\boldsymbol{z}_1,t}, \boldsymbol{c}_{\boldsymbol{z}_1,t}) &= \mathrm{LSTM}([x_{t-1}; \boldsymbol{z}_2], \boldsymbol{h}_{\boldsymbol{z}_1,t-1}, \boldsymbol{c}_{\boldsymbol{z}_1,t-1}; \phi_{\boldsymbol{z}_1}) \\
q(\boldsymbol{z}_1|x_{1:T}, \boldsymbol{z}_2) &= \mathcal{N}(\boldsymbol{z}_1|\, \mathrm{MLP}(\boldsymbol{h}_{\boldsymbol{z}_1,T}; \phi_{\mathrm{MLP}_{\boldsymbol{\mu}},\boldsymbol{z}_1}), \mathrm{diag}(\exp(\mathrm{MLP}(\boldsymbol{h}_{\boldsymbol{z}_1,T}; \phi_{\mathrm{MLP}_{\boldsymbol{\sigma}^2},\boldsymbol{z}_1})))) \\
(\boldsymbol{h}_{\boldsymbol{x},t}, \boldsymbol{c}_{\boldsymbol{x},t}) &= \mathrm{LSTM}([\boldsymbol{z}_1; \boldsymbol{z}_2], \boldsymbol{h}_{\boldsymbol{x},t-1}, \boldsymbol{c}_{\boldsymbol{x},t-1}; \phi_{\boldsymbol{x}}) \\
p(x_t|\boldsymbol{z}_1, \boldsymbol{z}_2) &= \mathcal{N}(\boldsymbol{x}_t|\, \mathrm{MLP}(\boldsymbol{h}_{\boldsymbol{x},t}; \phi_{\mathrm{MLP}_{\boldsymbol{\mu}},\boldsymbol{x}}), \mathrm{diag}(\exp(\mathrm{MLP}(\boldsymbol{h}_{\boldsymbol{x},t}; \phi_{\mathrm{MLP}_{\boldsymbol{\sigma}^2},\boldsymbol{x}})))),
\end{aligned}
$$

where LSTM refers to a long short-term memory recurrent neural network (Hochreiter and Schmidhuber, 1997), and MLP refers to a multi-layer perceptron, ϕ_* are the related weight matrices. None of the neural network parameters are shared. We refer to this model as Seq2Seq-FHVAE.

[2]Best viewed in color.

4.2 Experimental Setup

4.2.1 Datasets and Surface Representations

The following two corpora are used for our experiments: (1) **TIMIT** (Garofolo et al., 1993), which contains broadband 16kHz recordings of phonetically-balanced read speech. A total of 6300 utterances (5.4 hours) are presented with 10 sentences from each of 630 speakers, of which approximately 70% are male and 30% are female. (2) **Aurora-4** (Pearce, 2002), a broadband corpus designed for noisy speech recognition tasks based on the Wall Street Journal corpus (WSJ0) (Paul and Baker, 1992). Two microphone types, CLEAN/CHANNEL are included, and six noise types are artificially added to both microphone types, which results in four conditions: CLEAN, CHANNEL, NOISY, and CHANNEL+NOISY. Two 14 hour training sets are used, where one is clean and the other is a mix of all four conditions. The same noise types and microphones are used to generate the development and test sets, which both consist of 330 utterances from all four conditions, resulting in 4,620 utterances in total for each set.

All speech is represented as a sequence of 80 dimensional Mel-scale filter bank (FBank) features or 200 dimensional log-magnitude spectrum (only for audio reconstruction), computed every 10ms. Mel-scale features are a popular auditory approximation for many speech applications (Mogran et al., 2004). We consider a sample $\boldsymbol{x}$ to be a 200ms sub-sequence, which is on the order of the length of a syllable, and implies $T = 20$ for each $\boldsymbol{x}$.

4.2.2 FHVAE Model and Training Configurations

For the Seq2Seq-FHVAE model, each $LSTM$ network consists of one layer with 256 hidden units, while each MLP network is one layer with the output dimension equal to the variable whose mean or log variance the MLP parameterizes, and variances $\sigma^2_{\boldsymbol{z}_1} = \sigma^2_{\boldsymbol{\mu}_2} = 1$, $\sigma^2_{\boldsymbol{z}_2} = 0.25$. We experiment with various dimensions for the latent variable $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. All models were trained with stochastic gradient descent using

a mini-batch size of 256 to minimize the negative discriminative segment variational lower bound plus an $L2$-regularization with weight 10^{-4}. The Adam (Kingma and Ba, 2014) optimizer is used with $\beta_1 = 0.95$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and initial learning rate of 10^{-3}. Training continues for 100 epochs unless the segment variational lower bound on the development set does not improve for 10 epochs. The $\boldsymbol{\mu}_2$ for the sequences in the development set and the test set is estimated using the closed form solution in Section 4.1.4.

4.3 Comparison of FHVAE Model Architectures

Here we study the performance of our proposed architecture by replacing the LSTM module with three baseline architectures: a fully-connected feed-forward network (FC), a vanilla recurrent neural network (RNN), and a gated recurrent neural network (GRU) (Chung et al., 2015a). All the models have one hidden layer with 16 dimensions for both $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, and are trained with $\alpha = 0$. For the FC model, the entire segment is flattened and fed to the fully-connected layers; therefore the temporal structure is simply ignored.

Table 4.1 shows the segment variational lower bound on the TIMIT test set. We can see that the recurrent models (RNN, GRU, LSTM) outperform the feed-forward model using fewer parameters, which demonstrates the importance of considering the temporal structure within a segment. Figure 4-5 shows the reconstruction results using the FC model and the LSTM model. The LSTM model reconstructs sharper images that preserves more speech detail, and, in particular, presents superior high frequency harmonic structure that does the FC model, as highlighted in the red boxes.

Table 4.1: TIMIT test set segment variational lower bound results on different model architectures.

Models	#Hidden Units	#Params	$\mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)})$
FC	512	3.3M	-348.63
RNN	256	0.3M	-261.19
GRU	256	0.8M	-158.42
LSTM	256	1.1M	**-143.80**

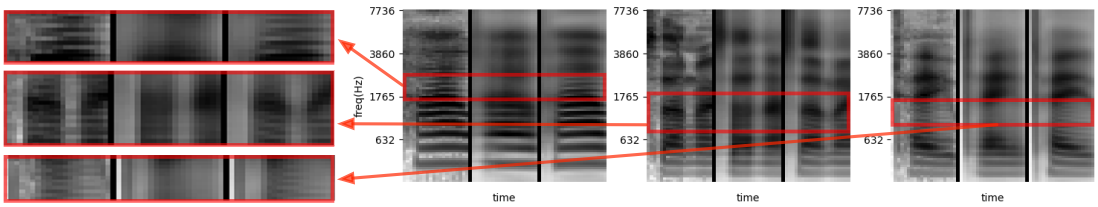


Figure 4-5: Three examples from different speakers. Within each example, from left to right are 1) the original segment, 2) FC reconstructed segment, and 3) LSTM reconstructed segment. The leftmost images show expanded views of the higher frequency harmonic structure (horizontal dark bands) of the spectrogram suggesting that the LSTM reconstruction is superior to the FC model.

4.4 Visualizing Latent Space Factorization

4.4.1 Re-combining Latent Segment and Sequence Variables

To qualitatively study the factorization of information between the latent segment variable $\boldsymbol{z}_1$ and the latent sequence variable $\boldsymbol{z}_2$, we generate examples $\boldsymbol{x}$ by varying each of them respectively. Figure 4-6 shows 40 examples in block 'C' of all the combinations of the 4 latent segment variables extracted from block 'A' and the 10 latent sequence variables extracted from block 'B.' The top two examples from block 'A' and the five leftmost examples from block 'B' are from male speakers, while the rest are from female speakers, which show higher fundamental frequencies and harmonics.[3]

We can observe that along each row in block 'C,' the linguistic phonetic-level content, which manifests itself in the form of the spectral contour and temporal position of formants, as well as the relative position between formants, is very similar between elements; the speaker identity however changes (e.g., harmonic structure). On the other hand, for each column we see that the speaker identity remains consistent, despite the change of linguistic content. The factorization of the sequence-level attributes and the segment-level attributes of our proposed Seq2Seq-FHVAE is clearly evident.

[3]The harmonics corresponds to horizontal dark stripes in the figure; the more widely these stripes are spaced vertically, the higher the fundamental frequency of the speaker is.

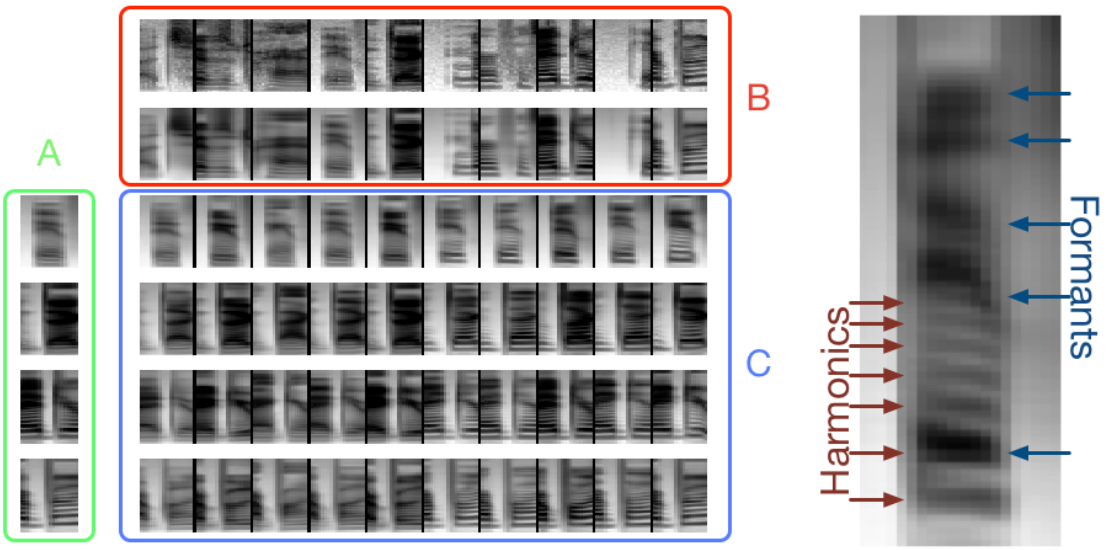


Figure 4-6: (left) Examples generated by varying different latent variables of a FHVAE model trained with $\alpha = 10$ on TIMIT dataset. The green block 'A' contains four reconstructed examples. The red block 'B' contains ten original examples on the first row and the corresponding reconstructed examples on the second row. The entry on the i-th row and the j-th column in the blue block 'C' is the reconstructed example using the latent segment variable $\boldsymbol{z}_1$ of the i-th row from the block 'A' and the latent sequence variable $\boldsymbol{z}_2$ of the j-th column from the block 'B.' (right) An illustration of harmonics and formants in filter bank images.

In Figures 4-7 and 4-8, we illustrate the results of the same experiments, but use the model trained on the Aurora-4 corpus instead. In particular, we sample two speakers, 441 and 443, from the test set and choose four noise conditions: clean, car, babble, and restaurant, without the microphone channel effect. Furthermore, since the noise is artificially added to each clean utterance in the test set, we can actually choose the corresponding segment in different noise conditions for a given speaker. The same eight examples are used in both block 'A' and block 'B,' which results in 64 combinations of latent segment variables and latent sequence variables in total. It can be observed that the latent sequence variables capture not only the speaker information, but also the noise information, which are both sequence-level attributes. Therefore, when modifying the latent sequence variables, we can not only transform speaker identities, but also carry out denoising or noise corruption. Moreover, the

disentanglement is evident for both the model trained without discriminative training ($\alpha = 0$) and the model trained with discriminative training ($\alpha = 10$).

4.4.2 Walking in the Latent Space

In this section, we present a qualitative analysis of traversing a single dimension of latent sequence variable or latent segment variable over the range $[-3, 3]$, while keeping the remaining latent variables fixed. In Figures 4-9, 4-10, 4-11 and 4-12, each row corresponds to a different seed $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ pair, inferred from some seed segment randomly drawn from the test set. The leftmost column in each figure shows the seed segments for each row. We use the same five seed segments for traversing each latent variable. The FHVAE model is trained on TIMIT with $\alpha = 0$, and a 200 dimensional log-magnitude spectrum is used for frame feature representations.

Figures 4-9 and 4-10 show examples of traversing five different dimensions of latent segment variable $\boldsymbol{z}_1$, while keeping the latent sequence variables fixed. It can be observed that these latent segment variables encode the information of segment-level attributes in speech data, such as rising/falling F2, back vowel/front vowel, vowel/fricative, and closure/non-closure.

In contrast, Figures 4-11 and 4-12 illustrate examples for traversing five different dimensions of latent sequence variable $\boldsymbol{z}_2$, while keeping the latent segment variables fixed. It can be seen that the spectral contour, temporal position, and relative frequency-axis position of formants remain almost intact when traversing these latent sequence variables. The attributes being changed when traversing these latent sequence variables are more related to sequence-level attributes, such as harmonic patterns (F0), volume, and offsets of formant frequencies. The results again demonstrate the ability of our proposed FHVAE to not only learn disentangled representations, but also to enable interpretation of the information captured by different sets of latent variables.

4.5 Conclusions

In this chapter, we introduce the factorized hierarchical variational autoencoder, which learns disentangled and interpretable representations for sequence-level and segment-level attributes without any supervision. We verify the disentangling ability by two qualitative methods. We first re-combine latent segment and latent sequence variables to examine if the corresponding attributes are exhibited in the generated segment, Next, we also explore the learned latent space by traversing one dimension at a time to discover the associated physical attributes being modeled.

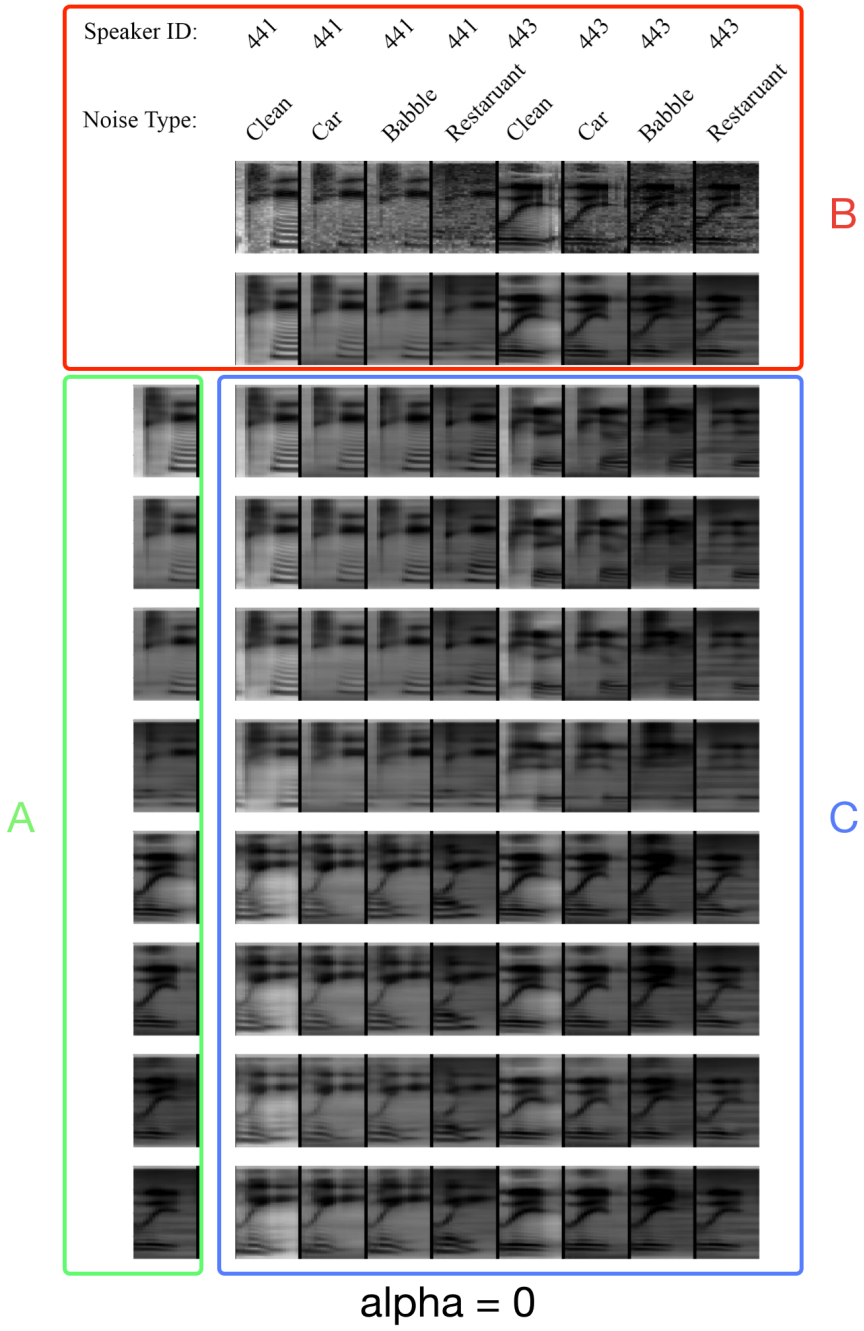


Figure 4-7: Examples generated by varying z_1 and z_2 of an FHVAE model trained with $\alpha = 0$ on Aurora-4 dataset. The green block 'A' and the red block 'B' contain the same eight examples from the test set. In block 'B,' original examples are shown in the first row and the corresponding reconstructed examples are shown in the second row. The entry on the i-th row and the j-th column in the blue block 'C' is the reconstructed example using the latent segment variable $\boldsymbol{z}_1$ of the i-th row from the block 'A' and the latent sequence variable $\boldsymbol{z}_2$ of the j-th column from the block 'B.'

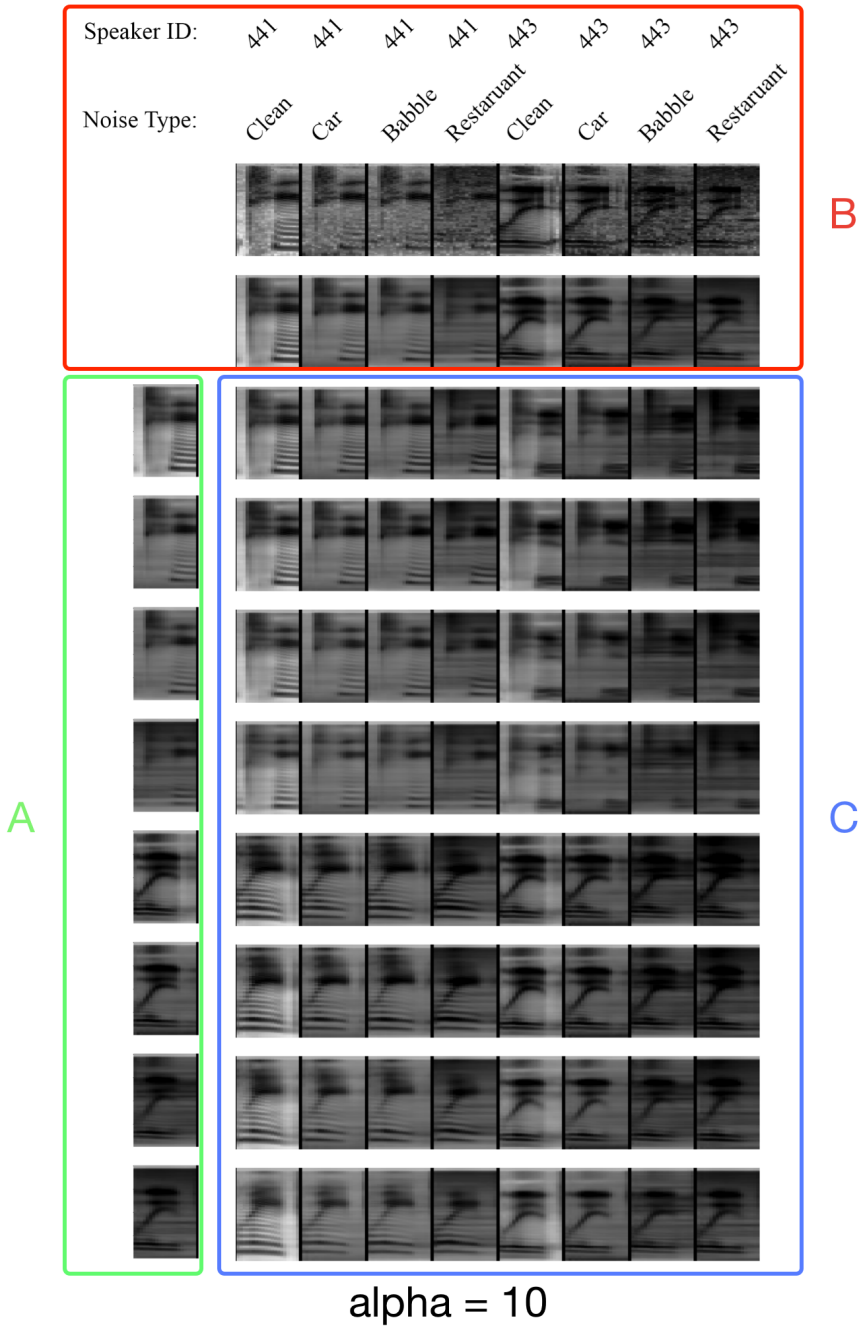


Figure 4-8: Examples generated by varying z_1 and z_2 of an FHVAE model trained with $\alpha = 10$ on Aurora-4 dataset. The green block 'A' and the red block 'B' contain the same eight examples from the test set. In block 'B,' original examples are shown in the first row and the corresponding reconstructed examples are shown in the second row. The entry on the i-th row and the j-th column in the blue block 'C' is the reconstructed example using the latent segment variable $\boldsymbol{z}_1$ of the i-th row from the block 'A' and the latent sequence variable $\boldsymbol{z}_2$ of the j-th column from the block 'B.'

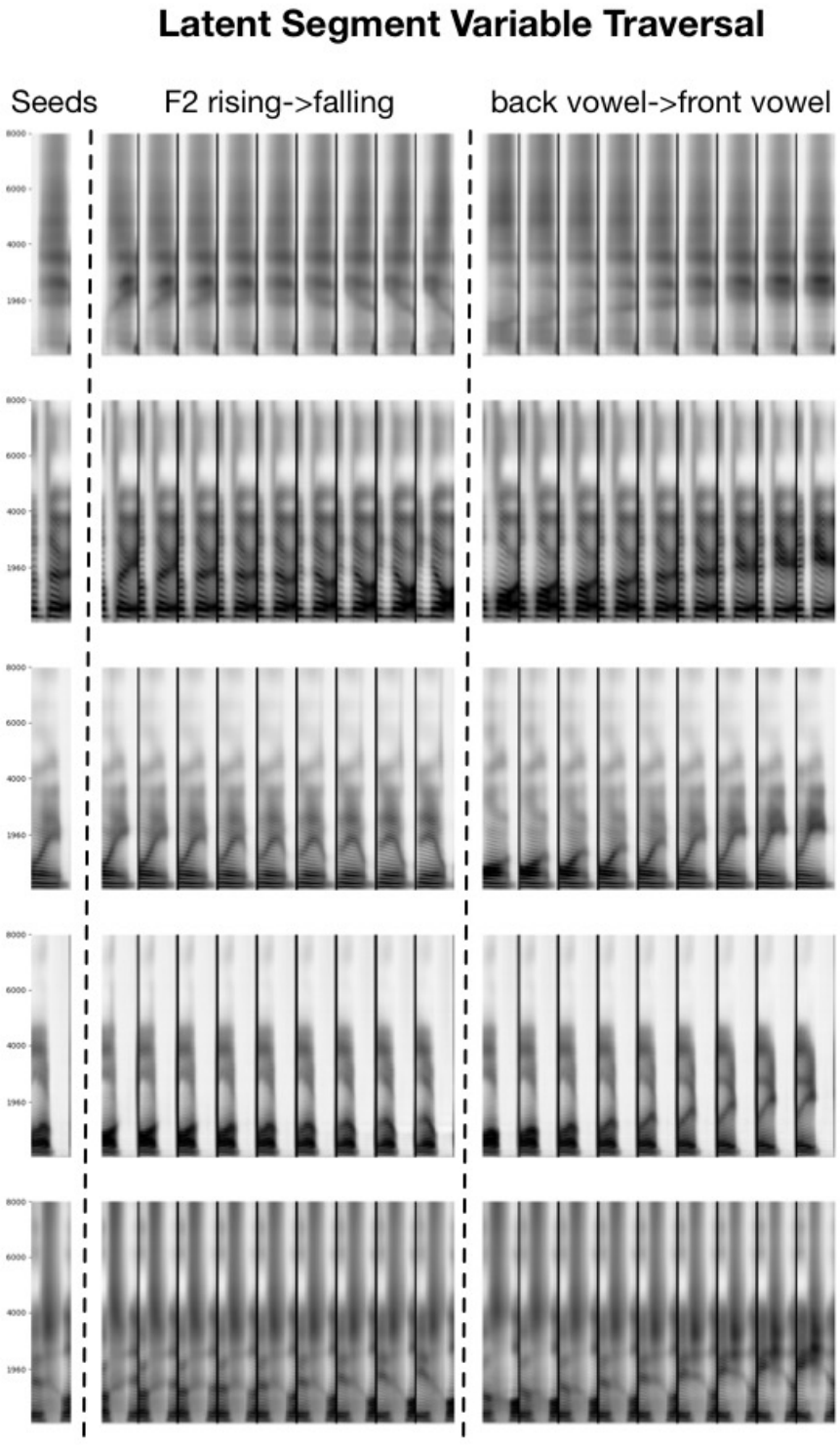


Figure 4-9: Traversing two different dimensions in the space of latent **segment** variables with five seed segments from the TIMIT test set using an FHVAE model trained on TIMIT with $\alpha = 0$.

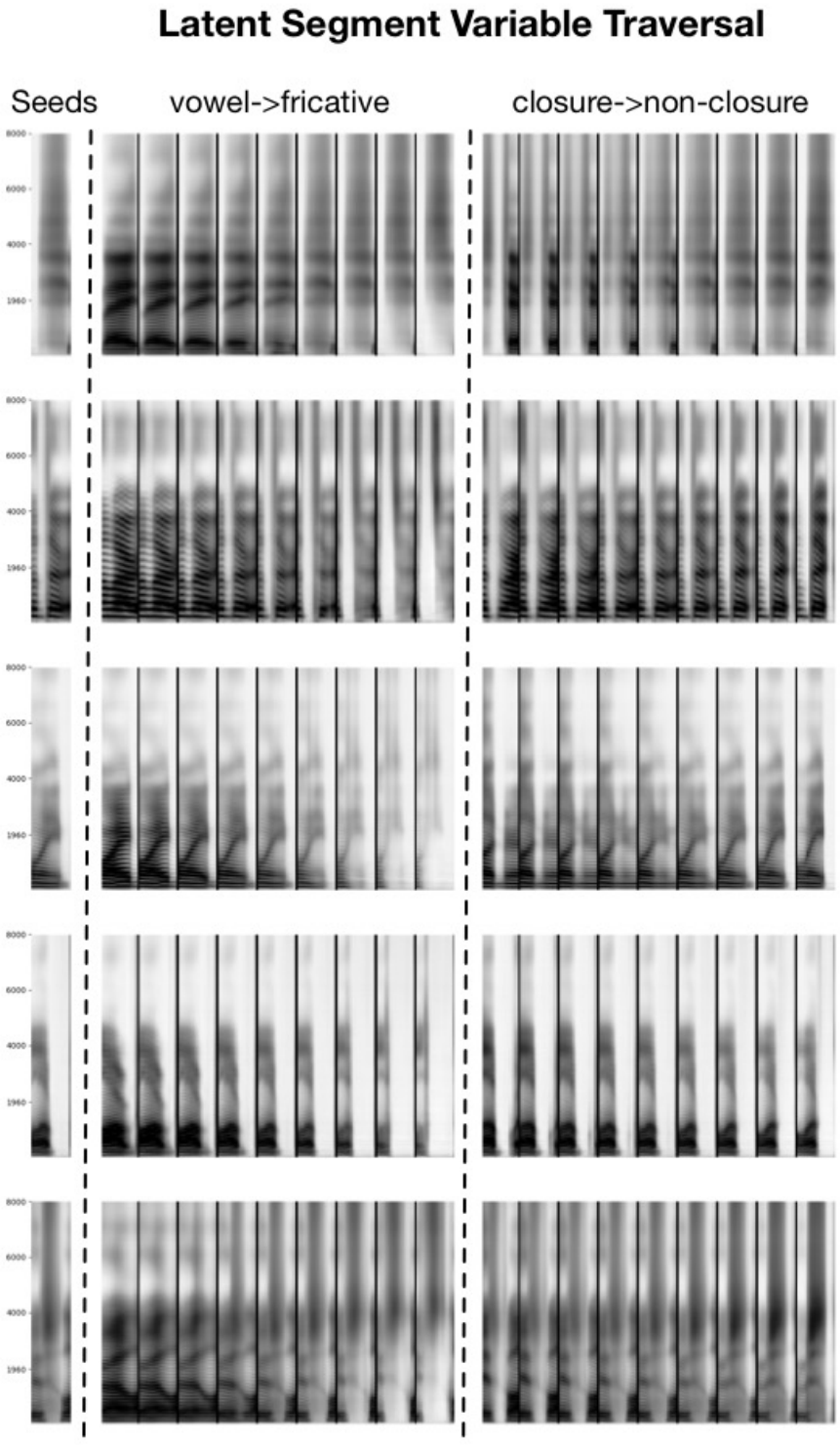


Figure 4-10: Traversing another two different dimensions in the space of latent **segment** variables with five seed segments from the TIMIT test set using an FHVAE model trained on TIMIT with $\alpha = 0$.

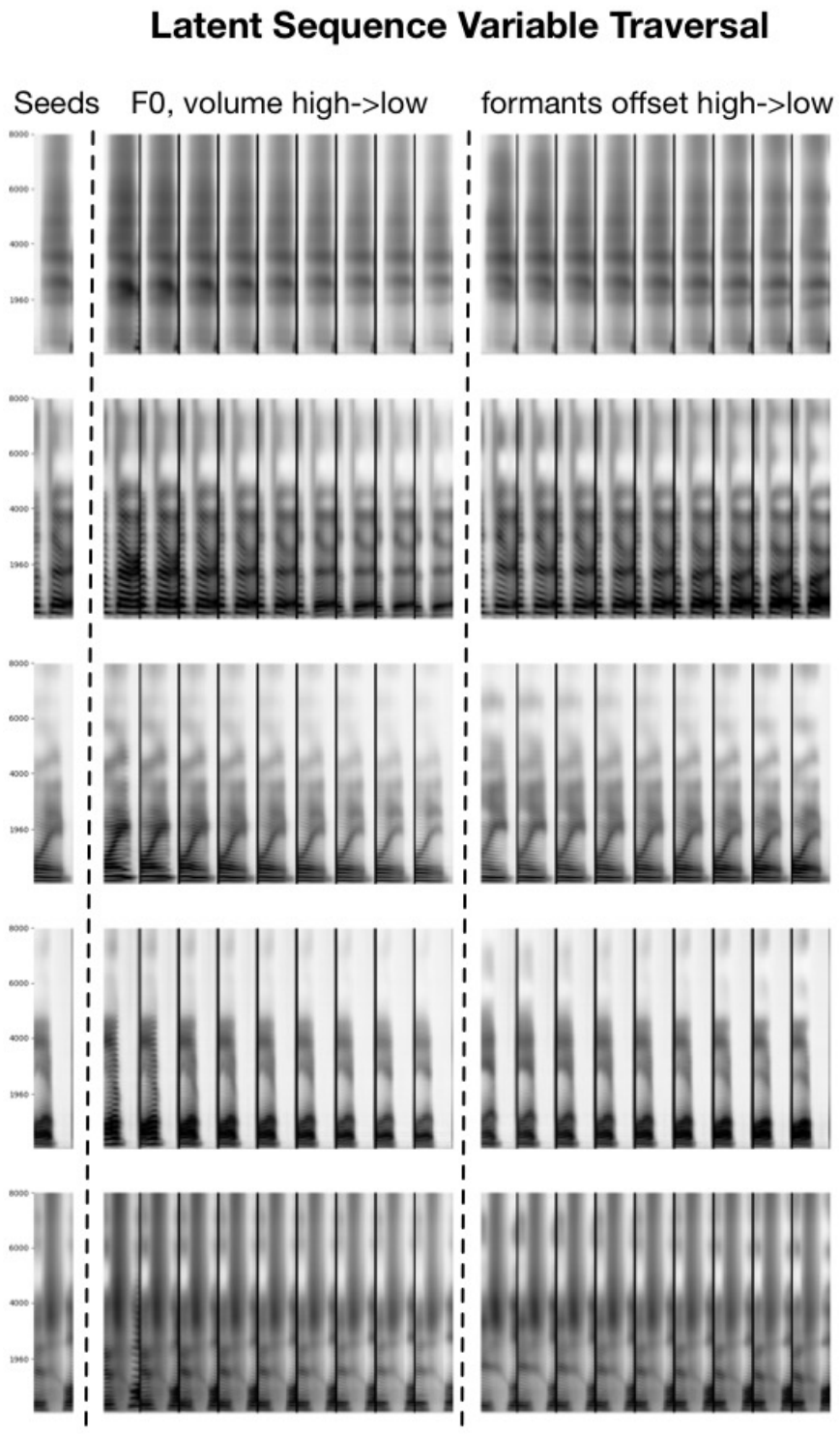


Figure 4-11: Traversing two different dimensions in the space of latent **sequence** variables $\boldsymbol{z}_2$ with five seed segments from the TIMIT test set using an FHVAE model trained on TIMIT with $\alpha = 0$.

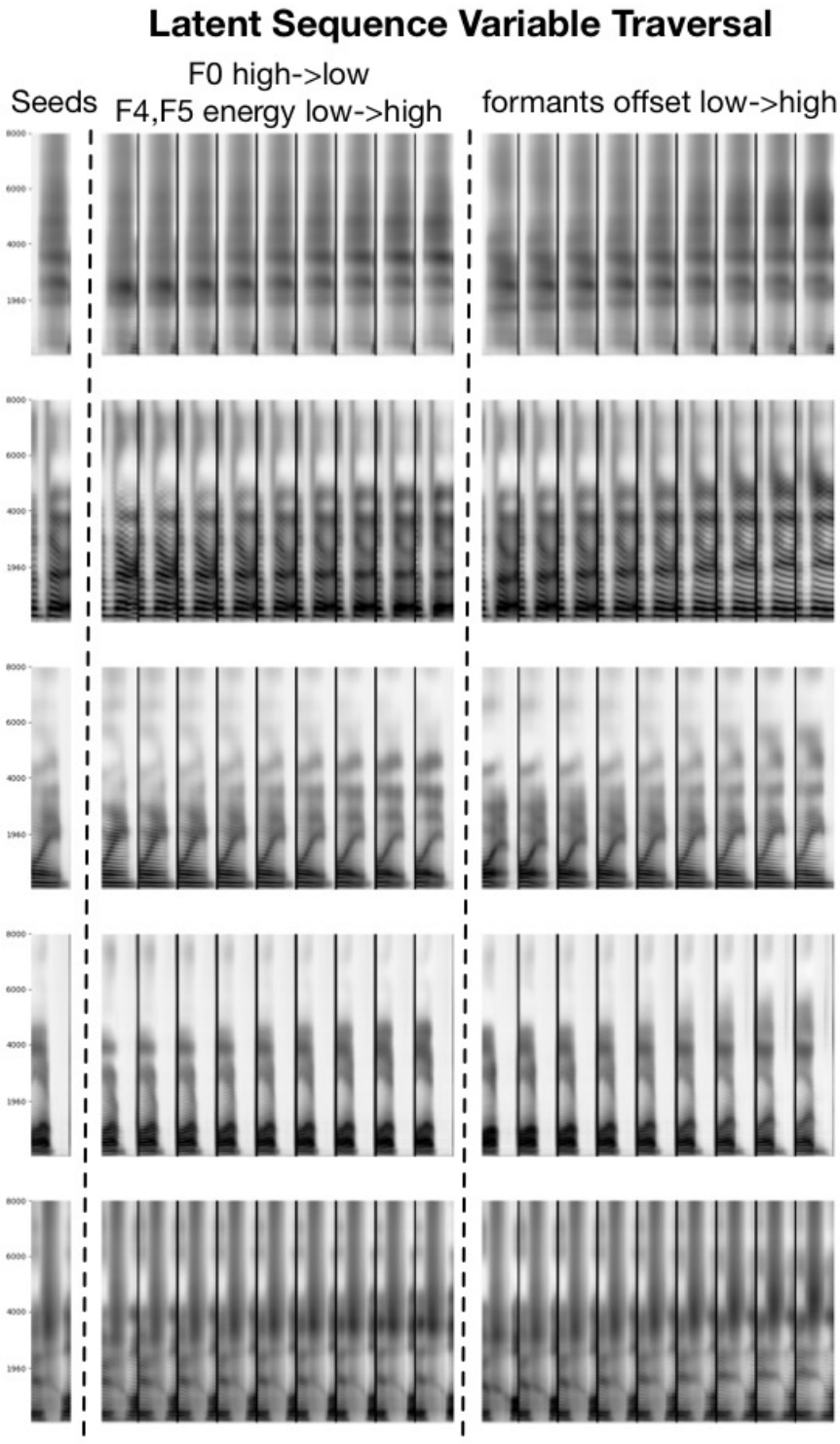


Figure 4-12: Traversing another two different dimensions in the space of latent **sequence** variables $\boldsymbol{z}_2$ with five seed segments from the TIMIT test set using an FHVAE model trained on TIMIT with $\alpha = 0$.

Chapter 5

Applications of Disentangled and Interpretable Representations

Since the learned latent representations are disentangled and interpretable, depending on the application, we can utilize different latent variables accordingly as a new feature representation. In addition, by manipulating different sets of latent variables, we can also synthesize new speech segments with desired attributes. In this chapter, we investigate applications of disentangled and interpretable representations to speech transformation, speaker verification, and robust automatic speech recognition. The results of the first three sections in this chapter were published in Hsu et al. (2017b), and the last section was published in Hsu and Glass (2018a).

5.1 Speech Transformation

In Section 4.4.1, we re-recombined latent segment variable $\boldsymbol{z}_1$ and latent sequence variable $\boldsymbol{z}_2$ from different segments, in order to generate a new speech segment that exhibits certain attributes from each of the segments. In addition to transforming a single segment, one may also be interested in transforming a target utterance $\boldsymbol{X}_{tar} = \{\boldsymbol{x}_{tar}^{(n)}\}_{n=1}^{N_{tar}}$ to be of a different speaker or a different noise condition from a reference utterance $\boldsymbol{X}_{ref} = \{\boldsymbol{x}_{ref}^{(n)}\}_{n=1}^{N_{ref}}$. Mathematically, it means mapping the distribution of the latent sequence variable from that of $\boldsymbol{X}_{tar}$ to that of $\boldsymbol{X}_{ref}$. Since the prior

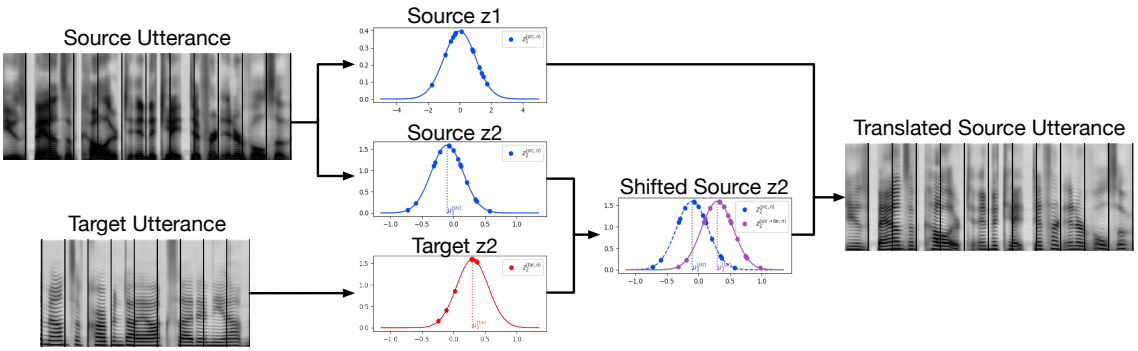


Figure 5-1: Graphical illustration of speech transformation by manipulating latent sequence variable of the target sequence.

distributions of $\boldsymbol{z}_2$ for both utterances are both assumed Gaussian with the same covariance matrices, centered at their own s-vectors $\boldsymbol{\mu}_{2,ref}$ and $\boldsymbol{\mu}_{2,tar}$, a simple solution for mapping the two distributions is to shift each latent sequence variable in the target utterance by the s-vector difference:

$$\Delta\boldsymbol{\mu}_2 = \boldsymbol{\mu}_{2,ref} - \boldsymbol{\mu}_{2,tar}. \tag{5.1}$$

Therefore, we first transform a target utterance given a reference utterance by shifting the $\boldsymbol{z}_2$ of each segment from the target utterance by $\Delta\boldsymbol{\mu}_2$ as follows:

$$\tilde{\boldsymbol{z}}_2 = \boldsymbol{z}_2 + \Delta\boldsymbol{\mu}_2. \tag{5.2}$$

Then, we decode each segment using the unmodified $\boldsymbol{z}_1$ and the modified $\tilde{\boldsymbol{z}}_2$, and concatenate those segments to form a sequence. This procedure is illustrated in Figure 5-1.

5.1.1 Denoising

Figures 5-2 and 5-3 show two examples of speech transformation for removing background noise. In these examples, both the target and the reference utterances are from the same speaker, but are recorded in different noise conditions. While the reference utterances are recorded in noiseless environments, the target utterances are

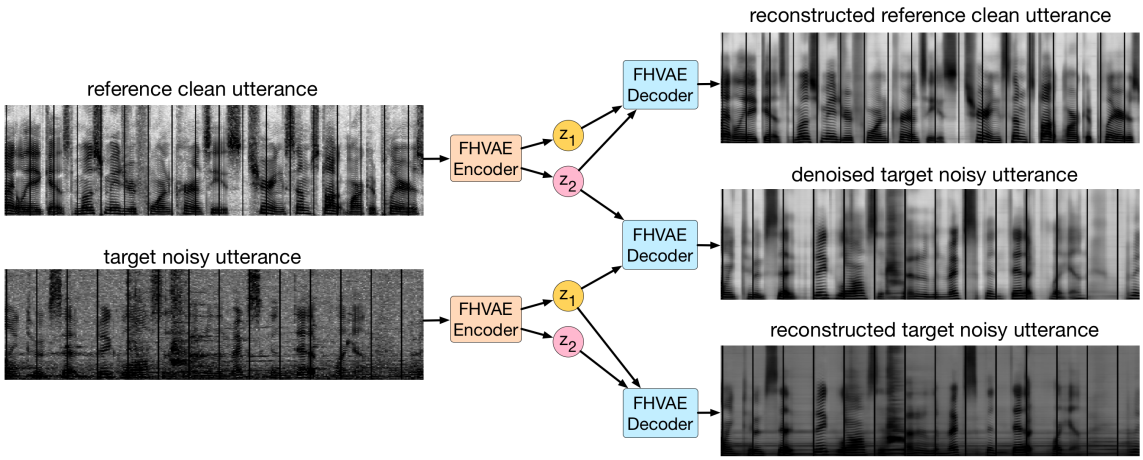


Figure 5-2: FHVAE ($\alpha = 0$) decoding results of three combinations of *latent segment variables* z_1 and *latent sequence variables* z_2 from two utterances in Aurora-4: a clean one (top-left) and a noisy one (bottom-left). FHVAEs learn to encode local attributes, such as linguistic content, into z_1, and encode global attributes, such as noise level, into z_2. Therefore, by replacing z_2 of a noisy utterance with z_2 of a clean utterance, an FHVAE decodes a denoised utterance (middle-right) that preserves the linguistic content. Reconstruction results of the clean and noisy utterances are also shown on the right. Audio samples are available at `https://youtu.be/naJZITvCfI4`.

corrupted with restaurant noise in the first example and with car noise in the second example. Note that the textual content is different between the reference utterance and the target utterance. It can be observed that after replacing the s-vector with that of a clean utterance, the noise in the target utterances is removed, without changing the speaker identity or the linguistic content. However, there are still certain limits regarding noise removal using the FHVAE model. Since the s-vector only captures the generating factors that are consistent within a sequence, noises that are non-stationary cannot be removed using the proposed speech transformation method. For example, in the first example, which contains restaurant noise, we can hear that the sound of silverware hitting each other is not removed.

5.1.2 Voice Conversion

Next we show two more examples in Figures 5-4, and 5-5. In these examples, speakers of the reference and the source utterances are of a different gender. It can be seen

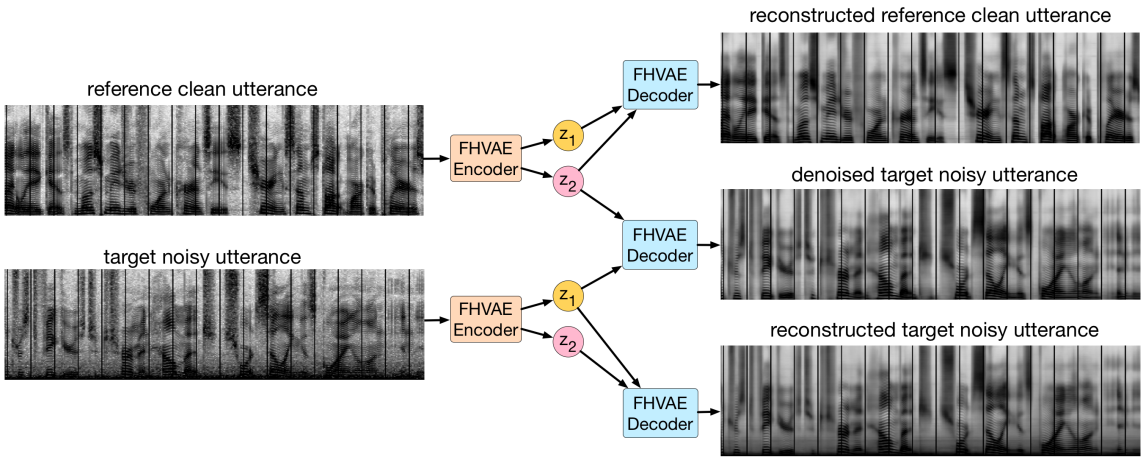


Figure 5-3: FHVAE ($\alpha = 0$) decoding results of three combinations of *latent segment variables* z_1 and *latent sequence variables* z_2 from one clean utterance (top-left) and one utterance with car noise (bottom-left) in Aurora-4. By replacing z_2 of a noisy utterance with z_2 of a clean utterance, an FHVAE decodes a denoised utterance (middle-right) that preserves the linguistic content. Audio samples are available at `https://youtu.be/pOP2DVZWRjM`.

that the harmonic patterns change after the transformation, which resemble those of the reference utterance. By shifting the latent segment variable distribution of the target utterance, we can also modify the speaker identity accordingly, and achieve voice conversion with the FHVAE model.

5.2 Speaker Verification

We next present experiments on a speaker verification task on the TIMIT corpus to evaluate how well the estimated $\boldsymbol{\mu}_2$ encodes speaker-level information.[1] As a sanity check, we modify Eq. 4.13 to estimate an alternative s-vector based on latent segment variables $\boldsymbol{z}_1$ as follows: $\boldsymbol{\mu}_1 = \sum_{n=1}^{N} g_{\mu_{\boldsymbol{z}_1}}(\boldsymbol{x}^{(n)})/(N + \sigma^2_{\boldsymbol{z}_1})$. We use the i-vector method (Dehak et al., 2011) as the baseline, which is the representation used in most state-of-the-art speaker verification systems. They are in a low dimensional subspace of

[1]TIMIT is not a standard corpus for speaker verification, but it is a good corpus to show the utterance-level attribute we learned via this task, because the main attribute that is consistent within an utterance is speaker identity, while in Aurora-4 both speaker identity and the background noise are consistent within an utterance.

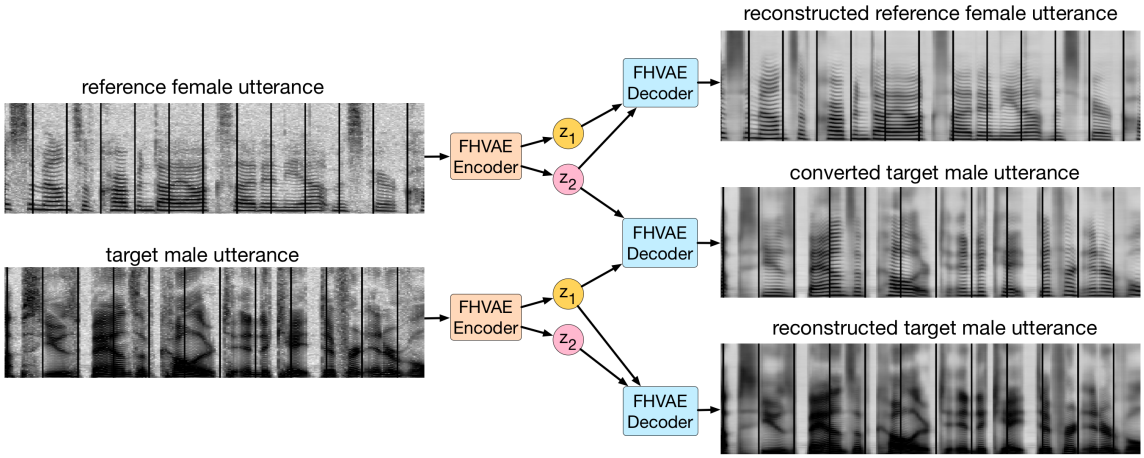


Figure 5-4: FHVAE ($\alpha = 0$) decoding results of three combinations of *latent segment variables* z_1 and *latent sequence variables* z_2 from one male-speaker utterance (top-left) and one female-speaker utterance (bottom-left) in Aurora-4. By replacing z_2 of a male-speaker utterance with z_2 of a female-speaker utterance, an FHVAE decodes a voice-converted utterance (middle-right) that preserves the linguistic content. Audio samples are available at `https://youtu.be/VMX3IZYWYdg`.

the Gaussian mixture model (GMM) mean supervector space, where the GMM is the universal background model (UBM) that models the generative process of speech. I-vectors, $\boldsymbol{\mu}_1$, and $\boldsymbol{\mu}_2$ can all be extracted without supervision; when speaker labels are available during training, techniques such as linear discriminative analysis (LDA) can be applied to further improve the linear separability of the representation.

For all experiments, we use the fast scoring approach in Dehak et al. (2009) that uses cosine similarity as the similarity metric. Verification performance is reported in terms of equal error rate (EER), where the false rejection rate equals the false acceptance rate. For our baseline system, we use the i-vectors (Dehak et al., 2011) provided by Kaldi (Povey et al., 2011), which are extracted using Mel-frequency cepstral coefficients (MFCCs), plus delta and delta-delta after voice activity detection (VAD). A full-covariance gender-independent UBM with 2,048 mixtures was trained on the training set and the i-vector dimensionality is tuned on the development set. The verification pairs were created from the test set as target/non-target. There are in total 24 speakers and 18,336 pairs for testing. For all the Seq2Seq-FHVAE model, $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ have the same dimension.

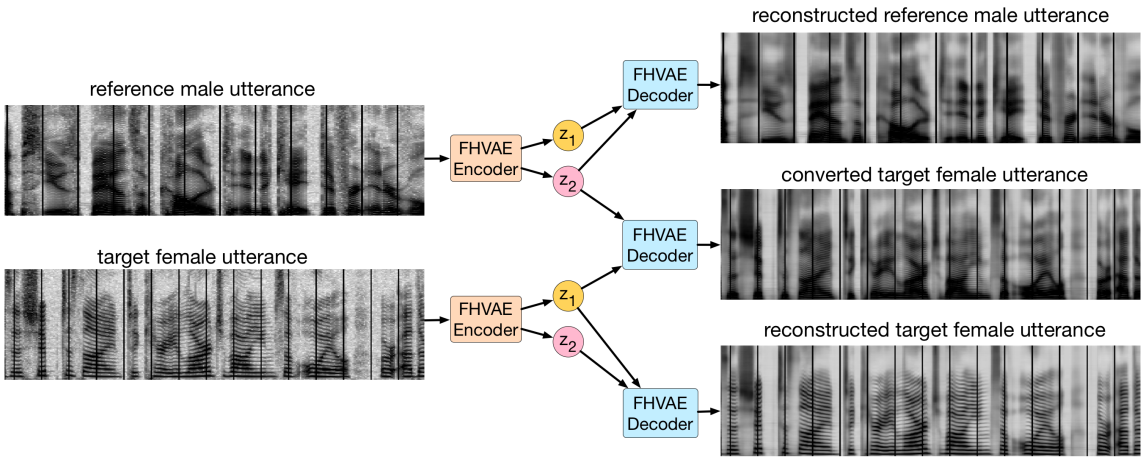


Figure 5-5: FHVAE ($\alpha = 0$) decoding results of three combinations of *latent segment variables* z_1 and *latent sequence variables* z_2 from one female-speaker utterance (top-left) and one male-speaker utterance (bottom-left) in Aurora-4. By replacing z_2 of a female-speaker utterance with z_2 of a male-speaker utterance, an FHVAE decodes a voice-converted utterance (middle-right) that preserves the linguistic content. Audio samples are available at `https://youtu.be/Rurj2ByNRs8`.

We compare different dimensions for both features as well as different α's in Eq. 4.11 for training FHVAE models. The results in Table 5.1 show that the 16 dimensional s-vectors $\boldsymbol{\mu}_2$ outperform i-vector baselines in both unsupervised (Raw) and supervised (LDA) settings for all α's as shown in the fourth column; the more discriminatively the FHVAE model is trained (i.e., with larger α), the better speaker verification results it achieves. Moreover, with the appropriately chosen dimension, a 32 dimensional $\boldsymbol{\mu}_2$ reaches an even lower EER at 1.34%. On the other hand, the negative results of using $\boldsymbol{\mu}_1$ also validate the success in disentangling utterance and segment level attributes.

5.3 Extracting Domain Invariant Features

Speaker adaptation and robust speech recognition in automatic speech recognition (ASR) can often be seen as domain adaptation problems, where available labeled data is limited and hence the data distributions during training and testing are mismatched. One approach to reduce the severity of this issue is to extract speaker/channel

Table 5.1: Comparison of speaker verification equal error rate (EER) on the TIMIT test set

Features	Dimension	α	EER (%)		
			Raw	LDA (12 dim)	LDA (24 dim)
i-vector	16	-	11.90	7.29	-
	48	-	10.12	6.25	5.95
	100	-	9.52	6.10	5.50
	200	-	9.82	6.54	6.10
$\boldsymbol{\mu}_2$	16	0	5.06	4.02	-
	16	10^{-1}	4.91	4.61	-
	16	10^{0}	3.87	3.86	-
	16	10^{1}	**2.38**	**2.08**	-
	32	10^{1}	**2.38**	**2.08**	**1.34**
$\boldsymbol{\mu}_1$	16	10^{0}	22.77	15.62	-
	16	10^{1}	27.68	22.17	-
	32	10^{1}	22.47	16.82	17.26

invariant features for the tasks.

As demonstrated in Section 5.2, the s-vector contains information about domains. Here we evaluate whether the latent segment variables contain domain invariant linguistic information by evaluating on an ASR task:

1. train our proposed Seq2Seq-FHVAE using FBank feature on a set that covers different domains.

2. train an LSTM acoustic model (Graves et al., 2013; Sak et al., 2014; Zhang et al., 2016) on the set that only covers partial domains using mean and log variance of the latent segment variable $\boldsymbol{z}_1$ extracted from the trained Seq2Seq-FHVAE.

3. test the ASR system on all domains.

As a baseline, we also train the same ASR models, but use the FBank features alone.

Here we detail the pipeline for building an ASR system. The Gaussian mixture model-hidden Markov model (GMM-HMM) systems are built first to generate the senone (tied triphone HMM state) alignments for the later neural network acoustic model training, which replaces the GMM acoustic model. In all tasks, the GMM-HMM system is built with Kaldi (Povey et al., 2011) using standard recipes. We

use the LSTM Graves et al. (2013) for the acoustic model in our hybrid DNN-HMM system, which are implemented using the CNTK Yu et al. (2014) toolkit. Our training recipe follows Zhang et al. (2016). The baseline uses 80-dimensional FBank features as input. The model has 3 LSTM-projection layers (Sak et al., 2014), where each layer has 1024 cells and the output is projected to a 512 dimensional space. The truncated BPTT is used to train the LSTM that unrolls 20 frames; 40 utterances are processed in parallel to form a mini-batch. For the Seq2Seq-FHVAE model, we use the same configuration as the one that achieved the best result on the speaker verification task: both $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are 32 dimensional, and the weight $\alpha = 10$ for discriminative training. For the VAE model, the dimension of the latent variable $\boldsymbol{z}$ is 64, and the number of hidden units of the LSTM encoder is 512. We doubled both the latent variable dimension and the number of hidden units for the encoder compared to the FHVAE model because the VAE model only has one set of latent variables and one encoder. Therefore, both the FHVAE and VAE models would have a comparable number of parameters as well as latent space dimensionality.

5.3.1 Robustness to Speaker Variation

For TIMIT we assume that male and female speakers constitute different domains, and show the results in Table 5.2. The first row of results shows that the ASR model trained on all domains (speakers) using FBank features as the upper bound. When trained on only male speakers, the phone error rate (PER) on female speakers increases by 16.1% for FBank features; however, for $\boldsymbol{z}_1$, despite the slight degradation on male speakers, the PER on the unseen domain, which are female speakers, improves by 6.6% compared to FBank features.

Table 5.2: TIMIT test phone error rate of acoustic models trained on different features and sets

Train Set and Configuration			Test PER (%) by Set		
ASR	FHVAE	Features	Male	Female	All
Train All	-	FBank	20.1	16.7	19.1
Train Male	-	FBank	**21.0**	32.8	25.2
	Train All, $\alpha = 10$	$\boldsymbol{z}_1$	22.0	**26.2**	**23.5**

5.3.2 Robustness to Noise and Channel Variation

On Aurora-4, four domains are considered: clean, noisy, channel, and noisy+channel (NC for short). We train the FHVAE on the development set for two purposes: (1) the FHVAE can be considered a general feature extractor, which can be trained on an arbitrary collection of data that does not necessarily include the data for subsequent applications. (2) the dev set of Aurora-4 contains the domain label for each utterance so it is possible to control which domain has been observed by the FHVAE.

Table 5.3: Aurora-4 test word error rate of acoustic models trained on different features and sets

Train Set and Configuration			Test WER (%) by Set				
ASR	{FH-,β-}VAE	Features	Clean	Noisy	Channel	NC	All
Train All	-	FBank	3.60	7.06	8.24	18.49	11.80
Train Clean	-	FBank	**3.47**	50.97	36.99	71.80	55.51
	Dev, $\beta = 1$	$\boldsymbol{z}$ (β-VAE)	4.95	23.54	31.12	46.21	32.47
	Dev, $\beta = 2$	$\boldsymbol{z}$ (β-VAE)	3.57	27.24	30.56	48.17	34.75
	Dev, $\beta = 4$	$\boldsymbol{z}$ (β-VAE)	3.89	24.40	29.80	47.87	33.38
	Dev, $\beta = 8$	$\boldsymbol{z}$ (β-VAE)	5.32	34.84	36.13	58.02	42.76
	Dev, $\alpha = 10$	$\boldsymbol{z}_1$ (FHVAE)	5.01	**16.42**	**20.29**	**36.33**	**24.41**
	Dev, $\alpha = 10$	$\boldsymbol{z}_2$ (FHVAE)	41.08	68.73	61.89	86.36	72.53
	Dev\NC, $\alpha = 10$	$\boldsymbol{z}_1$ (FHVAE)	5.25	16.52	19.30	40.59	26.23

Table 5.3 shows the word error rate (WER) results on Aurora-4, from which we can observe that the FBank representation suffers from severe domain mismatch problems; specifically, the WER increases by 53.3% when noise is presented in mismatched microphone recordings (NC). In contrast, when the FHVAE is trained on data from all domains, using the latent segment variables as features reduces WER from 16% to 35% compared to baseline on mismatched domains, with less than 2% WER degradation on the matched domain.

In addition, β-VAEs (Higgins et al., 2016) are trained on the same data as the FHVAE to serve as the baseline feature extractor, from which we extract the latent variables $\boldsymbol{z}$ as the ASR feature and show the result in the third to the sixth rows. The β-VAE features outperform FBank in all mismatched domains, but are inferior to the latent segment variable $\boldsymbol{z}_1$ from the FHVAE in those domains.

The results demonstrate the importance of learning not only disentangled, but

also interpretable representations, which can be achieved by our proposed FHVAE models. As a sanity check, we replace $\boldsymbol{z}_1$ with $\boldsymbol{z}_2$, the latent sequence variable and train an ASR, which results in terrible WER performance as shown in the eighth row as expected.

Finally, we train another FHVAE on all domains excluding the combinatory NC domain, and shows the results in the last row in Table 5.3. It can be observed that the latent segment variable still outperforms the baseline feature with 30% lower WER on noise and channel combined data, even though the FHAVE has only seen noise and channel variation independently.

5.4 Study of FHVAE Architecture for ASR Feature Extraction

In this section, we extend the previous section, and study how the model architecture, training objective, and generative model assumption of factorized hierarchical variational autoencoders can affect the robustness of extracted ASR features. The same architecture as well as the same training procedure of the acoustic model are used across different experiments, in order to isolate other plausible factors from affecting the comparison of robustness between different features extracted from FHVAEs. In addition, except for global mean and variance normalization, we omit all the feature preprocessing steps, such as per-utterance cepstral mean variance normalization (CMVN) (Liu et al., 1993).

Since the noise in the Aurora-4 dataset is added artificially, we would also like to verify the effectiveness of the proposed feature on a non-artificial noisy dataset. The CHiME-4 (Vincent et al., 2016) dataset contains real distant-talking recordings in noisy environments. We use the original 7,138 clean utterances and the 1,600 single channel real noisy utterances in the training partition to train the VAE and FHVAE models. The ASR system is trained on the original clean training set and evaluated on the CHiME-4 development set.

Table 5.4: Aurora-4 test_eval92 set word error rate of acoustic models trained on different features.

Setting						WER (%)	WER (%) by Condition			
Exp. Index	Feature	#Layers	#Units	α	Seq. Label	Avg.	A	B	C	D
1	FBank	-	-	-	-	65.64	3.21	61.61	51.78	82.39
	$\boldsymbol{z}$	1/1	256/256	-	-	44.79	4.22	38.16	36.11	59.63
	$\boldsymbol{z}$	1/1	512/256	-	-	40.31	4.35	33.83	34.43	53.77
	$\boldsymbol{z}_1$	1/1	256/256	10	uttid	**26.58**	4.54	19.28	20.85	38.50
2	$\boldsymbol{z}_1$	1/1	256/256	10	uttid	26.58	4.54	19.28	20.85	38.50
	$\boldsymbol{z}_1$	2/2	256/256	10	uttid	25.54	4.11	16.90	20.62	38.58
	$\boldsymbol{z}_1$	3/3	256/256	10	uttid	**24.30**	4.91	15.44	22.83	36.63
3	$\boldsymbol{z}_1$	1/1	128/128	10	uttid	34.66	5.06	26.70	25.39	49.09
	$\boldsymbol{z}_1$	1/1	256/256	10	uttid	**26.58**	4.54	19.28	20.85	38.50
	$\boldsymbol{z}_1$	1/1	512/512	10	uttid	26.97	5.32	18.18	23.13	40.01
4	$\boldsymbol{z}_1$	1/1	256/256	0	uttid	33.30	4.86	25.67	25.46	46.97
	$\boldsymbol{z}_1$	1/1	256/256	5	uttid	30.55	4.63	22.66	23.33	43.96
	$\boldsymbol{z}_1$	1/1	256/256	10	uttid	**26.58**	4.54	19.28	20.85	38.50
	$\boldsymbol{z}_1$	1/1	256/256	15	uttid	29.92	5.01	20.82	24.79	44.03
	$\boldsymbol{z}_1$	1/1	256/256	20	uttid	32.64	5.57	25.48	24.53	45.66
5	$\boldsymbol{z}_1$	1/1	256/256	10	uttid	**26.58**	4.54	19.28	20.85	38.50
	$\boldsymbol{z}_1$	1/1	256/256	10	noise	32.27	4.33	23.89	28.96	45.86
	$\boldsymbol{z}_1$	1/1	256/256	10	speaker	34.95	4.39	27.27	32.22	48.20
6	$\boldsymbol{z}_1$	1/1	256/256	10	uttid	**26.58**	4.54	19.28	20.85	38.50
	$\boldsymbol{z}_1$-$\boldsymbol{\mu}_2$	1/1	256/256	10	uttid	43.61	5.08	42.47	27.55	53.85

Tables 5.4 and 5.5 summarize the results on Aurora-4 and CHiME-4 respectively. For both tables, different experiments are separated by double horizontal lines and indexed by the *Exp. Index* on the first column. The second column, *Feature*, refers to the frame representations used for training ASR models. The third to the sixth column explains the model configuration and the discriminative training weight for VAE or FHVAE models. We separate the encoder and decoder parameters by "/" in the third and the fourth column. Averaged and by-condition word error rate (WER) are shown in the rest of the columns.

5.4.1 Baseline

We start with establishing Aurora-4 baseline results trained on different types of feature representations, including (1) FBank, (2) latent variable, $\boldsymbol{z}$, extracted from the VAE, and (3) latent segment variable, $\boldsymbol{z}_1$, extracted from the FHVAE. Because each FHVAE model has two encoders, to have a fair comparison between VAE and FHVAE models, we also consider a VAE model with 512 hidden units at each encoder

layer. The results are shown in Table 5.4 *Exp. Index 1.* As mentioned, condition A is the matched domain, while conditions B, C, and D are all mismatched domains.

FBank degrades significantly in the mismatched conditions, producing between 49% to 79% absolute WER increase. On the other hand, both VAE and FHVAE models improve the performance in the mismatched domains by a large margin, with only a slight degradation in the matched domain. In particular, the features learned by the FHVAE consistently outperform the VAE features in all mismatched conditions by 14% absolute WER reduction.

We believe that this experiment verifies that FHVAEs can successfully retain domain invariant linguistic features in $\boldsymbol{z}_1$, while encoding domain related information into $\boldsymbol{z}_2$. In contrast, as the results suggest, VAEs encode all the information into a single set of latent variables, $\boldsymbol{z}$, which still contain domain related information that can hurt ASR performance on the mismatched domains.

5.4.2 Comparing Model Architectures

We next explore the optimal FHVAE architectures for extracting domain invariant features. In particular, we study the effect of the number of hidden units at each layer and the number of layers. Results of each variant are listed in Table 5.4 *Exp. Index 2* and *Exp. Index 3* respectively. Regarding the averaged WER, the model with 256 hidden units at each layer and in total three layers achieves the lowest WER (24.30%). Interestingly, if we break down the WER by condition, it can be observed that increasing the FHVAE model capacity (i.e. increasing number of layers or hidden units) helps reducing the WER in the noisy condition (B), but deteriorates in the channel-mismatching condition (C) above 256 hidden units and 2 layers.

5.4.3 Effect of FHVAE Discriminative Training

Speaker verification experiments in Hsu et al. (2017b) suggest that discriminative training facilitates factorizing segment-level attributes and sequence-level attributes into two sets of latent variables. Here we study the effect of discriminative training on

learning robust ASR features, and show the results in Table 5.4 *Exp. Index 4*. When $\alpha = 0$, the model is not trained with the discriminative object. While increasing the discriminative weight from 0 to 10, we observe consistent improvement in all 4 conditions due to better factorization of segment and sequence information; however, when further increasing the weight to 20, the performance starts to degrade. This is because the discriminative object can inversely affect the modeling capacity by constraining the expressibility of the latent sequence variables.

5.4.4 Choice of Sequence Label

A core idea of FHVAE is to learn sequence-specific priors to model the generation of sequence-level attributes, which have a smaller amount of variation within a sequence. Suppose we treat each utterance as one sequence, then both speaker and noise information belongs to sequence-level attributes, because they are consistent within an utterance. Alternatively, we consider two FHVAE models that learn speaker-specific priors and noise-specific priors respectively. This can be easily achieved by concatenating sequences of the same speaker label or noise label, and treating it as one sequence used for FHVAE training. We report the results in Table 5.4 *Exp. Index 5*.

It may at first seem surprising that utilizing supervised information in this fashion does not improve performance. We believe that concatenating utterances actually discards some useful information with respect to learning domain invariant features. FHVAEs use latent segment variables to encode attributes that are not consistent within a sequence. By concatenating speaker utterances, noise information is no longer consistent within sequences, and would thus be encoded into latent segment variables; similarly, latent segment variables would not be speaker invariant in the other case.

5.4.5 Use of S-Vector

Lastly, we study the use of s-vectors, $\boldsymbol{\mu}_2$, derived from the FHVAE model, which can be seen as a summarization of sequence-level attributes of an utterance. We apply

the same procedure as i-vector based speaker adaptation (Saon et al., 2013): For each utterance, we first estimate its s-vector, and then concatenate s-vectors with the feature representation of each frame to generate the new feature sequence.

Results are shown in Table 5.4 *Exp. Index 6*, from which we observe a significant degradation of WER that is similar to those of the VAE models. This is reasonable because $\boldsymbol{z}_1$ and $\boldsymbol{\mu}_2$ in combination actually contains similar information as the latent variable $\boldsymbol{z}$ in VAE models, and the degradation is due to the mismatch between the distributions of $\boldsymbol{\mu}_2$ in the training and testing sets.

		Setting				WER (%)		WER (%) by Noise Type			
Exp. Index	ASR Feature	#Layers	#Units	α	Seq. Label	Clean	Noisy	BUS	CAF	PED	STR
	FBank	-	-	-	-	19.37	87.69	95.56	92.05	78.77	84.37
1	$\boldsymbol{z}$	1/1	512/256	-	-	19.47	73.95	70.10	91.45	64.26	69.99
	$\boldsymbol{z}_1$	1/1	256/256	10	uttid	19.57	**67.94**	71.96	79.37	59.32	61.11
	$\boldsymbol{z}_1$	1/1	256/256	10	uttid	19.57	67.94	71.96	79.37	59.32	61.11
2	$\boldsymbol{z}_1$	2/2	256/256	10	uttid	19.73	62.44	71.28	71.86	52.46	54.18
	$\boldsymbol{z}_1$	3/3	256/256	10	uttid	19.52	**60.39**	69.13	66.24	51.22	54.96

Table 5.5: CHiME-4 development set word error rate of acoustic models trained on different features.

5.4.6 Verifying Results on CHiME4

In this section, we repeat the baseline and the layer experiments on the CHiME-4 dataset, in order to verify the effectiveness of the FHVAE and the optimality of the FHVAE architecture on a non-artificial dataset. The results are shown in Table 5.5. From *Exp. Index 1*, we see that the same trend applies to the CHiME-4 dataset, where the latent segment variables from the FHVAE outperform those from the VAE, and both latent variable representations outperform FBank features. For the FHVAE architectures, a 7% absolute WER decrease is achieved by increasing the number of encoder/decoder layers from 1 to 3, which is also consistent with the trends we saw on Aurora-4.

Chapter 6

Scalable Factorized Hierarchical Variational Autoencoder Training

In previous chapters, we introduced a factorized hierarchical variational autoencoder (FHVAE), which is a variational inference-based deep generative model that learns interpretable and disentangled latent representations from sequential data without supervision by modeling a hierarchical generative process. In particular, we demonstrate that an FHVAE trained on speech data learns to encode sequence-level generating factors, such as speaker and channel condition, into one set of latent variables, while encoding segment-level generating factors, such as phonetic content, into another set of latent variables. The ability to disentangle latent factors has been beneficial for a wide range of tasks, including conditional data augmentation (Chapter 3), speaker verification (Section 5.2), domain adaptation (Section 5.3), and voice conversion (Section 5.1).

However, the original FHVAE training algorithm does not easily scale to datasets of over hundreds of thousands of utterances, making it less applicable to real world settings, where an unlabeled dataset of such size is common. This limitation is mainly due to the following issues: (1) the inference model of the sequence-level latent variable, and (2) the design of the discriminative objective. To be more specific, the original training algorithm reduces the complexity of inferring sequence-level latent variables by maintaining a lookup table, whose number of entries equals the number

of training sequences. In addition, the discriminative object, which encourages disentanglement, requires computing a partition function that sums over a function of each entry in that lookup table. The two facts combined lead to significant scalability issues.

In this chapter, we propose a hierarchical sampling algorithm to address these issues. In addition, a new method for qualitatively evaluating disentanglement performance based on a t-Distribution Stochastic Neighbor Embedding (Van Der Maaten, 2014) is also presented. The proposed training algorithm is evaluated on a wide variety of datasets, ranging from 3 to 1,000 hours and involving many different types of generating factors, such as recording conditions and noise types. Experimental results verify that the proposed algorithm is effective on all sizes of datasets and universally achieves desirable disentanglement performance.

6.1 Limitations of the Original FHVAE training

In this section, we briefly review the original FHVAE training algorithm and discuss its scalability issues.

6.1.1 Original FHVAE Training

Table 6.1: Family of distributions adopted for FHVAE generative and inference models.

generative model	
$p(\boldsymbol{\mu}_2)$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$p(\boldsymbol{z}_1)$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$p(\boldsymbol{z}_2\|\boldsymbol{\mu}_2)$	$\mathcal{N}(\boldsymbol{\mu}_2, \sigma^2_{\boldsymbol{z}_2}\boldsymbol{I})$
$p(\boldsymbol{x}\|\boldsymbol{z}_1, \boldsymbol{z}_2)$	$\mathcal{N}(f_{\mu_{\boldsymbol{x}}}(\boldsymbol{z}_1, \boldsymbol{z}_2), diag(f_{\sigma^2_{\boldsymbol{x}}}(\boldsymbol{z}_1, \boldsymbol{z}_2)))$
inference model	
$q(\boldsymbol{\mu}_2\|\boldsymbol{X})$	$\mathcal{N}(\sum_{n=1}^{N} g_{\mu_{\boldsymbol{z}_2}}(\boldsymbol{x}^{(n)})/(N+\sigma^2_{\boldsymbol{z}_2}), \boldsymbol{I})$
$q(\boldsymbol{z}_1\|\boldsymbol{x}, \boldsymbol{z}_2)$	$\mathcal{N}(g_{\mu_{\boldsymbol{z}_1}}(\boldsymbol{x}, \boldsymbol{z}_2), diag(g_{\sigma^2_{\boldsymbol{z}_1}}(\boldsymbol{x}, \boldsymbol{z}_2)))$
$q(\boldsymbol{z}_2\|\boldsymbol{x})$	$\mathcal{N}(g_{\mu_{\boldsymbol{z}_2}}(\boldsymbol{x}), diag(g_{\sigma^2_{\boldsymbol{z}_2}}(\boldsymbol{x})))$

In the variational inference framework, since the marginal likelihood of observed data is intractable, we optimize a lower bound of it instead, named the variational

lower bound. We summarize in Table 6.1 the family of distributions an FHVAE adopts for the generative model and the inference model. All the functions, $f_{\mu_{\boldsymbol{x}}}(\cdot,\cdot)$, $f_{\sigma^2_{\boldsymbol{x}}}(\cdot,\cdot)$, $g_{\mu_{\boldsymbol{z}_1}}(\cdot,\cdot)$, $g_{\sigma^2_{\boldsymbol{z}_1}}(\cdot,\cdot)$, and $g_{\mu_{\boldsymbol{z}_2}}(\cdot)$, $g_{\sigma^2_{\boldsymbol{z}_2}}(\cdot)$, are neural networks that parameterize the mean and variance of Gaussian distributions. We use θ to denote the set of parameters in the generative model, and ϕ to denote the set of parameters in the inference model. We can formulate the variational lower bound of a sequence $\boldsymbol{X}$, $\mathcal{L}(\theta,\phi;\boldsymbol{X})$, based on Table 6.1 as follows:

$$\begin{aligned}\mathcal{L}(\theta,\phi;\boldsymbol{X}) = &\sum_{n=1}^{N}\mathbb{E}_{q(\boldsymbol{z}_1,\boldsymbol{z}_2|\boldsymbol{x}^{(n)})}[\log p(\boldsymbol{x}^{(n)}|\boldsymbol{z}_1,\boldsymbol{z}_2)]\\ &-\sum_{n=1}^{N}\mathbb{E}_{q(\boldsymbol{z}_2|\boldsymbol{x}^{(n)})}[D_{KL}(q(\boldsymbol{z}_1|\boldsymbol{x}^{(n)},\boldsymbol{z}_2)||p(\boldsymbol{z}_1))]\\ &-\sum_{n=1}^{N}\mathbb{E}_{q(\boldsymbol{\mu}_2|\boldsymbol{X})}[D_{KL}(q(\boldsymbol{z}_2|\boldsymbol{x}^{(n)})||p(\boldsymbol{z}_2|\boldsymbol{\mu}_2))]\\ &-D_{KL}(q(\boldsymbol{\mu}_2|\boldsymbol{X})||p(\boldsymbol{\mu}_2)).\end{aligned}$$

However, this lower bound can only be optimized at the sequence level, because inferring $\boldsymbol{\mu}_2$ depends on an entire sequence, and would become infeasible if $\boldsymbol{X}$ is extremely long.

Instead, in the previous chapter we proposed replacing the maximum a posterior (MAP) estimation of $\boldsymbol{\mu}_2$'s posterior mean for training sequences with a lookup table $h_{\mu_{\boldsymbol{\mu}_2}}(i)$, where i indexes training sequences. In other words, the inference model for $\boldsymbol{\mu}_2$ becomes $q(\boldsymbol{\mu}_2|\boldsymbol{X}^{(i)}) = \mathcal{N}(h_{\mu_{\boldsymbol{\mu}_2}}(i), \boldsymbol{I})$. Therefore, the lower bound can be re-written as:

$$\mathcal{L}(\theta,\phi;\boldsymbol{X}^{(i)}) = \sum_{n=1}^{N^{(i)}}\mathcal{L}(\theta,\phi;\boldsymbol{x}^{(i,n)}|h_{\mu_{\boldsymbol{\mu}_2}}(i)) + \log p(h_{\mu_{\boldsymbol{\mu}_2}}(i)) \tag{6.1}$$

$$\begin{aligned}\mathcal{L}(\theta,\phi;\boldsymbol{x}^{(i,n)}|h_{\mu_{\boldsymbol{\mu}_2}}(i)) = &\mathbb{E}_{q(\boldsymbol{z}_1,\boldsymbol{z}_2|\boldsymbol{x}^{(i,n)})}[\log p(\boldsymbol{x}^{(i,n)}|\boldsymbol{z}_1,\boldsymbol{z}_2)]\\ &-\mathbb{E}_{q(\boldsymbol{z}_2|\boldsymbol{x}^{(i,n)})}[D_{KL}(q(\boldsymbol{z}_1|\boldsymbol{x}^{(i,n)},\boldsymbol{z}_2)||p(\boldsymbol{z}_1))]\\ &-D_{KL}(q(\boldsymbol{z}_2|\boldsymbol{x}^{(i,n)})||p(\boldsymbol{z}_2|h_{\mu_{\boldsymbol{\mu}_2}}(i))).\end{aligned} \tag{6.2}$$

We can now sample a batch at the segment level to optimize the following *segment variational lower bound*:

$$\mathcal{L}(\theta, \phi; \boldsymbol{x}^{(i,n)}) = \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(i,n)} | h_{\mu_{\boldsymbol{\mu}_2}}(i)) + \frac{1}{N^{(i)}} \log p(h_{\mu_{\boldsymbol{\mu}_2}}(i)). \tag{6.3}$$

Furthermore, to obtain meaningful disentanglement between $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, it is not desirable to have constant $\boldsymbol{\mu}_2$ for all sequences, which would result in interchangeable $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. To avoid such a condition, the following objective is added to encourage $\boldsymbol{\mu}_2$ to be discriminative between sequences:

$$\log p(i|\bar{\boldsymbol{z}}_2^{(i,n)}) := \log \frac{p(\bar{\boldsymbol{z}}_2^{(i,n)}|\bar{\boldsymbol{\mu}}_2^{(i)})}{\sum_{j=1}^{M} p(\bar{\boldsymbol{z}}_2^{(i,n)}|\bar{\boldsymbol{\mu}}_2^{(j)})}, \tag{6.4}$$

where M is the total number of training sequences, $\bar{\boldsymbol{z}}_2^{(i,n)}$ denotes the posterior mean of $\boldsymbol{z}_2$, $g_{\mu_{\boldsymbol{z}_2}}(\boldsymbol{x}^{(i,n)})$, and $\bar{\boldsymbol{\mu}}_2^{(i)}$ denotes the posterior mean of $\boldsymbol{\mu}_2$, $h_{\mu_{\boldsymbol{\mu}_2}}(i)$. This additional discriminative objective encourages $\boldsymbol{z}_2$ from the i-th sequence to be not only close to $\boldsymbol{\mu}_2$ of the i-th sequence, but also far from $\boldsymbol{\mu}_2$ of other sequences. Combining this discriminative objective and the segment variational lower bound with a weighting parameter, α, the objective function that an FHVAE maximizes then becomes:

$$\mathcal{L}^{dis}(\theta, \phi; \boldsymbol{x}^{(i,n)}) = \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(i,n)}) + \alpha \log p(i|\bar{\boldsymbol{z}}_2^{(i,n)}), \tag{6.5}$$

referred to as the *discriminative segment variational lower bound*.

6.1.2 Scalability Issues

The original FHVAE training addressed the scalability issue with respect to sequence length by decomposing a sequence variational lower bound into a sum of segmental variational lower bounds over segments. However, here we will show that this training objective is not scalable with respect to the number of training sequences.

First of all, the original FHVAE training maintains a lookup table, $h_{\mu_{\boldsymbol{\mu}_2}}(\cdot)$, that stores the posterior mean of $\boldsymbol{\mu}_2$ for each training sequence. The size of this table grows

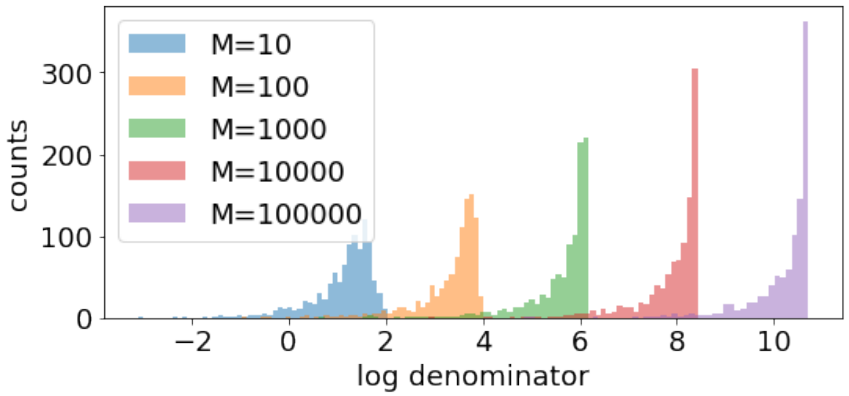


Figure 6-1: Histogram of $\log \sum_{i=1}^{M} p(\bar{\boldsymbol{z}}_2^{(i,n)} | \bar{\boldsymbol{\mu}}_2^{(j)})$ with respect to different $M \in \{10^1, 10^2, 10^3, 10^4, 10^5\}$. Distributions shift by roughly a constant when M increases by 10 times, implying the denominator scales proportionally to M.

proportionally to the number of training sequences. However, this is less problematic, unless the number of our training sequences is on the order of 10^8, which can make the lookup table size grow to 10 GB (suppose $\boldsymbol{z}_2$ are 32-dimensional 32-bit floating point vectors). The main limitation here is the computation of the discriminative objective. The denominator, $\sum_{i=1}^{M} p(\bar{\boldsymbol{z}}_2^{(i,n)} | \bar{\boldsymbol{\mu}}_2^{(j)})$, marginalizes over a function of the posterior mean of $\boldsymbol{\mu}_2$ for all training sequences, which increases the computation time proportionally to the number of sequences. Furthermore, when computing the gradient given a batch of training segments, we need to maintain a tensor of size $(bs, |\theta|)$, where bs is the batch size, and $|\theta|$ is the total number of trainable parameters involved in the computation of the objective function. Since the computation of the discriminative objective involves the entire lookup table, $h_{\mu_{\boldsymbol{\mu}_2}}(\cdot)$, the gradient tensor is of size at least bs times larger than the lookup table. With a batch size of 256, a dataset with 10^5 sequences can exhaust the memory of a single GPU during training.

In addition, from the hyperparameter optimization point of view, it can be observed that the distribution of the denominator, $\sum_{i=1}^{M} p(\bar{\boldsymbol{z}}_2^{(i,n)} | \bar{\boldsymbol{\mu}}_2^{(j)})$, also changes with respect to the number of training sequences. Specifically, the expected value of this terms scales proportionally to M asymptotically, as shown in Figure 6-1. Such behavior is not desirable, because the α parameter that balances the variational lower bound and the discriminative objective would need to be adjusted according to M.

6.2 Training with Hierarchical Sampling

In order to utilize the discriminative objective, while eliminating the memory, computation, and optimization issues induced by a large training set, we need to control the size of the lookup table as well as the denominator summation in the discriminative objective. Both of these can be achieved jointly with a hierarchical sampling algorithm.

Given a dataset of M training sequences, we maintain a lookup table of only K entries, where K is a dataset independent hyperparameter, and optimize an FHVAE model with the following procedure:

(1) A batch of K sequences, $\{\tilde{\boldsymbol{X}}^{(k)}\}_{k=1}^{K}$, is sampled from the entire training set, where $k \in [1, K]$ indexes the sampled training sequences.

(2) The k-th entry of the lookup table, $h_{\mu_{\boldsymbol{\mu}_2}}(k)$, is updated with the MAP estimation of the k-th sequence in the batch, $\sum_{n=1}^{N^{(k)}} g_{\mu_{\boldsymbol{z}_2}}(\tilde{\boldsymbol{x}}^{(k,n)})/(N + \sigma^2_{\boldsymbol{z}_2})$.

(3) NB_{seq} batches of segments are drawn from the K sequences.

(4) Each segment batch is used iteratively to estimate the discriminative segmental variational lower bound for optimizing the parameters of f_*, g_*, and $h_{\mu_{\boldsymbol{\mu}_2}}$ as before. The only difference is that the denominator of the discriminative object now sums over the K sampled training sequences, instead of the entire set of M training sequences.

(5) repeat the above steps until convergence.

We list the pseudo code in Algorithm 1.

We refer to the proposed algorithm as a hierarchical sampling algorithm, because we first sample at the sequence level, and then at the segment level, in order to reduce the effective number of training sequences from M to K for each segment batch. Compared with the proposed algorithm, the original training algorithm can be regarded as using a "flat" sampling algorithm, where we sample segments from the entire pool, so it is therefore necessary to maintain a lookup table of M entries. The

Algorithm 1 Training with Hierarchical Sampling

Input: $\{\boldsymbol{X}^{(i)}\}_{i=1}^{M}$: training set; K: sequence batch size; bs: segment batch size; NB_{seg}: number of segment batches; f_*/g_*: decoders/encoders; $h_{\mu_{\boldsymbol{\mu}_2}}$: lookup table of K entries; Opt: gradient descent-based optimizer

while not converged **do**
  sample a batch of K training sequences, $\{\tilde{\boldsymbol{X}}^{(k)}\}_{k=1}^{K}$
  for $k = 1 \dots K$ **do**
    $h_{\mu_{\boldsymbol{\mu}_2}}(k) \leftarrow \sum_{n=1}^{N^{(k)}} g_{\mu_{\boldsymbol{z}_2}}(\tilde{\boldsymbol{x}}^{(k,n)})/(N + \sigma_{\boldsymbol{z}_2}^2)$
  end for
  for $1 \dots NB_{seg}$ **do**
    sample segments $\{\tilde{\boldsymbol{x}}^{(k_b,n_b)}\}_{b=1}^{bs}$ from $\{\tilde{\boldsymbol{X}}^{(k)}\}_{k=1}^{K}$
    $loss_{dis}(b) \leftarrow \log \dfrac{p(g_{\mu_{\boldsymbol{z}_2}}(\tilde{\boldsymbol{x}}^{(k_b,n_b)})|h_{\mu_{\boldsymbol{\mu}_2}}(k_b))}{\sum_{k=1}^{K} p(g_{\mu_{\boldsymbol{z}_2}}(\tilde{\boldsymbol{x}}^{(k_b,n_b)})|h_{\mu_{\boldsymbol{\mu}_2}}(k))}$
    $loss_{gen}(b) \leftarrow \mathcal{L}(\theta, \phi; \tilde{\boldsymbol{x}}^{(k_b,n_b)})$
    $loss \leftarrow -\sum_{b=1}^{bs}(loss_{gen}(b) + \alpha * loss_{dis}(b))/bs$
    $f_*, g_*, h_{\mu_{\boldsymbol{\mu}_2}} \leftarrow Opt(loss, \{f_*, g_*, h_{\mu_{\boldsymbol{\mu}_2}}\})$
  end for
end while
return f_*, g_*

proposed algorithm introduces overhead associated with updating the lookup table whenever a new batch of sequences is sampled. However, this cost can be amortized by increasing the number of segment batches, NB_{seg}, for each batch of sequences.

6.3 Experimental Setup

We evaluate our training algorithm on a wide variety of datasets, ranging from 3 to 1,000 hours, including both clean and noisy, close-talking and distant speech. In this section, we describe the datasets, and introduce FHAVE models and their training configurations.

6.3.1 Datasets

Four different corpora are used for our experiments: TIMIT (Garofolo et al., 1993), Aurora-4 (Pearce, 2002), AMI (Carletta, 2007), and LibriSpeech (Panayotov et al., 2015), the former two of which were introduced in the previous chapters. In this

chapter, we use TIMIT to study the disentanglement performance between phonetic and speaker information, because it contains manually annotated time-aligned phonetic transcripts. For aurora4, we use it to study how speaker and noise information are represented in the space of the latent sequence variables.

The AMI corpus consists of 100 hours of meeting recordings, recorded in three different meeting rooms with different acoustic properties, and with multiple attendants. Multiple microphones are used for each session, including individual headset microphones (IHM), and far-field microphone arrays. IHM and single distant microphone (SDM) recordings from the training set are mixed to form a training set for the FHVAE models, including over 200,000 utterances according to the segmentation provided in the corpus.

The largest corpus we evaluate on is the LibriSpeech corpus, which contains 1,000 hours of read speech sampled at 16kHz. This corpus is based on the LibriVox's project, where world-wide volunteers record public domain texts to create free public domain audio books.

6.3.2 Training and Model Configurations

Speech segments of 20 frames, represented with 80-dimensional log Mel-scale filter bank coefficients (FBank), are used as inputs to FHVAE models. We denote each segment with $\boldsymbol{x} = [x_1, \cdots, x_{20}]$. The variance of $\boldsymbol{z}_2$'s prior is set to $\sigma^2_{\boldsymbol{z}_2} = 0.25$, and the dimension of $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are both 32. The conditional mean and variance predictor for each variable (i.e., $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, and $\boldsymbol{x}$) shares a common stacked LSTM pre-network, followed by two different single-layer affine transform networks, μ_* and σ^2_*, predicting the conditional mean and variance respectively. Specifically, a stacked LSTM with 2 layers and 256 memory cells are used for all three pre-networks, illustrated in Figure 6-2 with blocks filled with dark colors. Affine transform networks of $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ encoders take as input the output from the last time step of both layers, which sums to 512 dimension. As for the $\boldsymbol{x}$ decoder, the affine transform network takes as input the LSTM output of the last layer from each time step t, and predicts the probability distribution of the corresponding frame $p(x_t|\boldsymbol{z}_1, \boldsymbol{z}_2)$. The same sampled $\tilde{\boldsymbol{z}}_1$ and $\tilde{\boldsymbol{z}}_2$

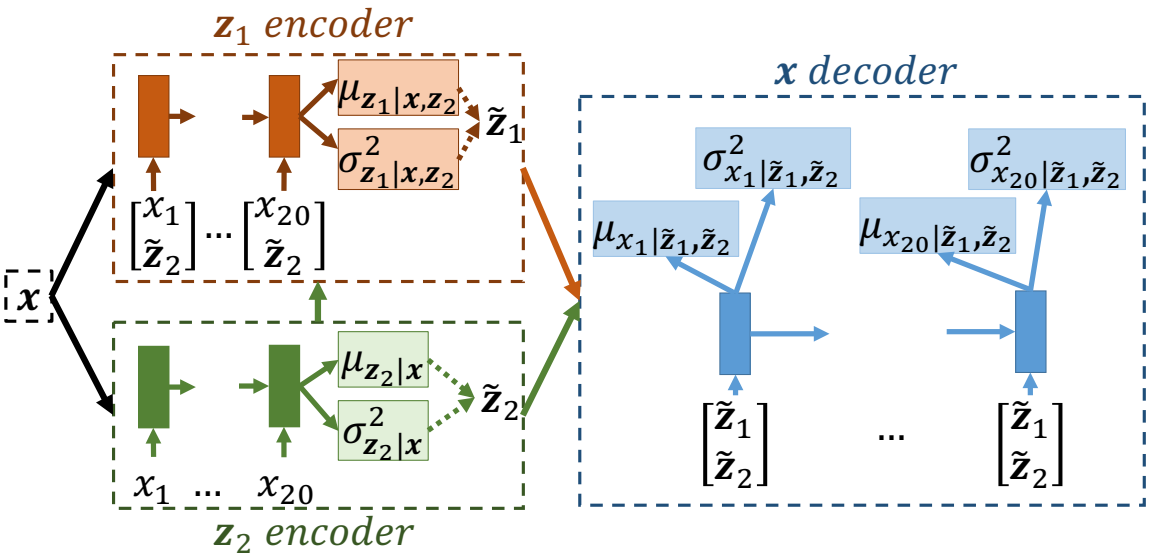


Figure 6-2: The proposed FHVAE architecture consists of two encoders (orange and green) and one decoder (blue). $\boldsymbol{x} = [x_1, \cdots, x_{20}]$ is a segment of 20 frames. Dotted lines in the encoders denote sampling from parametric distributions.

from the posterior distributions are concatenated and used as input for the LSTM decoder at each step. Sampling is done by introducing auxiliary input variables for the reparameterization trick (Kingma and Welling, 2013), in order to keep the entire network differentiable with respect to the objective.

FHVAE models are trained to optimize the discriminative segment variational lower bound with $\alpha = 10$. We set sequence batch size K =2,000 for TIMIT and Aurora-4, and K =5,000 for the others. Adam (Kingma and Ba, 2014) with $\beta_1 = 0.95$ and $\beta_2 = 0.999$ is used to optimize all models. Tensorflow (Abadi et al., 2016) is used for implementation. Training is done for 500,000 steps, terminating early if the segmental variational lower bound on a held-out validation set is not improved for 50,000 steps.

6.4 Results and and Discussion

6.4.1 Time and Memory Complexity

One feature of our proposed training algorithm is the ability to control memory complexity. We found that a training set with over 100,000 sequences would exhaust a single 8GB GPU memory when using the original training algorithm. Hence, it was not feasible for the AMI and the LibriSpeech corpus, while the proposed algorithm does not suffer from the same problem. Another feature of hierarchical sampling is to control the time complexity of computing the discriminative loss. To study how sequence batch size affects the optimization step (line 8 in Algorithm 1), we evaluate the processing time of that step by varying K from 20 to 20,000 and show the results in Table 6.2.

We can observe that when $K \leq$ 2,000, the time complexity of computing the discriminative loss is fractional compared to computing the variational lower bound. However, when $K >$2,000, the increased computation time grows proportional to the sequence batch size, so that computation of the discriminative loss starts to dominate the time complexity. In practice, given a new encoder/decoder architecture, we can investigate the computation overhead resulting from the discriminative loss using such a method, and it is possible to determine some K that introduces negligible overhead for optimization.

Table 6.2: Processing time of the optimization step with different sequence batch size K.

K	10	100	1000	2000	5000	10000	20000
Time (ms)	84	84	86	87	103	147	230

6.4.2 Evaluating Disentanglement Performance

To examine whether an FHVAE is successfully trained, we need to inspect its performance at disentangling sequence-level generating factors (e.g. speaker identity, noise condition, and channel condition) from segment-level generating factors (e.g.

phonetic content) in the latent space. For quantitative evaluation, we reproduce the speaker verification experiments in Hsu et al. (2017b). The FHVAE model trained with hierarchical sampling achieves 1.64% equal error rate on TIMIT, matching the performance of the original training algorithm (1.34%). In the following sections, we proceed with two qualitative evaluation methods.

t-SNE Visualization of Latent Variables

We start with selecting a batch of labeled segments $(\boldsymbol{x}, y)$, where y denotes the values of the associated generating factors, for example $y =$ (phone-id, speaker-id). We then infer $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ of these segments, and project them separately to a two-dimensional space using t-Distributed Stochastic Neighbor Embedding (t-SNE) (Van Der Maaten, 2014). Each generating factor is used to color-code both projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. Successful disentanglement would result in segments of the same sequence-level generating factors forming clusters in the projected $\boldsymbol{z}_2$ space but not in the projected $\boldsymbol{z}_1$ space, and vice versa.

For all four datasets, speaker label, a sequence-level generating factor, is available for each segment. Since time-aligned phonetic transcripts are available for TIMIT, it is also possible to derive phone labels, which is a segment-level generating factor. Following Halberstadt (1999), we further reduce the 61 phonemes to three phonetic subsets: sonorant (SON), obstruent (OBS), and silence (SIL) for better color-coding. In addition, noise types can be obtained for Aurora-4, and microphone types can be obtained for AMI, which are both sequence-level generating factors.

Results of t-SNE projections for models trained on each dataset are shown in Figures 6-3, 6-4, 6-5, and 6-6, where each point represents one segment. It can be observed that in each of the projected $\boldsymbol{z}_2$ spaces, segments of the same sequence-level generating factors (speaker/noise/channel) always form clusters. When segments are generated conditioned on multiple sequence-level generating factors, as in Aurora-4 and AMI, the segments actually cluster hierarchically. In contrast, the distribution of projected $\boldsymbol{z}_1$'s does not vary between different values of these generating factors, which implies that $\boldsymbol{z}_1$ does not contain much information about them. The opposite

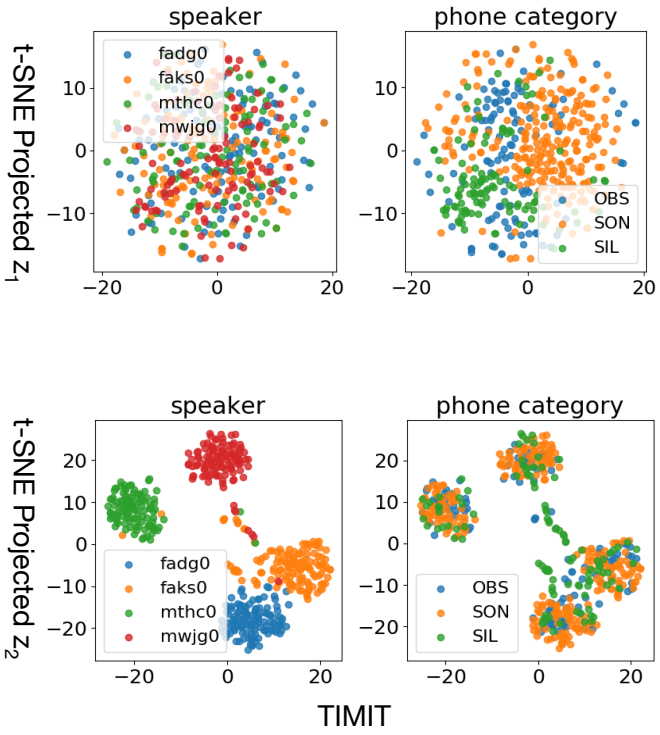


Figure 6-3: Scatter plots of t-SNE projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ with models trained on TIMIT. Each point represents one segment. Different colors are used to code segments of different labels with respect to the generating factor shown at the title of each plot.

phenomenon can be observed from the phone category-coded plots, where segments belonging to the same phonetic subset cluster in the projected $\boldsymbol{z}_1$ space, but not in the projected $\boldsymbol{z}_2$ space. These results suggest that FHVAEs trained with hierarchical sampling can achieve desirable disentanglement for these conditions.

Reconstructing Re-combined Latent Variables

Given two segments, $\boldsymbol{x}^A$ and $\boldsymbol{x}^B$, we sample a segment $\boldsymbol{x}^C \sim p(\boldsymbol{x}|\tilde{\boldsymbol{z}}_1^A, \tilde{\boldsymbol{z}}_2^B)$, where $\tilde{\boldsymbol{z}}_1^A$ is a sampled latent segment variable conditioned on $\boldsymbol{x}^A$, and $\tilde{\boldsymbol{z}}_2^B$ is a sampled latent sequence variable conditioned on $\boldsymbol{x}^B$. With a successfully trained FHVAE, $\boldsymbol{x}^C$ should exhibit the segment-level attributes of $\boldsymbol{x}^A$, and the sequence-level attributes of $\boldsymbol{x}^B$. Here we show results of the model trained on the AMI corpus in Figure 6-7. Eight segments are sampled for $\boldsymbol{x}^A$, as shown in the upper right corner of the figure. Among these segments, the four leftmost ones are close-talking while the rest are far-field,

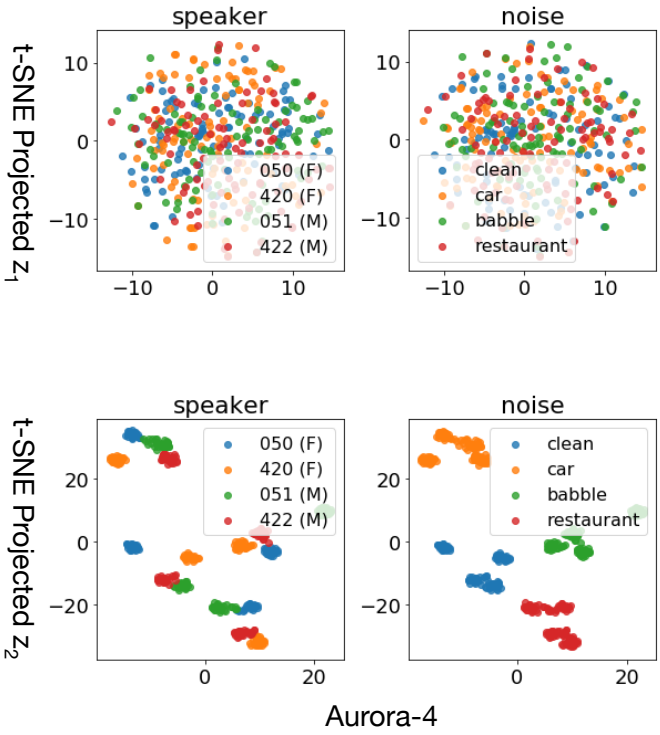


Figure 6-4: Scatter plots of t-SNE projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ with models trained on Aurora-4. Each point represents one segment. Different colors are used to code segments of different labels with respect to the generating factor shown at the title of each plot.

and the first, second, fifth, and sixth from left are female speakers while the rest are males. For $\boldsymbol{x}^B$, three segments of different sequence-level generating factors are sampled, as shown on the left half of the figure. The segments used to infer $\tilde{\boldsymbol{z}}_2^B$ are highlighted in red boxes; we show the surrounding frames of those segments to better illustrate how sequence-level generating factors affect realization of observations.

Samples of $\boldsymbol{x}^C$ generated by re-combining latent variables are shown in the lower right corner of the figure. It can be clearly observed that $\boldsymbol{x}^C$ presents the same sequence-level generating factors as $\boldsymbol{x}^B$,[1] whose latent sequence variable $\boldsymbol{x}^C$ conditions on. Meanwhile, the phonetic content of $\boldsymbol{x}^C$ stays consistent with $\boldsymbol{x}^A$,[2] whose latent segment variable $\boldsymbol{x}^C$ conditions on. The clear differentiation of generating factors encoded in each sets of latent variables again corroborates the success of our proposed

[1]In these images, harmonic spacing is the clearest cue for fundamental frequency differences. Far-field recordings tend to have lower signal-to-noise ratios, which results in blurrier images.

[2]Phonetic content can usually be determined by the spectral envelope, and relative position of formants.

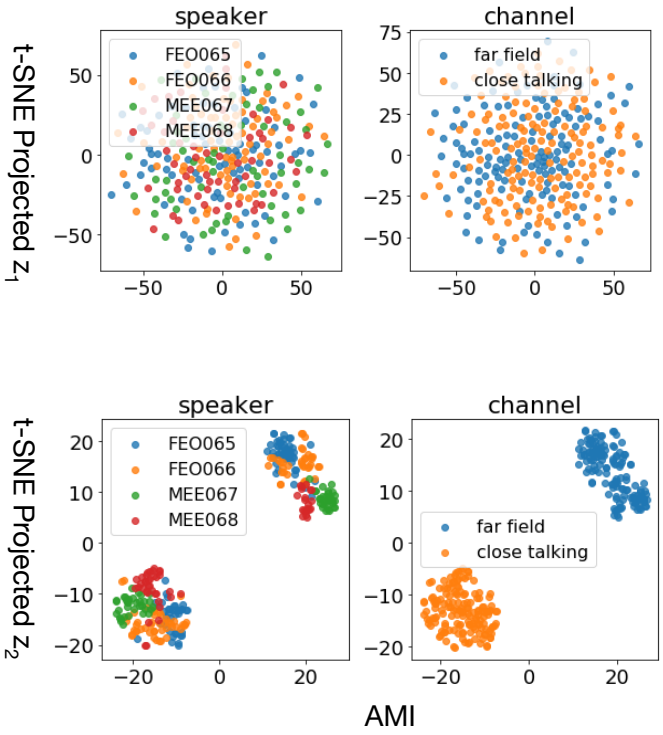


Figure 6-5: Scatter plots of t-SNE projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ with models trained on AMI. Each point represents one segment. Different colors are used to code segments of different labels with respect to the generating factor shown at the title of each plot.

algorithm in training FHVAE models.

6.5 Conclusions

In this chapter, we discussed the scalability limitations of the original FHVAE training algorithm in terms of runtime, memory, and hyperparameter optimization, and proposed a hierarchical sampling algorithm to address this problem. Comprehensive study of the memory and time complexity, as well as disentanglement performance, verified the effectiveness of the proposed algorithm on all scales of datasets, ranging from 3 to 1,000 hours.

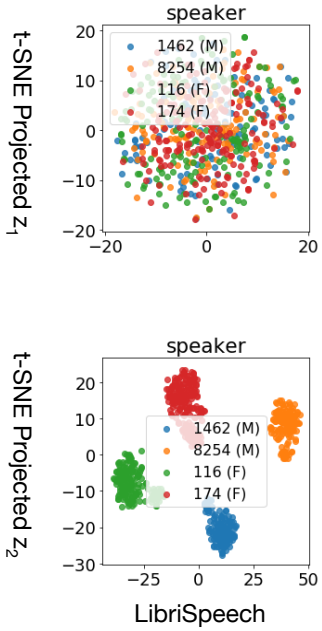


Figure 6-6: Scatter plots of t-SNE projected $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ with models trained on LibriSpeech. Each point represents one segment. Different colors are used to code segments of different labels with respect to the generating factor shown at the title of each plot.

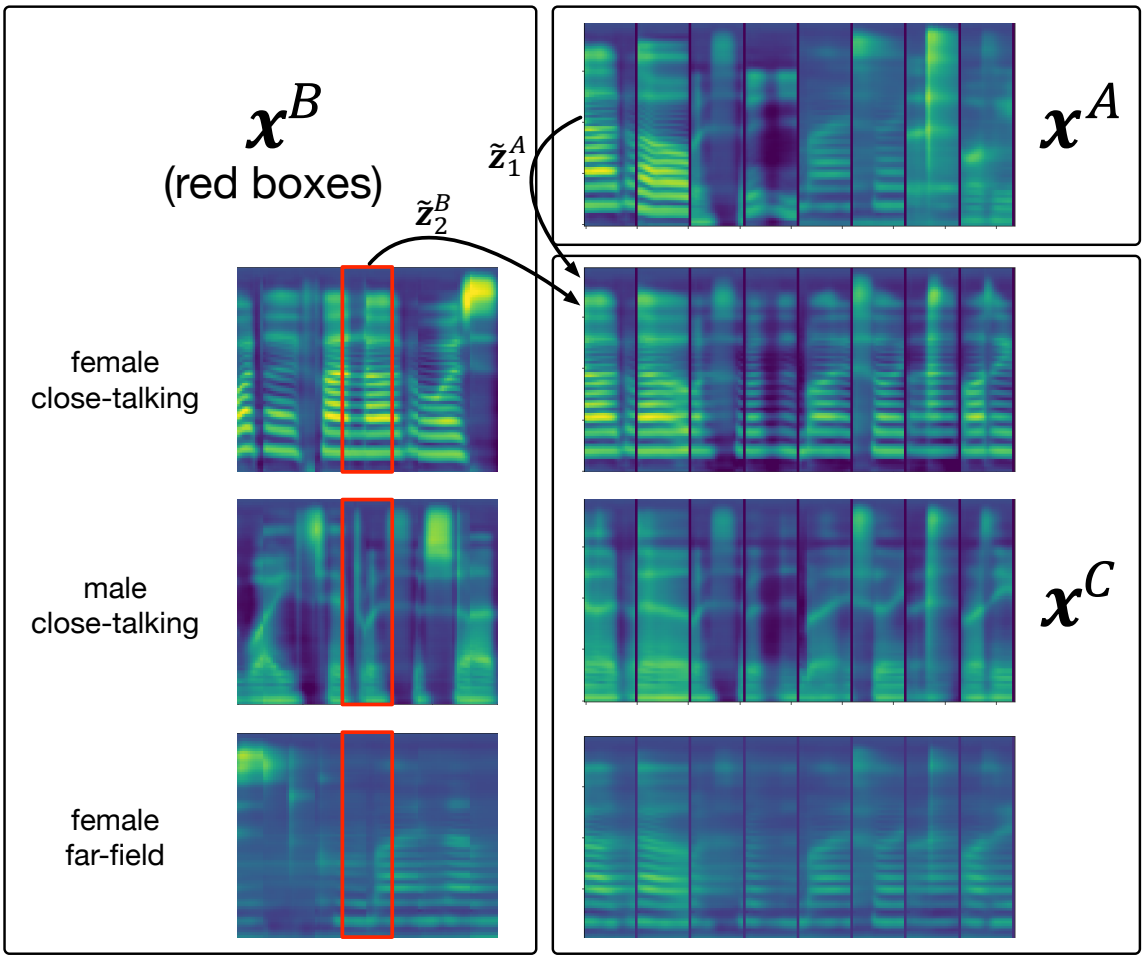


Figure 6-7: Results of decoding re-combined latent variables. A segment in the $\boldsymbol{x}^C$ block is generated conditioned on the latent segment variable of a segment in the block $\boldsymbol{x}^A$ of the same column, and conditioned on the latent sequence variable of a red-box highlighted segment in the block $\boldsymbol{x}^B$ of the same row.

Chapter 7

Conclusion and Future Work

7.1 Summary of Contributions

In this thesis, we investigate neural variational inference models for learning underlying explanatory factors of speech, such as speaker identity, phonetic content, and background noise type. Specifically, we:

1. Apply an existing VAE model to learn a representation for speech, and derive operations in the latent space to transform attributes of speech.

2. Propose an unsupervised domain adaptation technique for robust automatic speech recognition via data augmentation using VAE-based speech attribute transformation.

3. Present a novel FHVAE model that learns a disentangled and interpretable representation from sequential data without supervision.

4. Demonstrate that the learned disentangled representations from FHVAEs can be utilized in a wide variety of learning scenarios, including zero-shot learning, supervised learning, and domain adaptation.

5. Identify scalability issues in the original FHVAE training algorithm with respect to the dataset size, and propose a hierarchical sampling algorithm to scale FHVAE training to datasets of over a thousand hours of speech.

In Chapter 2, we apply a convolutional VAE model to learn a representation from short speech segments. We demonstrate that a VAE learns a smooth mapping from the latent space to the generated speech segment, such that by interpolating two latent variables, we can obtain a generated speech segment with interpolated attributes, for example, a segment with the fundamental frequency in between that of the two speech segments whose latent variables are inferred from. In addition, we derive operations for modifying attributes of a speech segment based on the assumption that independent attributes are modeled by orthogonal subspaces in the latent space, which we also verify empirically.

In Chapter 3, we first make an observation that nuisance attributes that are independent of the linguistic content are mostly consistent among segments within an utterance. By combining this observation with the latent space operations proposed in the previous chapter, we propose two unsupervised data augmentation techniques: *nuisance factor replacement* and *latent nuisance subspace perturbation*, which transform nuisance attributes of an utterance without altering the linguistic content. We apply these techniques to address the unsupervised domain adaptation scenario, where the distribution of the nuisance attributes of the annotated training data is mismatched from the testing data, by transforming the nuisance attributes of the training data such that they resemble those of the testing data.

In Chapter 4, we present a novel FHVAE model, which combines the neural variational inference framework with a hierarchical generative model that better describes sequential data generation. The proposed FHVAE model is capable of learning disentangled and interpretable representations, which separate sequence-level attributes (e.g. speaker identity, background noise type) and segment-level attributes (e.g. phonetic content) into separate latent variables.

In Chapter 5, we study the applications of the disentangled representations learned using FHVAE models in different learning scenarios. As a generative model, the FHVAE model is also capable of transforming attributes of an utterance, similar to what VAE models can do; furthermore, due to the explicit factorization assumption between the variables modeling sequence-level and segment-level attributes in the

graphical model, the FHVAE model achieves better speech transformation performance compared to the VAE model. We demonstrate that such an ability through voice conversion and de-noising using FHVAE models. In addition to speech generation, we also apply the disentangled representations to speaker verification and robust ASR tasks, where the sequence-level latent variable is used as a representation for speaker information, and the segment-level latent segment variable is used as a domain-invariant acoustic feature representation for ASR.

In Chapter 6, we analyze the original training algorithm for FHVAE models proposed in Chapter 4, and determine that the algorithm is not scalable with respect to dataset size and requires additional hyperparameter tuning for different sizes of datasets. To address this issue, we propose a hierarchical algorithm for training FHVAE models, which can scale to arbitrarily large datasets with a fixed run-time and memory cost per training step.

7.2 Future Work

This thesis work can be extended in three different directions:

1. Increase the capacity of the neural network modules to parameterize the distributions of the generative network and the inference network.

2. Explore a wider variety of speech applications based on our proposed models, and apply our framework to model different types of sequential data.

3. Utilize synchronized multimodal data and propose better graphical models to capture the generative process of such data.

From the modeling perspective, each neural network module used in our VAE or FHVAE models can be replaced with more powerful ones. To parameterize the generative networks, we adopt simple convolutional networks and non-autoregressive recurrent networks in this thesis. These models assume conditional independence between different observed frames in a sequence, which is actually an overly simplified assumption, because there exist strong correlations between consecutive speech

frames. To address such issues, conditional autoregressive models (Mehri et al., 2016; van den Oord et al., 2016) can be applied instead, which take not only the latent variable as input, but also the output from the previous step, to predict the distribution over the current frame.

As for the variational inference network, we assume a factorized Gaussian posterior distribution for both VAE and FHVAE, with its mean and variance parameterized by convolutional or recurrent networks. This module can also be improved by incorporating recently developed normalizing flow (Rezende and Mohamed, 2015) or inverse autoregressive flow (Kingma et al., 2016) techniques to include a larger family of posterior distributions that the variational inference model can characterize.

From the application point of view, besides the applications we explore in this thesis, many other speech applications can benefit from utilizing a representation that disentangles linguistic factors from speaker and noise-related factors. For example, to discover the set of acoustic units for under-resourced languages (Jansen et al., 2010; Lee and Glass, 2012) or to obtain semantically-rich speech embeddings (Harwath and Glass, 2017; Chung and Glass, 2018), it is desirable to remove non-linguistic factors from the acoustic features. On the other hand, to compute sequence-level statistics for speaker adaptation (Saon et al., 2013) or environment adaptation (Feng et al., 2017), the sequence-level latent variables (i.e., *s-vectors)* extracted from our FHVAE model can be exploited to replace i-vectors (Dehak et al., 2011).

In addition to modeling a generative process for speech, our FHVAE model can potentially be applied to model a generative process of texts, images, or videos, which all involve certain forms of hierarchy during generation, for learning disentangled representations. Take a video of a person walking for example, the subject and the scene are the sequence-level attributes that are consistent within a video clip, while the posture of the person is a frame-level attribute that changes from frame to frame.

Last but not least, in this thesis, we have only explored representation learning from unimodal data; however, such data is still very different from the form of data perceived by humans during their learning process, which are parallel multimodal data, such as parallel video and audio streams. To computational simulate a human-

like learning process and to learn representations disentangled at the semantic level, we should also extend our framework to model the generation of multimodal data, the synchronization of which can provide additional information for disentangling underlying generating factors (Harwath et al., 2016).

Appendix A

Derivation of Sequence Variational Lower Bound

The variational lower bound for the marginal probability of a sequence $\boldsymbol{X}$ can be derived as follows:

$$
\begin{aligned}
\log p(\boldsymbol{X}) \geq & \mathcal{L}(\theta, \phi; \boldsymbol{X}) \\
= & \mathbb{E}_{q(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{\mu}_2 | \boldsymbol{X})}\big[\log \frac{p(\boldsymbol{\mu}_2)\prod_{n=1}^{N} p(\boldsymbol{x}^{(n)}|\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)}) p(\boldsymbol{z}_1^{(n)}) p(\boldsymbol{z}_2^{(n)}|\boldsymbol{\mu}_2)}{q(\boldsymbol{\mu}_2)\prod_{n=1}^{N} q(\boldsymbol{z}_1^{(n)}|\boldsymbol{x}^{(n)}, \boldsymbol{z}_2^{(n)}) q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})}\big] \\
= & \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})}\big[\log p(\boldsymbol{x}^{(n)}|\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)})\big] \\
& - \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})}\big[D_{KL}(q(\boldsymbol{z}_1^{(n)}|\boldsymbol{x}^{(n)}, \boldsymbol{z}_2^{(n)})||p(\boldsymbol{z}_1^{(n)}))\big] \\
& - \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\mu}_2)}\big[D_{KL}(q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})||p(\boldsymbol{z}_2^{(n)}|\boldsymbol{\mu}_2))\big] \qquad \text{(A.1)} \\
& - D_{KL}(q(\boldsymbol{\mu}_2)||p(\boldsymbol{\mu}_2)). \qquad \text{(A.2)}
\end{aligned}
$$

The expected KL-divergence in Eq. A.1 of two Gaussian distributions, $q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})$ and $p(\boldsymbol{z}_2^{(n)}|\boldsymbol{\mu}_2)$, over a Gaussian $q(\boldsymbol{\mu}_2) = \mathcal{N}(\boldsymbol{\mu}_2|\tilde{\boldsymbol{\mu}}_2, \sigma^2_{\tilde{\boldsymbol{\mu}}_2}\boldsymbol{I})$ can be computed analytically. Let J be the dimensionality of $\boldsymbol{z}_2$. Let $\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_2}$ and $\hat{\boldsymbol{\sigma}}_{\boldsymbol{z}_2}$ denote the variational mean and standard deviation evaluated at $\boldsymbol{x}^{(n)}$, and let $\mu_{2,j}$, $\tilde{\mu}_{2,j}$, $\hat{\mu}_{\boldsymbol{z}_2,j}$ and $\hat{\sigma}_{\boldsymbol{z}_2,j}$ denote the

j-th element of these vectors. We have:

$$
\begin{aligned}
&\mathbb{E}_{q(\boldsymbol{\mu}_2)}\big[D_{KL}(q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})||p(\boldsymbol{z}_2^{(n)}|\boldsymbol{\mu}_2))\big] \\
&= \mathbb{E}_{q(\boldsymbol{\mu}_2)}\big[D_{KL}(\mathcal{N}(\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_2}, \hat{\boldsymbol{\sigma}}^2_{\boldsymbol{z}_2})||\mathcal{N}(\boldsymbol{\mu}_2, \sigma^2_{\boldsymbol{z}_2}\boldsymbol{I}))\big] \\
&= \mathbb{E}_{q(\boldsymbol{\mu}_2)}\big[-\frac{1}{2}\sum_{j=1}^{J}(1+\log\frac{\hat{\sigma}^2_{\boldsymbol{z}_2,j}}{\sigma^2_{\boldsymbol{z}_2}} - \frac{(\hat{\mu}_{\boldsymbol{z}_2,j}-\mu_{2,j})^2+\hat{\sigma}^2_{\boldsymbol{z}_2,j}}{\sigma^2_{\boldsymbol{z}_2}})\big] \\
&= -\frac{1}{2}\sum_{j=1}^{J}(1+\log\frac{\hat{\sigma}^2_{\boldsymbol{z}_2,j}}{\sigma^2_{\boldsymbol{z}_2}} - \frac{\hat{\sigma}^2_{\boldsymbol{z}_2,j}}{\sigma^2_{\boldsymbol{z}_2}} - \mathbb{E}_{q(\boldsymbol{\mu}_2)}\big[\frac{(\hat{\mu}_{\boldsymbol{z}_2,j}-\mu_{2,j})^2}{\sigma^2_{\boldsymbol{z}_2}}\big]) \\
&= D_{KL}(\mathcal{N}(\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_2}, \hat{\boldsymbol{\sigma}}^2_{\boldsymbol{z}_2})||\mathcal{N}(\tilde{\boldsymbol{\mu}}_2, \sigma^2_{\boldsymbol{z}_2})) + \frac{J}{2}\frac{\sigma^2_{\tilde{\boldsymbol{\mu}}_2}}{\sigma^2_{\boldsymbol{z}_2}} \\
&= D_{KL}(q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})||p(\boldsymbol{z}_2^{(n)}|\tilde{\boldsymbol{\mu}}_2)) + \frac{J}{2}\frac{\sigma^2_{\tilde{\boldsymbol{\mu}}_2}}{\sigma^2_{\boldsymbol{z}_2}}
\end{aligned} \tag{A.3}
$$

The KL-divergence in Eq. A.2 can also be computed analytically and rewritten as follows:

$$
\begin{aligned}
&D_{KL}(q(\boldsymbol{\mu}_2)||p(\boldsymbol{\mu}_2)) \\
&= D_{KL}(\mathcal{N}(\tilde{\boldsymbol{\mu}}_2, \sigma^2_{\tilde{\boldsymbol{\mu}}_2}\boldsymbol{I})||\mathcal{N}(0, \sigma^2_{\boldsymbol{\mu}_2}\boldsymbol{I})) \\
&= -\frac{1}{2}\sum_{j=1}^{J}(1+\log\frac{\sigma^2_{\tilde{\boldsymbol{\mu}}_2}}{\sigma^2_{\boldsymbol{\mu}_2}} - \frac{(\tilde{\mu}_{2,j}-0)^2+\sigma^2_{\tilde{\boldsymbol{\mu}}_2}}{\sigma^2_{\boldsymbol{\mu}_2}}) \\
&= -\frac{1}{2}\sum_{j=1}^{J}(1+\log\sigma^2_{\tilde{\boldsymbol{\mu}}_2}) - \frac{1}{2}\log 2\pi - \log p(\tilde{\boldsymbol{\mu}}_2)
\end{aligned} \tag{A.4}
$$

By replacing Eq. A.1 and A.2 with Eq. A.3 and A.4 respectively, we rewrite the

variational lower bound for a sequence $\boldsymbol{X}$ as follows:

$$
\begin{aligned}
\mathcal{L}(\theta, \phi; \boldsymbol{X}) =& \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})}\big[\log p(\boldsymbol{x}^{(n)}|\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)})\big] \\
& - \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})}\big[D_{KL}(q(\boldsymbol{z}_1^{(n)}|\boldsymbol{x}^{(n)}, \boldsymbol{z}_2^{(n)})||p(\boldsymbol{z}_1^{(n)}))\big] \\
& - \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\mu}_2)}\big[D_{KL}(q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})||p(\boldsymbol{z}_2^{(n)}|\boldsymbol{\mu}_2))\big] \\
& - D_{KL}(q(\boldsymbol{\mu}_2)||p(\boldsymbol{\mu}_2)). \\
=& \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})}\big[\log p(\boldsymbol{x}^{(n)}|\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)})\big] \\
& - \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})}\big[D_{KL}(q(\boldsymbol{z}_1^{(n)}|\boldsymbol{x}^{(n)}, \boldsymbol{z}_2^{(n)})||p(\boldsymbol{z}_1^{(n)}))\big] \\
& - \sum_{n=1}^{N} D_{KL}(q(\boldsymbol{z}_2^{(n)}|\boldsymbol{x}^{(n)})||p(\boldsymbol{z}_2^{(n)}|\tilde{\boldsymbol{\mu}}_2)) - \frac{J}{2}\frac{\sigma^2_{\tilde{\boldsymbol{\mu}}_2}}{\sigma^2_{\boldsymbol{z}_2}} \\
& + \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma^2_{\tilde{\boldsymbol{\mu}}_2}) + \frac{1}{2}\log 2\pi + \log p(\tilde{\boldsymbol{\mu}}_2) \\
=& \sum_{n=1}^{N}(\mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)}|\tilde{\boldsymbol{\mu}}_2) - \frac{J}{2}\frac{\sigma^2_{\tilde{\boldsymbol{\mu}}_2}}{\sigma^2_{\boldsymbol{z}_2}}) + \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma^2_{\tilde{\boldsymbol{\mu}}_2}) + \frac{1}{2}\log 2\pi + \log p(\tilde{\boldsymbol{\mu}}_2) \\
=& \sum_{n=1}^{N} \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)}|\tilde{\boldsymbol{\mu}}_2) + \log p(\tilde{\boldsymbol{\mu}}_2) + const
\end{aligned}
$$

Appendix B

Derivation of the Inferred S-Vector

As described in Section 2.2, inference of the s-vector $\boldsymbol{\mu}_2$ of an unseen utterance $\boldsymbol{X} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$ is cast as an approximated maximum a posterior estimation problem, which uses the conditional segment variational lower bound, $\mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)}|\boldsymbol{\mu}_2)$, to approximate the conditional likelihood of a segment, $\log p_\theta(\boldsymbol{x}^{(n)}|\boldsymbol{\mu}_2)$. Let J be the dimensionality of $\boldsymbol{z}_2$. Let $\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_2}^{(n)}$ denote the variational mean of $\boldsymbol{z}_2$ evaluated at $\boldsymbol{x}^{(n)}$, and let $\mu_{2,j}$ and $\hat{\mu}_{\boldsymbol{z}_2,j}^{(n)}$ denote the j-th element of these vectors. The optimal $\boldsymbol{\mu}_2^*$ can be derived as follows:

$$
\begin{aligned}
\boldsymbol{\mu}_2^* &= \underset{\boldsymbol{\mu}_2}{\operatorname{argmax}} \sum_{n=1}^{N} \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(n)}|\boldsymbol{\mu}_2) + \log p_\theta(\boldsymbol{\mu}_2) \\
&= \underset{\boldsymbol{\mu}_2}{\operatorname{argmax}} \sum_{n=1}^{N} -D_{KL}(q_\phi(\boldsymbol{z}_2^{(n)}|\tilde{x}^{(n)})||p_\theta(\boldsymbol{z}_2^{(n)}|\boldsymbol{\mu}_2)) + \log p_\theta(\boldsymbol{\mu}_2) \\
&= \underset{\boldsymbol{\mu}_2}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{j=1}^{J} \frac{-(\hat{\mu}_{\boldsymbol{z}_2,j}^{(n)} - \mu_{2,j})^2}{\sigma_{\boldsymbol{z}_2}^2} + \sum_{j=1}^{J} \frac{-(\mu_{2,j} - 0)^2}{\sigma_{\boldsymbol{\mu}_2}^2} \\
&= \underset{\boldsymbol{\mu}_2}{\operatorname{argmax}}\, f(\boldsymbol{\mu}_2),
\end{aligned}
$$

where $f(\cdot)$ is a concave quadratic function that has only one maximum point. We then have:

$$\frac{\partial f(\boldsymbol{\mu}_2)}{\partial \boldsymbol{\mu}_2}\Big|_{\boldsymbol{\mu}_2=\tilde{\boldsymbol{\mu}}_2^*} = 0$$
$$\boldsymbol{\mu}_2^* = \frac{\sum_{n=1}^{N} \hat{\boldsymbol{\mu}}_{\boldsymbol{z}_2}^{(n)}}{N + {}^{\sigma_{\boldsymbol{z}_2}^2}/_{\sigma_{\boldsymbol{\mu}_2}^2}}$$

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

Xavier Anguera, Chuck Wooters, and Javier Hernando. Acoustic beamforming for speaker diarization of meetings. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2011–2022, 2007.

Jean Carletta. Unleashing the killer corpus: experiences in creating the multi-everything ami meeting corpus. *Language Resources and Evaluation*, 41(2):181–190, 2007.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, page 2172–2180, 2016.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Katya Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. *arXiv preprint arXiv:1712.01769*, 2017.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015a.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015b.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Yu-An Chung and James Glass. Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech. *arXiv preprint arXiv:1803.08976*, 2018.

Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(9):1469–1477, 2015.

Najim Dehak, Reda Dehak, Patrick Kenny, Niko Brümmer, Pierre Ouellet, and Pierre Dumouchel. Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification. In *Interspeech*, volume 9, pages 1559–1562, 2009.

Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

Jun Du, Qing Wang, Tian Gao, Yong Xu, Li-Rong Dai, and Chin-Hui Lee. Robust speech recognition with speech enhanced deep neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Harrison Edwards and Amos Storkey. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

Hakan Erdogan, Tomoki Hayashi, John R Hershey, Takaaki Hori, Chiori Hori, Wei-Ning Hsu, Suyoun Kim, Jonathan Le Roux, Zhong Meng, and Shinji Watanabe. Multi-channel speech recognition: Lstms all the way through. In *CHiME-4 workshop*, 2016.

Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

Xue Feng, Yaodong Zhang, and James Glass. Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1759–1763. IEEE, 2014.

Xue Feng, Brigitte Richardson, Scott Amman, and James Glass. An environmental feature representation for robust speech recognition and for environment identification. *Proc. of Interspeech. IEEE*, pages 3078–3082, 2017.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, pages 2199–2207, 2016.

Josué Fredes, José Novoa, Simon King, Richard M Stern, and Nestor Becerra Yoma. Locally normalized filter banks applied to deep neural-network-based robust speech recognition. *IEEE Signal Processing Letters*, 24(4):377–381, 2017.

Mark JF Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

John Garofalo, David Graff, Doug Paul, and David Pallett. Csr-i (wsj0) complete. *Linguistic Data Consortium, Philadelphia*, 2007.

John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013.

Andrew King Halberstadt. *Heterogeneous acoustic measurements and multiple classifiers for speech recognition*. PhD thesis, Massachusetts Institute of Technology, 1999.

David Harwath and James R Glass. Learning word-like units from joint audio-visual analysis. *arXiv preprint arXiv:1701.07481*, 2017.

David Harwath, Antonio Torralba, and James Glass. Unsupervised learning of spoken language with visual context. In *Advances in Neural Information Processing Systems*, pages 1858–1866, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Wei-Ning Hsu and James Glass. Extracting domain invariant features by unsupervised learning for robust automatic speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018a.

Wei-Ning Hsu and James Glass. Scalable factorized hierarchical variational autoencoder training. *arXiv preprint arXiv:1804.03201*, 2018b.

Wei-Ning Hsu, Yu Zhang, and James Glass. A prioritized grid long short-term memory rnn for speech recognition. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 467–473. IEEE, 2016a.

Wei-Ning Hsu, Yu Zhang, Ann Lee, and James R Glass. Exploiting depth and highway connections in convolutional recurrent deep neural networks for speech recognition. In *INTERSPEECH*, pages 395–399, 2016b.

Wei-Ning Hsu, Yu Zhang, and James Glass. Learning latent representations for speech generation and transformation. In *Interspeech*, pages 1273–1277, 2017a.

Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in Neural Information Processing Systems*, 2017b.

Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In *Automatic Speech Recognition and Understanding (ASRU), 2017 IEEE Workshop on*. IEEE, 2017c.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Controllable text generation. *arXiv preprint arXiv:1703.00955*, 2017.

Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R Hershey. Single-channel multi-speaker separation using deep clustering. *arXiv preprint arXiv:1607.02173*, 2016.

Navdeep Jaitly and Geoffrey E. Hinton. Vocal tract length perturbation (VTLP) improves speech recognition. In *ICML workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

Aren Jansen, Kenneth Church, and Hynek Hermansky. Towards spoken term discovery at scale with zero resources. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954, 2016.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233, 1999.

Alexander Kain and Michael W. Macon. Spectral voice conversion for text-to-speech synthesis. In *ICASSP*, 1998.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

Brian ED Kingsbury, Nelson Morgan, and Steven Greenberg. Robust speech recognition using the modulation spectrogram. *Speech communication*, 25(1):117–132, 1998.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.

Chia-ying Lee and James Glass. A nonparametric bayesian approach to acoustic model discovery. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 40–49. Association for Computational Linguistics, 2012.

Bo Li, Tara Sainath, Arun Narayanan, Joe Caroselli, Michiel Bacchiani, Ananya Misra, Izhak Shafran, Hasim Sak, Golan Pundak, Kean Chin, et al. Acoustic modeling for google home. *INTERSPEECH-2017*, pages 399–403, 2017.

Jinyu Li, Dong Yu, Jui-Ting Huang, and Yifan Gong. Improving wideband speech recognition using mixed-bandwidth training data in cd-dnn-hmm. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 131–136. IEEE, 2012.

Fu-Hua Liu, Richard M Stern, Xuedong Huang, and Alejandro Acero. Efficient cepstral normalization for robust speech recognition. In *Proceedings of the workshop on Human Language Technology*, pages 69–74. Association for Computational Linguistics, 1993.

Andrew L Maas, Quoc V Le, Tyler M O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng. Recurrent neural networks for noise reduction in robust asr. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

Zhong Meng, Zhuo Chen, Vadim Mazalov, Jinyu Li, and Yifan Gong. Unsupervised adaptation with domain separation networks for robust speech recognition. *arXiv preprint arXiv:1711.08010*, 2017.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014.

Nelson Mogran, Hervé Bourlard, and Hynek Hermansky. Automatic speech recognition: An auditory perspective. In *Speech processing in the auditory system*, pages 309–338. Springer, 2004.

Arun Narayanan and DeLiang Wang. Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7092–7096. IEEE, 2013.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5206–5210. IEEE, 2015.

Douglas B Paul and Janet M Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

David Pearce. *Aurora working group: DSR front end LVCSR evaluation AU/384/02*. PhD thesis, Mississippi State University, 2002.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, 2011.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4153–4156. IEEE, 2012.

Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Interspeech*, pages 338–342, 2014.

George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *ASRU*, pages 55–59, 2013.

George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. *arXiv preprint arXiv:1703.02136*, 2017.

Michael L Seltzer, Dong Yu, and Yongqiang Wang. An investigation of deep neural networks for noise robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7398–7402. IEEE, 2013.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Richard M Stern and Nelson Morgan. Features based on auditory physiology and perception. *Techniques for Noise Robustness in Automatic Speech Recognition*, page 193227, 2012.

Yannis Stylianou. Voice transformation: A survey. In *ICASSP*, pages 3585–3588. IEEE, 2009.

Sining Sun, Binbin Zhang, Lei Xie, and Yanning Zhang. An unsupervised deep domain adaptation approach for robust speech recognition. *Neurocomputing*, 2017.

Tomoki Toda, Yamato Ohtani, and Kiyohiro Shikan. Eigenvoice conversion based on gaussian mixture model. In *Interspeech*, pages 2446–2449, 2006.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016.

Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

Emmanuel Vincent, Shinji Watanabe, Aditya Arie Nugraha, Jon Barker, and Ricard Marxer. An analysis of environment, microphone and data simulation mismatches in robust speech recognition. *Computer Speech & Language*, 2016.

Oriol Vinyals and Suman V Ravuri. Comparing multilayer perceptron to deep belief network tandem features for robust asr. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4596–4599. IEEE, 2011.

Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

D Wong, B Juang, and D Cheng. Very low data rate speech compression with LPC vector and matrix quantization. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'83.*, volume 8, pages 65–68. IEEE, 1983.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*, 2016.

Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7893–7897. IEEE, 2013.

Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Huaming Wang, et al. An introduction to computational networks and the computational network toolkit. Technical report, Tech. Rep. MSR, Microsoft Research, 2014, http://codebox/cntk, 2014.

Yu Zhang, Dong Yu, Michael L Seltzer, and Jasha Droppo. Speech recognition with prediction-adaptation-correction recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5004–5008. IEEE, 2015.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. Highway long short-term memory RNNs for distant speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5755–5759. IEEE, 2016.

Victor Zue, Stephanie Seneff, and James Glass. Speech database development at MIT: TIMIT and beyond. *Speech Communication*, 9(4):351–356, 1990.