Unsupervised Speech Representation Learning

by

Gilles BOULIANNE

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, AUGUST 27, 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Pierre Dumouchel, Thesis Supervisor
Department of Software and IT Engineering, École de technologie supérieure

Mr. Éric Granger, President of the Board of Examiners
Department of Systems Engineering, École de technologie supérieure

Ms. Sylvie Ratté, Member of the Jury
Department of Software and IT Engineering, École de technologie supérieure

Mr. Douglas O'Shaughnessy, External Independent Examiner
Centre Énergie Matériaux Télécommunications, Institut national de la recherche scientifique

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON JULY 14, 2020

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I would like to thank first my thesis supervisor, Pierre Dumouchel, for his unwavering confidence through so many years.

My thanks also go to the board of examiners, for spending time and effort to review and comment my thesis. I am very grateful to them for accepting this extra burden on their already loaded schedules.

I would also like to mention Patrick Kenny, former colleague and mentor, who sparked my interest for the whole field of Bayesian and generative modelling. Discussions between me, the engineer, and him, the mathematician, were always deep and instructive.

But my deepest gratitude goes, without question, to my partner in life, Jacinthe, who encouraged and supported me unfailingly during the whole course of this long adventure.

Finally, this work would not have been possible without CRIM's (Centre de recherche informatique de Montréal) support and its high performance computing software and infrastructure.

Apprentissage non supervisé de représentations distribuées pour la parole

Gilles BOULIANNE

RÉSUMÉ

Les représentations distribuées visent à extraire l'information de haut niveau contenue dans des données brutes, habituellement sous forme de vecteurs de faible dimension. Lorsqu'utilisées comme entrée pour des tâches de classification, elles réduisent la complexité du classificateur, et facilitent l'apprentissage par transfert et l'adaptation au domaine. La représentation est dite interprétable lorsqu'elle saisit des facteurs sous-jacents compréhensibles; elle est alors utile pour explorer et comprendre les données, ou résoudre des problèmes mettant en jeu ces facteurs. En traitement automatique du langage naturel (TALN), des représentations telles que les plongements de mots ou de phrases ("embeddings") sont récemment devenues incontournables. Ces représentations peuvent être apprises sans supervision, sur de grands ensembles de données non annotées, rendant possible l'entraînement de modèles puissants capables de saisir les relations sémantiques requises pour plusieurs problèmes de TALN. En traitement de la parole, des représentations telles que les paramètres à goulot d'étranglement ("bottleneck features") et x-vecteurs ont été proposées, mais leur apprentissage doit être entièrement ou partiellement supervisé avec des annotations, et elles ne visent pas à extraire des facteurs sous-jacents interprétables.

Une représentation non supervisée de la parole, qui serait apprise directement sur un grand corpus enregistré, sans transcription, aurait un impact majeur sur plusieurs applications du traitement de la parole. La transcription est une tâche manuelle coûteuse et se révèle souvent une contrainte importante, particulièrement dans le cas des langues à faibles ressources. Une représentation découplant la variabilité due au locuteur de celle due au contenu phonétique permettrait d'éliminer une des sources principales de confusion, que ce soit pour la transcription automatique ou bien la reconnaissance du locuteur. Malgré ce potentiel intéressant, l'apprentissage non supervisé d'une représentation pour la parole a été moins étudié que l'apprentissage supervisé.

Dans cette thèse, nous présentons un modèle génératif capable d'apprendre une représentation interprétable sans supervision. Plus précisément, nous proposons plusieurs extensions au modèle d'autoencodeur variationnel (VAE), une approche probabiliste qui conjugue l'approche générative et les réseaux neuronaux profonds. Pour inciter le modèle à capturer des facteurs sous-jacents interprétables et découplés, nous lui imposons des biais inductifs inspirés des théories acoustiques et articulatoires de la production de la parole.

Nous proposons d'abord le filtrage temporel comme biais induisant une représentation avec une échelle temporelle différente pour chacune des variables latentes. Il permet de répartir les variables latentes sur une échelle continue, au lieu de l'opposition binaire ou de la structure hiérarchique qui ont été proposées antérieurement.

Nous montrons également comment imposer des distributions a priori multimodales afin de capturer des variables latentes discrètes, et nous présentons pour le VAE deux nouvelles fonctions

de pertes applicables aux variables discrètes, utilisant la réestimation espérance-maximisation avec divergence pairée et l'échantillonnage de la divergence.

De plus, nous proposons l'auto-attention pour ajouter au modèle VAE la capacité de prédire des suites, à notre connaissance la première application de l'auto-attention pour de l'apprentissage *non supervisé en parole*.

Avec des données simulées, nous confirmons que le modèle proposé peut retrouver exactement les facteurs sous-jacents correspondants à des locuteurs et à des phonèmes. Nous observons qu'en utilisant en entrée seulement des banques de filtres logarithmiques, complexes et de grande dimension, le modèle récupère les facteurs utilisés pour la génération des données, et que deux variables, aux niveaux local et global, sont essentielles pour une reconstruction exacte et une représentation bien découplée.

Sur TIMIT, un corpus de parole lue, en anglais, les biais proposés encouragent les représentations à découpler les locuteurs et les phonèmes, comme le montrent les résultats de classification obtenus en aval à l'aide d'un simple classificateur k-means non supervisé. L'optimisation conjointe de plusieurs variables latentes, avec chacune son biais propre, permet de découpler des facteurs sous-jacents qu'une seule variable ne peut représenter simultanément.

Nous avons exploré quelques-uns des facteurs sous-jacents potentiellement utiles aux applications pour lesquelles peu ou pas de données annotées sont disponibles. L'approche proposée dans cette thèse, qui encourage un modèle génératif à apprendre des représentations interprétables et découplées, ouvre la porte à l'exploration d'autres facteurs et biais inductifs.

Mots-clés : apprentissage non-supervisé, représentation distribuée, traitement de la parole, autoencodeur variationnel

Unsupervised speech representation learning

Gilles BOULIANNE

ABSTRACT

Representations aim to capture significant, high-level information from raw data, most commonly as low-dimensional vectors. When considered as input features for a downstream classification task, they reduce classifier complexity, and help in transfer learning and domain adaptation. An interpretable representation captures underlying meaningful factors, and can be used for understanding data, or to solve tasks that need access to these factors. In natural language processing (NLP), representations such as word or sentence embeddings have recently become important components of most natural language understanding models. They are trained without supervision on very large, unannotated corpora, allowing powerful models that capture semantic relations important in many NLP tasks. In speech processing, deep network-based representations such as bottlenecks and x-vectors have had some success, but are limited to supervised or partly supervised settings where annotations are available and are not optimized to separate underlying factors.

An unsupervised representation for speech, i.e. one that could be trained directly with large amounts of unlabelled speech recordings, would have a major impact on many speech processing tasks. Annotating speech data requires expensive manual transcription and is often a limiting factor, especially for low-resource languages. Disentangling speaker and phonetic variability in the representation would eliminate major nuisance factors for downstream tasks in speech or speaker recognition. But despite this potential, unsupervised representation has received less attention than its supervised counterpart.

In this thesis, we propose a non-supervised generative model that can learn interpretable speech representations. More specifically, we propose several extensions to the variational autoencoder (VAE) model, a unified probabilistic framework which combines generative modelling and deep neural networks. To induce the model to capture and disentangle meaningful underlying factors, we impose biases inspired by articulatory and acoustic theories of speech production.

We first propose time filtering as a bias to induce representations at a different time scale for each latent variable. It allows the model to separate several latent variables along a continuous range of time scale properties, as opposed to binary oppositions or hierarchical factorization that have been previously proposed.

We also show how to impose a multimodal prior to induce discrete latent variables, and present two new tractable VAE loss functions that apply to discrete variables, using expectation-maximization reestimation with matched divergence, and divergence sampling.

In addition, we propose self-attention to add sequence modelling capacity to the VAE model, to our knowledge the first time self-attention is used for learning in an *unsupervised speech* task.

We use simulated data to confirm that the proposed model can accurately recover phonetic and speaker underlying factors. We find that, given only a realistic high-dimensional log filterbank signal, the model is able to accurately recover the generating factors, and that both frame and sequence level variables are essential for accurate reconstruction and well-disentangled representation.

On TIMIT, a corpus of read English speech, the proposed biases yield representations that separate phonetic and speaker information, as evidenced by unsupervised results on downstream phoneme and speaker classification tasks using a simple k-means classifier. Jointly optimizing for multiple latent variables, with a distinct bias for each one, makes it possible to disentangle underlying factors that a single latent variable is not able to capture simultaneously.

We explored some of the underlying factors potentially useful for applications where annotated data is scarce or non-existent. The approach proposed in this thesis, which induces a generative model to learn disentangled and interpretable representations, opens the way for exploration of new factors and inductive biases.

Keywords: unsupervised learning, speech representation, representation learning, variational autoencoder

TABLE OF CONTENTS

Page

INTRODUCTION 1

CHAPTER 1 UNSUPERVISED LEARNING 7
1.1 Unsupervised classification 7
1.1.1 K-means 8
1.1.2 Gaussian Mixture Model 9
1.1.3 Latent Dirichlet Allocation 9
1.2 Weak supervision 10
1.3 Representations 12
1.3.1 Sequences 15
1.4 Unsupervised representations 16
1.5 Summary 19

CHAPTER 2 THE VARIATIONAL AUTOENCODER 21
2.1 Inference in VAE 22
2.1.1 VAE objective 22
2.1.2 Expected lower bound derivation 23
2.1.3 Reconstruction and representation 24
2.1.4 Posterior and prior choice 26
2.1.5 Discrete latent variables 27
2.2 Interpretability and disentanglement 28
2.3 Sequence learning 29
2.4 Application to speech 30

CHAPTER 3 EXTENDING THE VAE MODEL 33
3.1 Introduction 33
3.1.1 Underlying factors in speech 33
3.1.2 Short-term variability: discrete prior 35
3.1.3 Long-term variability: sequence modelling 36
3.1.4 Summary 36
3.2 Conventional VAE 38
3.2.1 Functional diagram 38
3.3 Mixture VAE 39
3.3.1 Expected lower bound 41
3.4 Multimodal VAE 43
3.4.1 E-M reestimation 44
3.4.1.1 Matched KL divergence 45
3.4.2 Divergence sampling 47
3.4.3 Summary for multimodal VAE 48

3.5 Sequence and frame levels 49
3.5.1 Tying and pooling 51
3.5.2 Filtering 52
3.5.3 Attention 55
3.6 Multiple filtered latent variables (MFL) VAE 57
3.7 Preliminary experiments 58
3.8 Summary 61

CHAPTER 4 EXPERIMENTAL RESULTS 63
4.1 Datasets 63
4.2 Model implementation 63
4.3 Methodology 64
4.3.1 Accuracy computation 65
4.3.1.1 Confidence interval estimation 66
4.3.2 Data preparation 66
4.3.3 Training and evaluation 67
4.4 Experiments on simulated vowels 69
4.4.1 Vowel signal generation 69
4.4.2 Synthetic subsets 73
4.4.3 Model parameters 74
4.4.4 Results 75
4.5 Experiments on TIMIT 80
4.5.1 Data preparation 80
4.5.2 Baseline results 81
4.5.3 Experimental results 83
4.5.3.1 Time filtering 84
4.5.3.2 Multiple latent variables 87
4.5.3.3 Self-attention 89
4.5.4 Visualizations 90
4.6 Summary 92

CHAPTER 5 DISCUSSION 95

CONCLUSION AND RECOMMENDATIONS 99

APPENDIX I HYPERPARAMETERS 103

LIST OF REFERENCES 104

LIST OF TABLES

Page

Table 4.1 Formant frequencies of simulated vowels (Hz), step 1 71

Table 4.2 Synthetic vowel datasets 73

Table 4.3 Default parameters for experiments 74

Table 4.4 Multimodal VAE accuracy with and without sequence level variable 76

Table 4.5 Frame-wise accuracy for baseline phoneme and speaker classifiers on TIMIT development and test sets, with filterbank features spliced ±2 81

Table 4.6 Plain VAE and β-VAE representations and the effect of filtering on TIMIT development set, filterbank features spliced ±2 85

Table 4.7 Multiple variables modelling on TIMIT development set, filterbank features spliced ±2 87

Table 4.8 Multiple variables modelling on TIMIT development set, filterbank features spliced ±5 88

Table 4.9 MFL-VAE with self-attention, on TIMIT development set, filterbank features spliced ±5 89

Table 4.10 Phoneme confusion matrix, TIMIT dev set, model 7 from Table 4.9 92

LIST OF FIGURES

Page

Figure 1.1 Representation learning. Top: training, Bottom: use 12

Figure 1.2 Classical autoencoder. Hidden layers in white, bottleneck in grey 17

Figure 2.1 VAE graphical model. Top: generator (decoder), bottom: recognizer (encoder) 21

Figure 2.2 Deeper network's impacts in VAE models. Upper: representation of latent variable. Lower: ground truth (odd columns) and reconstruction samples (even columns) 25

Figure 3.1 VAE functional diagram and closed-form divergence 39

Figure 3.2 Generative model (decoder) for mixture VAE 40

Figure 3.3 Posterior model (encoder) for mixture VAE 41

Figure 3.4 VAE with E-M reestimation 44

Figure 3.5 VAE with sampling divergence 47

Figure 3.6 Generative model (decoder) for N sequences of length L_n. The frame-level variable is z, and the sequence-level variable is s 49

Figure 3.7 Posterior model (encoder) with N sequences of length L_n 50

Figure 3.8 Proposed VAE model architecture with filtering layers. Only latent variable z is illustrated for simplicity 53

Figure 3.9 (a) Single-head attention block, for frame t. (b) Multi-head attention using single-head attention blocks. K, Q, and V mean key, query, value, respectively 56

Figure 3.10 Multiple latent variables model. Filtering layers are not illustrated 58

Figure 3.11 Modelling synthetic vowels with frame and sequence latent variables. Top row, from left to right: original filterbank features, frame-level prior, sequence-level prior. Bottom row: reconstructed signal, samples of frame-level variable, samples from sequence-level variable 60

Figure 4.1 Evaluation of development FPA and FSA for two latent variables 67

Figure 4.2 Observed vowel triangle for eight standard Indonesian speakers 70

Figure 4.3 Steps of the vowel simulation process 70

Figure 4.4 Simulated vowel triangle for 5 randomly chosen speakers, step 3 71

Figure 4.5 Modulated spectrum envelope over 512 FFT bins, step 4 72

Figure 4.6 Simulated spectrograms of 40 mel-filterbank features. Five vowels ordered from left to right, with each vowel produced by 40 speakers, step 6 73

Figure 4.7 Sequential VAE on vowels_4 test set, sequence length 20, accuracy 99.6% 77

Figure 4.8 Sequential VAE on vowels_2, 3, 4, 5 test sets 78

Figure 4.9 Multimodal VAE on vowels_4 test set, no sequence modelling, accuracy 76.8% 79

Figure 4.10 Conventional VAE on vowels_4 test set, no sequence modelling, accuracy 28.5% 79

Figure 4.11 Modelling TIMIT dev set with frame and sequence variables c and s. Top row, from left to right: Mel filterbank features, c prior and s prior. Bottom row, from left to right: reconstructed signal, c samples colored by phoneme, s samples colored by speaker 84

Figure 4.12 Batch FPA and FSA on the TIMIT validation set with varying filter lengths 86

Figure 4.13 t-SNE projection of raw features (left) and s representation. Each point is one frame of 40 utterances from TIMIT development set. Colors represent speakers 90

Figure 4.14 t-SNE projection of raw features (left) and c representation. Each point is one frame of 40 utterances from TIMIT development set. Colors represent phonemes 91

LIST OF ABBREVIATIONS

AAE	Adversarial Auto Encoder
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
BLSTM	Bidirectional Long Short-Term Memory network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
ELBO	Evidence Lower Bound
EM	Expectation Maximization
FFT	Fast Fourier Transform
FHVAE	Factored Hierarchical Variational AutoEncoder
FMLLR	Feature Maximum Likelihood Regression
FPA	Framewise Phoneme Accuracy
FSA	Framewise Speaker Accuracy
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
JFA	Joint Factor Analysis
KL	Kullback-Leibler divergence
LDA	Latent Dirichlet Allocation or Latent Discriminant Analysis

LSTM	Long Short-Term Memory network
MAP	Maximum A Posteriori
MCMC	Monte Carlo Markov Chains
MFCC	Mel Frequency Cepstral Coefficients
MFL	Multiple filtered Latent variables
ML	Maximum Likelihood
MLP	Multilayer Perceptron
MMD	Maximum Mean Discrepancy
NLP	Natural Language Processing
NN	Neural Network
PLDA	Probabilistic Linear Discriminant Analysis
PLP	Perceptual Linear Prediction
RBM	Restricted Boltzmann machine
RNN	Recurrent Neural Network
SAT	Speaker Adaptive Training
SGD	Stochastic Gradient Descent
SVI	Stochastic Variational Inference
SGVB	Stochastic Gradient Variational Bayes
TDNN	Time-Delay Neural Network
UBM	Universal Background Model

VAE	Variational Auto Encoder
VT	Vocal Tract

LIST OF SYMBOLS AND UNITS OF MEASUREMENT

Bern	Bernoulli distribution
Cat	Categorical distribution
Dir	Dirichlet distribution
$\mathbb{E}$	Expectation
$\mathcal{L}_{ELBO}$	Expected lower-bound
D_{KL}	Kullback-Leibler divergence
$\boldsymbol{\mu}$	Mean vector
$\mathcal{N}$	Normal distribution
$\mathcal{R}$	Real number set
$\boldsymbol{\sigma}$	Variance vector

INTRODUCTION

Representations aim to capture significant, high-level information from raw data, most commonly as low-dimensional vectors. When considered as input features for a downstream classification task, they reduce classifier complexity, and help in transfer learning and domain adaptation. An interpretable representation is one which captures underlying meaningful factors, and can be used for understanding data, or to solve tasks that need access to these factors. A disentangled representation is also able to place different underlying factors, for example style and content, into separate vector dimensions, so that even factor combinations that were not seen in training can be well represented.

In natural language processing (NLP), word embeddings such as *word2vec* (Mikolov, Sutskever, Chen, Corrado & Dean, 2013), or sentence embeddings as in *ELMO* (Clark, Lee, Zettlemoyer, Peters, Neumann, Iyyer, Gardner, Clark, Lee & Zettlemoyer, 2018), are examples of representations that have become important components of many natural language understanding models. Two main factors explain this success. Firstly, these representations are trained without supervision, so they can exploit very large, unannotated text corpora, allowing more powerful models than those limited to smaller, annotated corpora. Secondly, they have a meaningful interpretation as their vector space encodes analogies, as in the vector equation $king - queen = man - woman$ (Pennington, Socher & Manning, 2014). This ability to capture underlying relations to some degree is key to their success in a number of tasks where semantics plays an important role.

Representations have also been used in speech processing, where deep neural network based representations such as bottleneck and tandem features (Knill, Gales, Ragni & Rath, 2014) have been used to complement more classical speech features. In speaker recognition (Snyder, Garcia-Romero, Povey & Khudanpur, 2016b) speaker representations are derived from linear layers before the last output layer of a neural network. Speaker embeddings have also been used

for other tasks such as speaker diarization (Garcia-Romero, Snyder, Sell, Povey & McCree, 2017) and speech recognition (Povey, Hadian, Ghahremani, Li & Khudanpur, 2018). However, in contrast with NLP, these speech representations must be trained with supervision, using speech frames labelled by speaker, or with semi-supervision using the text transcription of the speech. Furthermore, in general these approaches do not aim for interpretable representations.

0.1 Problem statement and motivation

An unsupervised representation for speech, i.e., one that could be trained directly with large amounts of unlabelled speech recordings, and could disentangle the main factors underlying speech variability, would have a major impact on many speech processing tasks, for the following reasons.

Labelling of data for speech is done through manual transcription and is very expensive. Transcribing an audio recording into a time-aligned text requires between 40 and 100 hours of work for each hour of recording (Seifart, Evans, Hammarström & Levinson, 2018). Collecting a reasonable amount of labelled data can even be infeasible for some languages where there are simply not enough transcribers available. On the other hand, large collections of unannotated recordings are readily available, even for rare and endangered languages: this disproportion between the ease of collecting audio recordings, and their transcription cost, has been termed the "transcription bottleneck" (Seifart *et al.*, 2018). Leveraging large amounts of untranscribed speech through unsupervised representations would extend speech recognition to tasks that are not currently feasible, for example low-resource languages.

Major sources of variability in speech are often described as phonetic, speaker and channel related, because of their impact on speech recognition (Lippmann, 1997), speaker recognition (Kenny, Boulianne, Ouellet & Dumouchel, 2007b) or emotion detection (Cummins, Epps, Sethu & Krajewski, 2014). These claims are supported by empirical studies of speech variability

in the spectral domain, such as Kajarekar, Malayath & Hermansky (1999), which conclude that phonemes and phonetic context account for 59.7% of total variability, while speaker and channel variability accounts for 40.3%. In speech-to-text, spoken term retrieval, or language recognition, speaker variations are a nuisance factor that degrade performance, while for speaker diarization or text-independent speaker verification, it is the phonetic content which must be ignored or compensated for. These tasks would all benefit from a representation able to disentangle speaker and phonetic variations. Feeding the downstream classifier only the relevant part of the representation frees the classifier from dealing with unrelated variability, allows it to be simpler and reduces its annotated data requirements.

Unsupervised representation learning for speech has received less attention, compared to supervised representations, with most previous work relying on generative models that represent underlying factors of variation, but which do not produce disentangled and interpretable representations. Modified objective functions have been used to encourage disentangling, such as β-VAE (Higgins, Matthey, Pal, Burgess, Glorot, Botvinick, Mohamed & Lerchner, 2017) and mutual information (Chen, Duan, Houthooft, Schulman, Sutskever & Abbeel, 2016; Phuong, Welling, Kushman, Tomioka & Nowozin, 2018).

However, as noted in Locatello, Bauer, Lucic, Rätsch, Gelly, Schölkopf & Bachem (2018), purely unsupervised learning of disentangled representations is not possible without inductive biases on both the model and data. Some recent work (Chorowski, Weiss, Bengio & van den

Oord, 2019; Hsu, Zhang & Glass, 2017b; Li & Mandt, 2018) has used an inductive bias in the form of a binary opposition of frame vs. utterance levels.

0.2 Research objectives and contributions

Given the potential of unsupervised speech representation, and the relatively small amount of related previous work, the objective of this research is to develop an unsupervised model that can learn interpretable and disentangled representations for speech.

The leading approach to unsupervised representation learning uses generative models (Eastwood & Williams, 2018). In our early work, we started with posteriorgram representations based on Gaussian mixture models (GMMs), aiming to capture phonetic content and exclude speaker variability (Boulianne, 2015). However GMMs are not the model of choice when large amounts of data are available, and deep neural networks offer more capacity and diversity of architecture. Here we select the variational autoencoder (VAE) as it combines generative modelling and deep neural networks in a unified framework. The generative model has latent variables that have a natural explicative function. Deep neural networks can model complex non-linear relationships between these latent variables and the observed real-world signals. In this work, we extend the variational autoencoder (VAE) model to an arbitrary number of latent variables, each with its own prior, so we can incorporate inductive biases as prior distributions on these variables. The main contributions of this thesis are:

1. We propose time filtering as a bias to induce representations at a different time scale for each latent variable, on a continuum that is not limited to a binary frame vs. utterance dichotomy;
2. We show how to impose a multimodal prior for discrete latent variables, and present two new tractable VAE loss functions using expectation-maximization with matched divergence, and divergence sampling;

3. Using a realistic simulated vowel dataset, we examine how well time scale and discreteness biases encourage latent variables to recover the underlying vowel and speaker factors used to generate the audio features;
4. On a real English speech database, TIMIT, we evaluate disentangling and interpretability of the proposed representation with downstream phoneme and speaker classification tasks using a simple k-means classifier. Jointly estimating the latent variables is shown to be crucial for better disentangling;
5. We propose self-attention to add sequence modelling capacity to the VAE model. This is the first time self-attention is used for learning in an *unsupervised speech* task.
6. A manuscript was submitted to the IEEE/ACM Transactions on Audio, Speech and Language Processing, with contents mostly taken from Chapter 3 and 4; currently it requires minor revisions for English usage before acceptance.

0.3 Thesis outline

Together, the first two chapters constitute a literature review. Chapter 1 covers unsupervised learning and representations, while Chapter 2 concentrates on the variational autoencoder and the approaches previously proposed in this framework. In Chapter 3, we introduce our proposed model and derivations for multiple latent variables, time-scale biases, multimodal priors, and sequence modelling, along with their deep neural network implementation. Chapter 4 presents our experimental results on simulated data and TIMIT, with a detailed methodology for evaluating interpretability and disentanglement of the representations, using downstream phoneme and speaker classification tasks. We discuss our results in Chapter 5 and conclude in Chapter 6. A detailed list of model hyperparameters, as they were used in the experiments, is provided in Appendix I.

CHAPTER 1

UNSUPERVISED LEARNING

The recent success of deep learning in speech was obtained with supervised methods, which depend on large amounts of *labelled data* in the form of text transcriptions. This dependency is a limiting factor for many applications where large amounts of labelled data are not available. It is difficult to find large labelled corpora for very specialized domains, for example conversations between a control tower and airplane pilots, as was the case in the recently held Airbus Air Traffic Control challenge (Delpech, Laignelet, Pimm, Raynal, Trzos, Arnold & Pronto, 2018). In fact, it is a real problem for a lot of the world's languages, which are qualified as "low-resource languages" because they lack the amount of labelled recordings required to get good performance with current speech models, a problem that the IARPA Babel program (IARPA, 2012) specifically addresses. However, today it is relatively easy to obtain large amounts of *unlabelled* recorded video and speech, even for specialized applications and low-resource languages. Could it be possible to learn from large amounts of unlabelled data, i.e., in an unsupervised manner, to obtain models as powerful as our current supervised models?

This question has generated a large body of literature in machine learning, ever since its beginning. We don't attempt here a full review, but limit ourselves to the main topics which form the supporting background for the material to be introduced in the following chapters. This chapter will present a literature review of three important approaches: unsupervised training, partly supervised training, and representation learning.

1.1 Unsupervised classification

Among the earliest strategies proposed for unsupervised learning we find clustering, whose goal is to discover natural groupings (clusters) in the unlabelled data. Although there exist algorithms which find hierarchies of nested clusters, we will consider here algorithms that partition data without imposing a hierarchical structure.

The most well-known of these partitional algorithms is the classical K-means algorithm which was introduced about 50 years ago (Jain, 2010).

1.1.1 K-means

Given a number N of objects (observations) represented by features of dimension D, K-means clusters objects into K groups such that observations within a group are *closer* than observations across groups. To evaluate closeness, conventional K-means relies on the Euclidean distance measure. More precisely, it groups observations such that the sum of the squared error between the empirical mean of a cluster and the observations in the cluster is minimized. The resulting clustering can be used for classification, data visualization, or as an initialization for more expensive clustering algorithms.

The K-means algorithm has several advantages: it is simple and scalable, easy to implement, and works well for a variety of applications (Kulis & Jordan, 2012). Its time complexity is linear in N, D and K. It also has well-known limitations: it can only detect compact hyperspherical clusters, the number of clusters must be specified in advance, each observation is assigned to a single cluster with a hard decision, and it converges to a local minimum of its objective function.

Several extensions to K-means have been proposed to overcome these limitations: selection of the number of clusters with the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC) (Pelleg & Moore, 2000), use of other distance measures such as Mahalanobis, L1 or Itakura-Saito (Celebi, Kingravi & Vela, 2013), and a large spectrum of initialization methods (Celebi *et al.*, 2013). Spectral clustering for K-means (Zha, He, Ding & Simon, 2001) and kernel K-means (Scholkopf, Smola & Muller, 1996) were introduced mainly to allow more arbitrary shaped clusters, but can be shown to be equivalent (Dhillon, Guan & Kulis, 2005).

Spectral clustering has been used in speech enhancement (Hershey, Chen, Le Roux & Watanabe, 2016). As an unsupervised learning method in speech, K-means has had some early successes (Astrahan, 1970) but it was quickly replaced with a better performing approach, Gaussian Mixture Models (GMMs).

1.1.2 Gaussian Mixture Model

The Gaussian Mixture Model is a generative model: observations are assumed to be sampled from a mixture of gaussian distribution, of which each gaussian component is specified by a mean and a covariance. The generative process is to first sample an index for the component (from a prior distribution of component indices), then sample an observation from the associated gaussian distribution, thus giving rise to a mixture of gaussians. To find which mixture component was used in generating each observation (the per-observation component assignment), so that the overall likelihood of observations is maximized, the most commonly used algorithm is Expectation Maximization (EM). EM has been shown to be a probabilistic generalization of K-means by (Welling, 2009) and (Kulis & Jordan, 2012). Thus clustering with a GMM is closely related to K-means, but differs by replacing hard cluster assignments with probabilities and using a covariance-based distance measure.

In speech, GMMs have been used as part of Hidden Markov Models (HMMs) in the GMM-HMM globally supervised speech recognition model (Deng & Jaitly, 2016). As an unsupervised learning method, their most common use is in speaker recognition as a Universal Background Model (UBM) for i-vector extraction (Dehak, Kenny, Dehak, Dumouchel & Ouellet, 2011).

1.1.3 Latent Dirichlet Allocation

In some applications, the assumption that each observation belong to a single cluster is not realistic. For example, in document classification, it is natural to assume that each document can be labelled with several topics. Mixed-membership models were introduced to relax this assumption, so that each observation may belong partly to several categories, instead of belonging to a single cluster (Airoldi, Blei, Erosheva & Fienberg, 2014).

A very successful mixed-membership model is Latent Dirichlet Allocation (LDA) (Blei & Lafferty, 2009). LDA reshaped the topic modelling community and has become a standard tool in document analysis (Kulis & Jordan, 2012). LDA pushes the idea of latent variables in a generative model to a two-level hierarchy. Each document is assumed to be generated from

multiple topics, where a topic is a particular distribution of words in the vocabulary. The hidden variables are the per-document topic proportions and the per-word topic assignments. These hidden variables have been used as features for language model adaptation in speech recognition (Deena, Hasan, Doulaty, Saz & Hain, 2016).

K-means, GMMs, and LDA introduced the main ideas that still underlie the more recent purely unsupervised learning methods: natural groupings can be found in the data, not at the superficial level of features, but in deeper, latent factors, that can even form a hierarchy.

1.2 Weak supervision

Now we turn to other approaches that seek some help to assist unsupervised learning and fall under the general term of weak supervision. Zhou (2018) recognizes three typical types of weak supervision: incomplete, inexact and inaccurate.

Incomplete supervision (or semi-supervised learning) combines small amounts of explicit classification information from the labelled data with large amounts of implicit information from unlabelled data.

Generative models formulate incomplete supervision as a missing data imputation task for the classification problem (Barber, 2016). In a generative model, the observed data distribution is modelled by a parametric distribution, whose parameters are treated as unobserved random variables. Incomplete supervision is naturally handled by also treating class labels as unobserved random variables. Unobserved label variables are marginalized, and for labelled observations, the corresponding latent variable probability is set to one (if discrete) or to a delta function (when continuous) (Barber, 2016).

Self-training is another example of incomplete supervision. It starts with a supervised classifier that is trained on a small training set, then iteratively retrained on new examples labelled by the current classifier. To avoid "training bias" (badly labelled examples will make the model drift farther), it is common to strictly filter auto-labelled examples with some criteria e.g., active

learning. Self-training is a wrapper method in the sense that it can be applied to any supervised learning algorithm, but it needs good confidence predictions in order to work (Tanha, van Someren & Afsarmanesh, 2017). Co-training takes the same approach with two supervised classifiers that provide each other labels for the unlabelled instances.

Some unsupervised deep learning models allow incorporation of explicitly labelled examples in their training set, and this has been applied with the Variational Autoencoder (Kingma & Welling, 2014) and Adversarial Autoencoder (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville & Bengio, 2014), which will be described in more details in Chapter 2.

In transfer learning, unsupervised models trained on an auxiliary task are used to train supervised models on another task, as an initialization for deep supervised networks e.g., Boltzmann machines (Hsu, Hwang, Wu, Tsao & Wang, 2017a)), or to simulate additional labelled training data (Dai & Le, 2015). Representation learning, a special case of transfer learning, has become so important recently that it is deferred to next section wholly devoted to it.

Inexact supervision corresponds to the case where supervision is available only indirectly. In multi-instance learning, labels are not available for individual instances, but only for groups of instances (Foulds & Frank, 2010). Even when no labels at all are available, groupings themselves are informative: siamese networks (Gundogdu & Saraclar, 2017; Zeghidour, Synnaeve, Usunier, Dupoux & Etudes, 2016) can exploit knowledge about speech observation in word pairs. Similarly, observation triples are used in triplet loss (Bredin, 2017), or even larger groupings that partition the data in groups with common factors of variation (Rudolph, Ruiz, Athey & Blei, 2017).

Finally, **inaccurate supervision** concerns situations where labels may contain errors or noise, as when labels are obtained from crowdsourcing. One approach is to identify potentially mislabelled instances and correct them, as in data editing (Muhlenbach, Lallich & Zighed, 2004).

1.3 Representations

Input features are an important part of classifier performance, and a lot of research efforts have been dedicated to feature design and tuning (feature engineering). While any features extracted from data can be thought of as a form of data representation, what is usually meant in the context of "representation learning" is a more complex transformation, learned in a supervised or unsupervised way, which transforms low-level input features into higher-level features. For example, consider a deep neural network model with many hidden layers trained for a task, the auxiliary task in Figure 1.1. As we go from input to output, each layer is a more and more complex transformation of the input. Since each layer is the sole input for the following layers, it represents *all of the information* that the network can use to perform the task. Thus each hidden layer in the deep network can potentially play the role of a representation, in the form of a low dimensional vector of real numbers. Typically a linear layer close to the output layer is chosen, like layer h_3 at top of Figure 1.1.

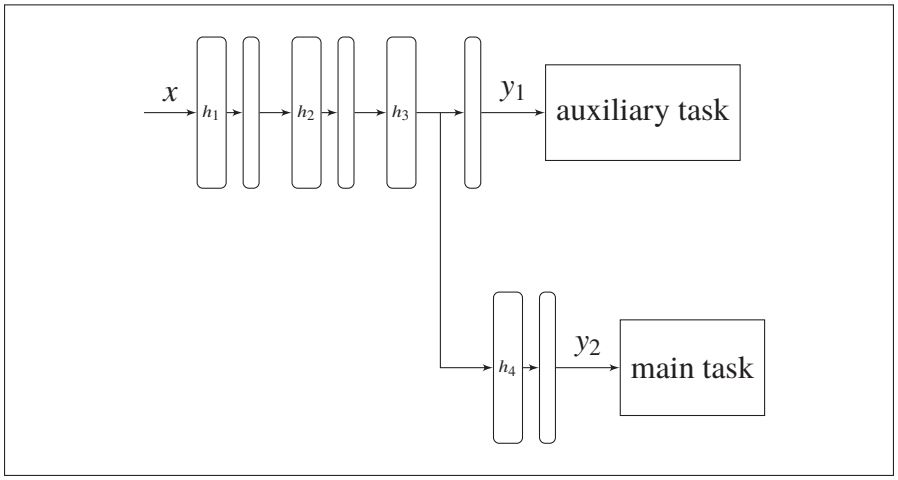


Figure 1.1 Representation learning. Top: training, Bottom: use

Such learned feature transformations are also known as *embeddings*, or *distributed representations* (as opposed to local representations such as clustering, which classify observations in one of K classes) (Bengio, Courville & Vincent, 2013). Representations typically have smaller dimensions than input features, capture a lot of the information / redundancy, and eliminate

feature engineering. They are claimed to lead to better performance, make learning easier with reduced training time, require smaller number of examples, offer better convergence and overcome classifier limitations, as argued in Bengio *et al.* (2013); Le, Ranzato, Monga, Devin, Chen, Dean, Corrado & Ng (2012).

The idea of representation learning can be traced backed to a long history in machine learning. As seen in the previous section, spectral clustering and kernel K-means attempt to find more complex-shaped clusters in the feature space by operating in a simpler, underlying space. Good representations learned from data can be used for various tasks such as data visualization, summarization and exploration (Ridgeway, 2016), or as an input for a subsequent step of classification, for example in generative modelling and semi-supervised learning (Kingma, Rezende, Mohamed & Welling, 2014; Maaløe, Sønderby, Sønderby & Winther, 2017b; Rezende, Mohamed & Wierstra, 2014).

In feature transfer, a representation learned to perform one task is used for another, different task. In Figure 1.1, a first deep neural network is trained on an auxiliary task (top), and the representation learned is then used to train a second, shallow supervised classifier on the main task (bottom).

This special case of transfer learning (Lacoste, Oreshkin, Chung, Boquet, Rostamzadeh & Krueger, 2018) is useful when only small amounts of training data are available for the main task, but an auxiliary task with large amounts of training data is available.

Recently, this approach has had a large success in many fields, to the point where representation learning has become a field in itself in the machine learning community (Bengio *et al.*, 2013; Clark *et al.*, 2018).

For example, in image recognition, features trained on ImageNet were shown to generalize to other tasks (Donahue, Jia, Vinyals, Hoffman, Zhang, Tzeng & Darrell, 2013). In this case, the auxiliary task is supervised, i.e. a large set of one million images with labels. Feature transfer

was shown to surpass fine-tuning of a pretrained network for image classification (Mahajan, Girshick, Ramanathan, He, Paluriixuan, Li, Bharambe & van der Maaten, 2018).

In natural language processing (NLP), word embeddings have obtained similar successes: *GloVe* (Pennington *et al.*, 2014), *FastText* (Bojanowski, Grave, Joulin & Mikolov, 2016), *ELMO* (Clark *et al.*, 2018), and the most widely used *word2vec* developed by Mikolov *et al.* (2013). These representations are obtained by training a recurrent neural network with large amounts of text, to predict the next word given its current hidden state. Note that in this case, the auxiliary task is *unsupervised*. A softmax activation output layer encodes the predicted probability for each word in the vocabulary. The linear layer just before the output activation is taken as the low-dimensional representation of a word. In the more recent approaches (such as ELMO), the embeddings represent whole sequences of words rather than a single one (Clark *et al.*, 2018).

In speech recognition, bottleneck and tandem features (Knill *et al.*, 2014) are used to replace, or to supplement, respectively, more classical features such as MFCC and PLP. Posteriorgrams have been obtained from supervised phonetic models for spoken term detection (Hazen, Shen & White, 2009), from Deep Neural Networks (DNNs) for mispronunciation detection (Lee, Zhang & Glass, 2013), or siamese networks trained on text-audio pairs for keyword search (Gundogdu & Saraclar, 2017).

In speaker recognition, the auxiliary task is usually speaker classification (Bhattacharya, Alam & Kenny, 2017; Li, Chen, Shi, Tang & Wang, 2017; Snyder *et al.*, 2016b). In that case, representations are derived from linear layers located before the last output layer. Such speaker representations followed by a simple classifier offer comparable or better performance than state-of-the-art i-vector/PLDA systems (Wang, Li, Tang & Zheng, 2017). Speaker embeddings have also been used for other tasks such as speaker diarization (Garcia-Romero *et al.*, 2017; Rouvier, Bousquet & Favre, 2015), the latter author using super-vectors as input features.

1.3.1 Sequences

Speech is a time series, and most tasks in speech processing can be described as translating from a sequence of frames representing the speech time-series to another sequence of words or classes. For speaker recognition, emotion detection, or language identification, an ideal representation would capture information about a variable-length input sequence in a single, fixed-dimension vector.

Such a representation is obtained with Joint Factor Analysis (JFA) or i-vectors, which convert a whole sequence of speech frames to first-order statistics and derive a low-dimension vector (Dehak *et al.*, 2011). A recurrent neural network (RNN or LSTM) can also be applied to this problem: the history from last recurrent layer is a summary of the whole input sequence (Weiss, Chorowski, Jaitly, Wu & Chen, 2017). Convolutional neural networks (CNN or TDNN) can also successively reduce a varying-length input to a fixed dimension (Cyrta, Trzciński & Stokowiec, 2018; Hsu & Glass, 2018b). These methods preserve temporal ordering information. When temporal order can be ignored, one can combine outputs of several frame-level hidden layers with a simple average, or max-pooling, or use an attention model (Bhattacharya *et al.*, 2017), where the optimal weight for each time-step is learned.

Recently, self-attention has been proposed as a different mechanism that can take into account large contexts and produce state-of-the-art results, but is much faster to train than recurrent networks (Chowdhury, Wang, Moreno & Wan, 2018; Sperber, Niehues, Neubig, Stüker & Waibel, 2018). In the seminal "Transformer" paper by Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser & Polosukhin (2017), self-attention was used as a supervised learning method, given pairs of sentences in source and target language. Training is done by predicting future target words given past source words, with an encoder/decoder tandem. Training can also be framed as an unsupervised task, if input and output sentences are the same, for learning a language model, for example (Al-Rfou, Choe, Constant, Guo & Jones, 2019).

Attention was historically introduced for supervised learning in text processing, and has been since applied to speech recognition and speaker recognition, but always in supervised settings

(Chorowski, Bahdanau, Serdyuk, Cho & Bengio, 2015; Chowdhury *et al.*, 2018; Povey *et al.*, 2018; Sperber *et al.*, 2018; Wang, Okabe, Lee, Yamamoto & Koshinaka, 2019; Zeinali, Burget, Rohdin, Stafylakis & Cernocky, 2019; Zhu, Ko, Snyder, Mak & Povey, 2018).

1.4 Unsupervised representations

Most examples we have seen up to now learn a representation through an auxiliary supervised task. Unsupervised representations are based on models that can be trained without labelled data, such as generative models. Today, a significant fraction of unsupervised representation learning research is driven by generative modelling (Chen *et al.*, 2016; Rouvier *et al.*, 2015).

Generative models explicitly incorporate simple latent variables which generate observations with complicated distributions in feature space. Their goal is to learn a posterior latent variable distribution that explains the observed data well (Bengio *et al.*, 2013); latent variables potentially constitute an ideal representation. Generative modelling has become one of the leading approaches to unsupervised representation learning, with several recent works imposing additional learning constraints to encourage the model to learn disentangled representations (Eastwood & Williams, 2018). Today, the most commonly used models for this task are GMMs, autoencoders and variational autoencoders.

For GMMs, the vector of posterior probabilities, which represents the estimated responsibility of each component, is also called a "posteriorgram". Posteriorgrams have been used in spoken content retrieval (Zhang & Glass, 2009), zero-resource speech recognition (Muscariello, Gravier & Al, 2011), and passphrase verification (Boulianne, 2015). Some approaches treat the GMM supervector[1], estimated on a frame sequence, as input features for a downstream task, such as learning a representation for language identification (Zhang & Hansen, 2018), or speaker diarization (Rouvier *et al.*, 2015).

In speaker recognition, the i-vector (Dehak *et al.*, 2011) is another representation derived from a GMM and has been the mainstream approach up to now. This generative model assumes that

[1] Concatenation of the component means of a GMM obtained by adaptation of a UBM.

the observed supervector for each utterance is sampled from a supervector distribution, whose mean depends on an utterance-specific, low-dimensional, latent vector called an i-vector. The model was derived from JFA (Kenny, Boulianne, Ouellet & Dumouchel, 2007a) by ignoring the speaker label supervision. Each i-vector summarizes a whole speech sequence.

Another main unsupervised approach is the classical autoencoder (Hinton & Salakhutdinov, 2006). It is a deep neural network whose objective is to reconstruct its output x' from its input x, as illustrated in Figure 1.2. The reconstruction task acts as a weak supervision, similar in spirit to the word prediction task in *word2vec*. When $x' = x$ (output is same as input), a narrow hidden layer (bottleneck layer) is used to avoid the trivial solution of just copying the input, and provides a low-dimensional, high-level representation of the input. Other solutions include making x' a noisy version of x, as in the denoising autoencoder (Vincent, Larochelle, Bengio & Manzagol, 2008), or adding a sparsity cost to the objective (Le *et al.*, 2012), or using for x' a different observation from the same class as x (Zhang & Hansen, 2018). Note that in the classical autoencoder, the input, model parameters and reconstruction costs are all deterministic values. In speech emotion recognition Deng, Xu, Zhang, Fruhholz & Schuller (2018) demonstrated with an autoencoder the importance of reconstruction error as a regularizer when learning with few labelled examples.

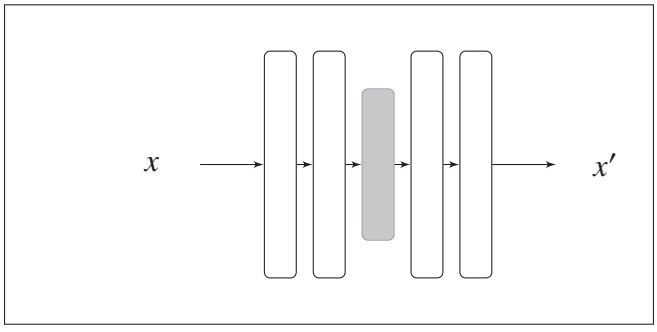


Figure 1.2 Classical autoencoder. Hidden layers in white, bottleneck in grey

A third approach is represented by the variational autoencoder (VAE) of Kingma & Welling (2014). Despite its underlying similarity with the denoising autoencoder (Rezende *et al.*,

2014), the VAE has a very different objective, based on the variational principle: reduce the Kullback-Leibler (KL) divergence between an estimated posterior distribution and true posterior distribution (Bishop, 2006). Deep neural networks with non-linearities are used to define complex distributions from simpler, random variables. Stochastic gradient descent is used to find model parameters that optimize the variational objective cost. With auto-differentiation, there is no need to derive gradient equations. These methods expand the range of distributions that can be handled beyond simple conjugate priors, as previous Bayesian models were, and importantly, scale to large datasets (Blaauw & Bonada, 2016).

Following introduction of the VAE, many similar generative models have been introduced with various objective functions. The Generative Adversarial Networks (GAN) (Goodfellow *et al.*, 2014) keep the generative network of an autoencoder but replace the recognition network with a discriminator that distinguishes between real and generated samples; GANs are optimized for generative tasks and bypass inference of latent variables (Hsu *et al.*, 2017b). In order to infer latent variables, the Adversarial AutoEncoder (AAE) (Makhzani, Shlens, Jaitly, Goodfellow & Frey, 2015) keeps both recognition and generative networks, but adds a third network whose role is to distinguish between samples generated by the model and true samples; the network is trained with a dual objective that combines reconstruction error and an adversarial criterion. As shown in Makhzani *et al.* (2015), AAE and VAE only differ on the specific objective function that is used to match the posterior to the prior. Deep Latent Gaussian Mixtures (DLGMM), introduced by Nalisnick, Hertel & Smyth (2016), add a Dirichlet process prior and a differentiable approximation to the discrete posterior to solve difficulties in mixture model inference.

A Restricted Boltzmann machine (RBM) is another unsupervised network, with a single layer. Stacking individual RBM layers yields the Deep Belief Network (DBN) (Bengio, Lamblin, Popovici & Larochelle, 2006). This undirected probabilistic model is limited to binary variables (Blaauw & Bonada, 2016) and is trained layer-by-layer with Contrastive divergence. Its use has been limited to initialize deep supervised models. In contrast, the VAE has continuous latent variables, in an easier to interpret Bayesian framework, with a simple approximation to true likelihood.

1.5 Summary

Unsupervised learning has been pursued in the field of machine learning since its beginning, with methods such as K-means and GMMs. Representation learning with deep neural networks has generated growing interest; it allows unsupervised classification and transfer learning from large amounts of unlabelled data to a low-resource supervised classification task. Generative models inherently support unsupervised or partly supervised learning. Recently, VAEs have emerged at the intersection of both fields, integrating generative and deep learning in a unified framework, and have become the model of choice for unsupervised representation learning.

CHAPTER 2

THE VARIATIONAL AUTOENCODER

The literature on VAE has literally exploded in recent years[1]. Hence this chapter is devoted to an overview of previous work and current directions in VAE research. A detailed technical description is also given to make it easier to follow developments in the literature.

The classic VAE was first introduced by Kingma & Welling (2014) and Rezende *et al.* (2014). It has the general structure illustrated by the graphical model shown in Figure 2.1.

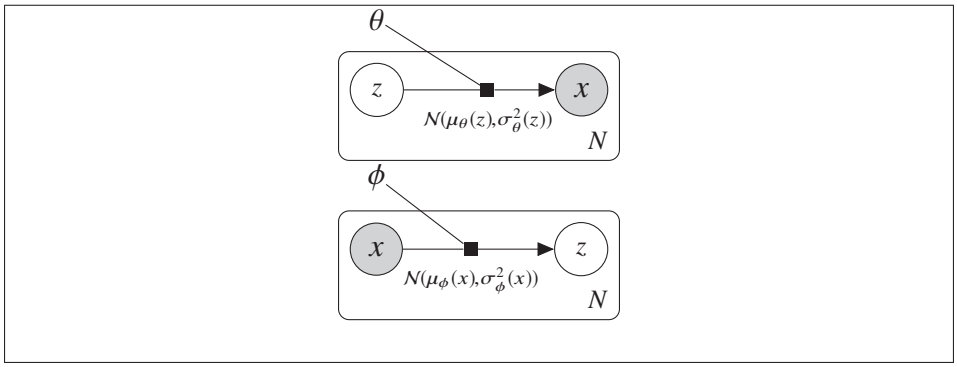


Figure 2.1 VAE graphical model. Top: generator (decoder), bottom: recognizer (encoder)

The top part shows observation x generated from latent variable z so that $p_\theta(x, z) = p_\theta(x|z)p(z)$. The parametric distribution $p_\theta(x|z)$ is modeled with a Gaussian distribution $\mathcal{N}(\mu_\theta(z), \sigma^2_\theta(z))$ where μ_θ and σ_θ are non-linear functions specified by a deep neural network, with parameters θ, applied to z. This represents the generative part of the model, also called the decoder, since it uses the code z to produce a distribution for values of x.

As for any generative model, the aim is to estimate the true posterior distribution $p_\theta(z|x)$. The VAE approximates this true posterior with a parametric posterior distribution $q_\phi(z|x)$. The bottom of Figure 2.1 shows how this approximate posterior is specified by $\mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x))$,

[1] In March 2019, Google Scholar reported approximately 5600 papers with "variational autoencoder" in the title, of which about 2600 were published just since the beginning of 2018.

where μ_ϕ and σ_ϕ are computed by another deep neural network, with parameters ϕ, operating on x. This is the recognition part of the model, also called the encoder, since for each data point x it produces a distribution of possible codes z that could have generated x.

2.1 Inference in VAE

Estimating the posterior distribution in Bayesian models have traditionally relied on methods that are computationally expensive (Blei, Kucukelbir & McAuliffe, 2016) such as Monte Carlo Markov Chains (MCMC) and Gibbs sampling. Variational mean-field approximation turns this inference problem into an optimization problem and is much less expensive (Barber, 2016). Both MCMC and mean-field inference can only be applied to relatively small datasets. Stochastic variational inference (SVI) (Hoffman, Blei, Wang & Paisley, 2013) is able to follow noisy gradient estimates obtained from minibatches and scales to large datasets. Since most Bayesian models have as many local parameters as observations, the inference cost is still linear with respect to dataset size.

The variational objective of the VAE is fully differentiable and thus amenable to gradient-based optimization. Furthermore, parameters to be estimated, θ and ϕ, are global (shared for all observations) rather than local. Exploiting this property on top of SVI Kingma & Welling (2014) proposed Stochastic Gradient Variational Bayes (SGVB) for VAE inference, amortizing inference cost over the dataset, and using a reparameterization trick to reduce variance of the loss estimate. With these improvements, VAE inference becomes scalable to very large datasets.

2.1.1 VAE objective

Here we give a full derivation of the model, following Huszár (2017) and Kingma & Welling (2014).

We have N i.i.d. observations $X = \{x_1, \cdots, x_N\}$ and we assume each observation is generated by a two-step process involving an unobserved continuous variable z. First, a value z_n is generated from a prior distribution $p(z)$; then, an observation x_n is generated from a conditional distribution

$p_\theta(x|z)$. The parameters θ and values of the latent variable z_n are not observable but we want to estimate them. In the general case, the model $p_\theta(x)$ and true posterior $p_\theta(z|x)$ are intractable; therefore, we introduce $q_\phi(z|x)$, a parametric approximation to the true posterior.

2.1.2 Expected lower bound derivation

The model evidence (or marginal likelihood) is the product of the marginal likelihoods of individual observations x_n, so that:

$$\log p_\theta(X) = \log p_\theta(x_1, \cdots, x_N) = \sum_{n=1}^{N} \log p_\theta(x_n) \tag{2.1}$$

In variational inference, each of these individual likelihoods is decomposed into the following sum (see for example eq. (10.2) in Bishop (2006)):

$$\log p_\theta(x) = \mathcal{L}_{ELBO}(\theta, \phi; x) + \mathrm{D}_{KL}[q_\phi(z|x) \parallel p_\theta(z|x)] \tag{2.2}$$

where

$$\mathcal{L}_{ELBO}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)] \tag{2.3}$$

The index n has been dropped and will not be used in what follows unless needed. In that case, just recall that on the whole observation set X, we have:

$$\mathcal{L}_{ELBO}(\theta, \phi; X) = \mathbb{E}_{p_{data}(x)}\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)] \tag{2.4}$$

ELBO stands for "Evidence Lower Bound": since KL divergence is positive, $\mathcal{L}_{ELBO}$ is a lower bound to $p_\theta(x)$, and is maximized when the approximate posterior $q_\phi(z|x)$ exactly matches the true posterior $p_\theta(z|x)$, making the KL divergence vanish.

Using the generative model $p_\theta(x, z) = p_\theta(x|z)\, p(z)$, $\mathcal{L}_{ELBO}$ can be rewritten:

$$\mathcal{L}_{ELBO}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)] \tag{2.5}$$

Grouping posterior $q_\phi(z|x)$ with corresponding prior $p(z)$ and using the definition of KL divergence as $-\mathrm{D}_{KL}(q \parallel p) = -\mathbb{E}_q(\log p - \log q)$, $\mathcal{L}_{ELBO}$ can be reduced to an expression which contains only tractable values:

$$\mathcal{L}_{ELBO}(\theta, \phi; x) \;=\; \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \;-\; \mathrm{D}_{KL}[q_\phi(z|x) \parallel p(z)] \tag{2.6}$$

The right-hand side contains two terms: the first is a reconstruction term and the second is a regularization term that we will call z-prior term.

The reconstruction term is an expectation that can be approximated by Monte Carlo sampling (Anderson, 1999):

$$\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) \approx \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(x \mid z_l) \tag{2.7}$$

with L samples $z_l \sim q_\phi(z|x)$.

The z-prior term can be efficiently computed using the closed-form definition of D_{KL}, i.e., directly from the parameters of $q_\phi(z|x)$ and $p(z)$ rather than from sampled values.

2.1.3 Reconstruction and representation

First VAEs used shallow networks that showed good performance on image datasets such as MNIST (Kingma *et al.*, 2014), so improvements were expected with deeper networks to generate more complex distributions. Instead, it can be observed (Zheng, Yao, Zhang & Tsang, 2018) that a deeper decoder leads to better reconstruction, but produces latent variable representations which are less meaningful and more difficult to classify, as can be seen comparing the first two models on the left of Figure 2.2.

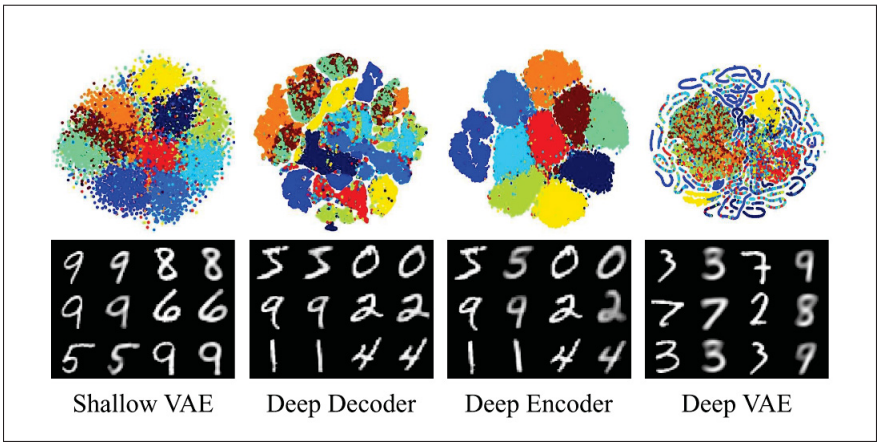


Figure 2.2 Deeper network's impacts in VAE models.
Upper: representation of latent variable. Lower: ground truth (odd columns) and reconstruction samples (even columns)
Taken from Zheng *et al.* (2018, p. 1)

On the other hand, a deeper encoder produces better representations but worse reconstruction quality (third model in Figure 2.2). And when both networks are deep, both reconstruction and representation are worse than for a shallow VAE (rightmost model of Figure 2.2).

In fact Chen, Kingma, Salimans, Duan, Dhariwal, Schulman, Sutskever & Abbeel (2017) show that if a powerful enough decoder is used, the VAE objective will favor the degenerate solution $q_\phi(z|x) = p(z)$, where the latent variable is not used at all.

Several attempts have been made to overcome this degeneration: annealing the KL term (Sønderby, Raiko, Maaløe, Sønderby & Winther, 2016), using free bits (Kingma, Salimans, Jozefowicz, Chen, Sutskever & Welling, 2016), Cluster-aware Generative Models (Maaløe, Fraccaro & Winther, 2017a), or using skip connections in the neural networks (Zheng *et al.*, 2018).

To balance between reconstruction error and representation quality, explicit control of the amount of information captured in the latent code was explored, by weighting the KL term in β-VAE (Higgins *et al.*, 2017) or using a mutual information constraint in the Mutual autoencoder (Phuong *et al.*, 2018). Recently, the VAE objective of Equation (3.1) has been decomposed in up to six sub-terms (Esmaeili, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy & van de Meent, 2018); by weighting each sub-term's contribution to the total objective,

one can control independently several properties of the model, such as disentanglement, reconstruction quality, or the tendency to ignore the latent code.

An alternative approach to avoid the $q_\phi(z|x) = p(z)$ degeneration is to use a more flexible prior $p(z)$ (Section 2.1.4).

2.1.4 Posterior and prior choice

The ability of a VAE to handle non-conjugate priors and arbitrary non-linear mappings of random variables distributions leaves open many design choices.

The VAE approximate posterior $q_\phi(z|x)$ is usually chosen as a product of diagonal Gaussian distributions (mean-field approximation). This variational approximation is one of two components at the origin of the *inference gap* (Cremer, Li & Duvenaud, 2018), i.e. the difference between $\mathcal{L}_{ELBO}$ and the true log posterior. The other component is the amortization gap, which results from global parameters estimated over the whole training, rather than local parameters optimized for each training example individually.

The variational gap is addressed by improvements in expressiveness of approximate posteriors: normalizing flow (Rezende & Mohamed, 2015) and inverse autoregressive flow (Kingma *et al.*, 2016). These methods open the door to arbitrarily complex posterior densities; in particular, inverse autoregressive flow is well suited to high-dimensional latent variables.

The prior $p(z)$ for VAE is usually chosen as a simple Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, based on the assumption that the latent variables should have a simple distribution. This could also drive the solution towards $q_\phi(z|x) = p(z)$ by over-regularization through the KL term in the $\mathcal{L}_{ELBO}$, Equation (3.1). To avoid that, more flexible prior distributions have been proposed, such as mixtures of Gaussians (Dilokthanakul, Mediano, Garnelo, Lee, Salimbeni, Arulkumaran & Shanahan, 2017; Jiang, Zheng, Tan, Tang & Zhou, 2017), multimodal priors coupled to the variational posterior (VampPrior) (Tomczak & Welling, 2018), nonparametric stick-breaking distributions (Nalisnick & Smyth, 2017a), hyperspherical priors (Davidson, Falorsi, De Cao, Kipf & Tomczak,

2018), automatically derived reference priors (Nalisnick & Smyth, 2017b), flexible priors obtained from generator inversion (Kilcher, Lucchi & Hofmann, 2017), or the variational lossy autoencoder (VLAE) using an autoregressive prior (Chen *et al.*, 2017). Such priors yield better representations not only on datasets which have a latent structure matching the prior, but also on those which do not have an obvious matching underlying structure (Davidson *et al.*, 2018).

2.1.5 Discrete latent variables

Some tasks are naturally expressed by discrete variables with a categorical distribution[2]. A classic example is the MNIST digit classification (Lecun, Bottou, Bengio & Haffner, 1998), in which an image obviously represents one of ten discrete classes (digits 0 to 9), with continuous variations in style, width, etc. Clustering is another important problem described by categorical variables.

Several efforts have attempted to incorporate discrete latent variables into the VAE framework. However, inference with discrete variables is prone to several well-known problems in mixture models, such as identifiability (symmetry breaking) and difficulty of mixing (Barber, 2016). Discrete distributions also preclude use of the reparametrization trick, and of gradient descent, because these apply only to differentiable operations.

Nevertheless, a VAE directly modelling discrete latent variables with multimodal priors was proposed by Makhzani *et al.* (2015) using an Adversarial VAE; however, the modes found do not correspond well to natural clusters in the dataset (Rolfe, 2017). Other approaches use a smoothed RBM (to handle binary variables) coupled with a hierarchical approximate posterior (Rolfe, 2017), or marginalize the discrete latent variables, as in Cluster-aware Generative Models (Maaløe *et al.*, 2017a).

Another approach, relaxation, replaces discrete variables with continuous variables that have approximately equivalent distributions. The Gumbel-softmax distribution (Jang, Gu & Poole, 2017) or the Concrete distribution (Maddison, Mnih & Teh, 2017) are continuous distributions

[2] Which describes possible results of a random variable that can take one of K possible values.

that approximate a categorical distribution more closely as their temperature parameter is made smaller[3]. However, in the context of VAEs, care must be taken since the approximation introduces an additional mismatch between the true and approximate posteriors (Aitchison, Adam & Turaga, 2018). In addition, relaxation introduces a large number of local minimums and makes the optimization problem much harder unless explicitly encouraged to find a more global minimum (Murray & Ng, 2010).

Finally, combining a Dirichlet process prior and a Kumaraswamy approximation to the Dirichlet marginal yields a fully differentiable approximation which produces better latent representations on MNIST than the single Gaussian prior, as shown with Deep Latent Gaussian Mixtures (DLGMM) (Nalisnick *et al.*, 2016).

2.2 Interpretability and disentanglement

An interpretable representation is one which captures underlying meaningful factors, and can be used for understanding data, or to solve tasks that need access to these factors. In NLP, word or sentence embeddings have shown their ability to capture meaningful relations between words, as in the vector equation $king - queen = man - woman$ (Pennington *et al.*, 2014).

A disentangled representation is also able to place different underlying factors, for example style and content, into separate vector dimensions, so that even factor combinations that were not seen in training can be well represented. For example, a pink elephant is easily pictured by a human even if he has never seen such a combination. Feeding the downstream classifier only the relevant part of a disentangled representation frees the classifier from dealing with unrelated variability, allows it to be simpler and reduces its annotated data requirements.

Interpretability can be imposed with partial or weak supervision in several ways: semi-supervision (Kingma *et al.*, 2014), weak supervision (Siddharth, Paige, van de Meent, Desmaison, Goodman, Kohli, Wood & Torr, 2017), triplet supervision (Karaletsos, Belongie & Rätsch,

[3] This approach is implemented in Tensorflow through the RelaxedOneHotCategorical distribution.

2016), paired occurrences (Rudolph, Ruiz, Mandt & Blei, 2016), or grouping data according to an implicit task rather than labelling samples (Bouchacourt, 2017).

The VAE has the potential to learn disentangled and interpretable representations, since it explains observations with a hierarchy of stochastic latent variables and explicit dependencies, from local to global features. Thus *fully unsupervised* interpretability has been attempted with various modifications of the VAE objective, as discussed in Section 2.1.3, including β-VAE (Higgins *et al.*, 2017) and mutual information (Chen *et al.*, 2016; Phuong *et al.*, 2018).

However, as noted in Locatello *et al.* (2018), purely unsupervised learning of disentangled representations is not possible without inductive biases on both the model and data. A clustering example can be used to illustrate the problem (Jain, 2010): when classifying animals, different groupings of mammals vs birds, or predators vs non-predators, are both equally valid and uncover meaningful structures, but the choice depends on the end goal of the user. So inductive bias can force the unsupervised representation to select factors of interest among all the possible ones.

In more recent work, some approaches explicitly use the sequence versus observation level distinction as an inductive bias, and model the distinction with latent variables, such as the Factorized Hierarchical Variational AutoEncoder (FHVAE) (Hsu & Glass, 2018b; Hsu *et al.*, 2017b) and the Sequential AutoEncoder (Li & Mandt, 2018). FHVAE uses a hierarchical prior where the sequence level variable is allowed to capture sequence level information, while the other latent variable encodes other residual information.

2.3 Sequence learning

Solutions presented in the previous chapter that derive representations for sequences can also be applied for the VAE: averaging, max-pooling, attention. For example Tan & Sim (2017) use max pooling over frame-level hidden layer activations to obtain parameters for an utterance-level representation variable.

A more Bayesian approach is to consider each sequence as a context, a dataset or a task. The Neural statistician (Edwards & Storkey, 2017) collapses samples in a group with a pooling operation. Similarly, ML-VAE (Bouchacourt, 2017), SVAE (Johnson, Duvenaud, Wiltschko, Datta & Adams, 2016) have group-level and observation-level variables. Another solution proposed for tasks (Lacoste *et al.*, 2018) also applies to sequences: a posterior which factorizes independently over all tasks, reducing the joint ELBO to a sum of ELBO. Several works explicitly capture the sequence versus observation level distinction in latent variables, such as the FHVAE (Hsu & Glass, 2018b; Hsu *et al.*, 2017b) and the sequential autoencoder (Li & Mandt, 2018).

Using sequence capable networks in the VAE, such as recurrent or convolutional networks, has also been proposed. In NLP, generative modelling of text with a variational encoder has been tried with RNN (Bowman, Vilnis, Vinyals, Dai, Jozefowicz & Bengio, 2016), or Dilated CNN (Yang, Hu, Salakhutdinov & Berg-Kirkpatrick, 2017). Hsu, Zhang & Glass (2017d) model speech with an encoder/decoder of 2-layer LSTMs : the latent variable z is obtained from a fully connected Gaussian parameter layer which uses both LSTM layer outputs concatenated as its input. z is then input to a 2-layer LSTM which generates the output sequence. In another work Hsu, Zhang & Glass (2017c) used convolutional layers to successively reduce a variable-length input sequence to a fixed dimension which forms the input to the parameter generating network.

2.4 Application to speech

Publications that applied VAE on speech were not so frequent up to and including 2018. Among them, we find:

- voice reconstruction and interpolation in the latent space (Blaauw & Bonada, 2016) in a non-sequential way (independent frames);
- speech generation and transformation (Hsu *et al.*, 2017a);
- speech synthesis (Henter, Wang & Yamagishi, 2018);
- voice conversion (Chou, Yeh, Lee & Lee, 2018; Hsu *et al.*, 2017c);

- speech emotion classification (Latif, Rana, Qadir & Epps, 2017);
- speaker verification with VAE using i-vectors as input features and sequences modelled by pooling representations (Villalba, Brümmer & Dehak, 2017);
- S-vectors (Hsu *et al.*, 2017d) obtained with a hierarchical factorization into sequence-level and utterance-level latent variables;
- speech recognition with VAE to derive features added to conventional ones in training a DNN speech recognizer (Tan & Sim, 2017);
- unsupervised domain adaptation for speech recognition by augmenting training data with VAE generated data for out-of-domain speech (Hsu *et al.*, 2017d; Hsu, Tang & Glass, 2018), and distant speech (Hsu & Glass, 2018a).

CHAPTER 3

EXTENDING THE VAE MODEL

3.1 Introduction

Previous chapters have presented the general theory behind variational autoencoding and the large number of variants that have been proposed to improve its perceived weaknesses. Now we turn to the question of how to apply VAE to unsupervised representation learning *for speech*. What latent variables should we postulate, what kinds of distributions should we select for the prior and posterior, and how should we formulate the objective function? The previous chapters have shown how flexible VAE can be and the large number of possible designs. Fortunately, research in speech has already explored a large spectrum of modelling approaches, and in the next section we will be guided by this prior knowledge in our modelling choices.

3.1.1 Underlying factors in speech

As mentioned in the introduction, major sources of variability in speech are often described as phonetic, speaker and channel related, because of their impact on various speech tasks, and phonemes and phonetic content account for most of the total spectral variability.

Since phonemes are the main source of variability and are discrete information-bearing units in speech, discreteness would be a natural candidate for main underlying factors in speech. Indirect evidence for this also comes from the human categorical perception of phonemes, in which discrete labels are perceived even though the stimuli range on a continuous scale (Liberman, Harris, Howard & Griffith, 1957). We can also hypothesize that underlying factors are low-dimensional: speech production models are limited to a few free parameters offered by physiological articulators such as tongue, lips, jaw, and larynx.

Finally, phonetic underlying factors are short-term. We can roughly estimate the temporal scale of phonetic variations. At usual rates, speech contains from 10 to 20 phones per second,

which corresponds to 50 ms to 100 ms duration. In spectral analysis, speech is considered a quasi-stationary process over windows (also called frames) of around 25 ms.

The other major sources of speech variability, speaker and channel, are factors which vary on a longer term. Speaker characteristics come from physiological characteristics (vocal-tract length, fundamental frequency) and behavioural characteristics (speech rate, prosody), which can be best described as inhabiting a continuous space. Each speaker is a sample from that continuous multidimensional space. For example, the distribution of formant frequencies is strongly correlated with vocal-tract length on theoretical as well as empirical grounds (Labov, Ash & Boberg, 2006) and shows a strong correlation with body size (Fitch & Giedd, 1999). Channel variability includes microphone characteristics as well as environmental conditions such as background noise level and reverberation, which also clearly vary along continuous scales. Also note that both speaker and channel characteristics can be considered constant or only slowly changing during an utterance or over several seconds. The field of speech recognition boasts many examples of successes based on assuming long-term underlying factors such as utterance-based speaker adaptation (MAP and FMLLR) (Gales, 1998), or i-vector adaptation for DNNs per utterance (Gupta, Kenny, Ouellet & Stafylakis, 2014) or with 3-second windows (Snyder, Garcia-Romero & Povey, 2016a). Thus we will say *utterance-level* or *sequence-level* when speaking of long-term variations, such as over 3 second windows, in contrast to *frame-level* variations occurring in less than 50 ms.

In summary, speech can be described as a sequence of observations, with variability induced by latent variables at the frame and utterance levels. Frame-level variables are short-term, low-dimensional, and have a distribution with multiple modes corresponding to phoneme-like discrete units. Utterance-level continuous variables underlie the long-term variations.

Note that similar assumptions underlie the success of JFA (Kenny *et al.*, 2007a) and i-vector (Dehak *et al.*, 2011) approaches for speaker verification, which are models that assume an underlying Gaussian mixture distribution (a collection of means) at the frame-level, modified for each

utterance by a low-dimensional, linear transformation corresponding to the speaker and channel factors (JFA) or all combined factors of variability (i-vectors).

3.1.2 Short-term variability: discrete prior

As discussed in Section 2.1.4 of Chapter 2, there has been a lot of previous work on the choice of a prior in VAE (VAMP, hyperspherical, flexible, reference prior). These studies have shown that priors other than the simple Gaussian often yield more interpretable representations, especially priors that match the data's underlying structure (Davidson *et al.*, 2018). Also, work on β-VAE also shows that reducing the relative contribution of the simple prior improves disentangling in the representation (Burgess, Higgins, Pal, Matthey, Watters, Desjardins & Lerchner, 2017).

Thus according to our discussion about underlying factors in speech, we should use a multimodal prior for short-term phonetic latent variables, rather than the single Gaussian prior used in conventional VAE. However, a multimodal prior introduces discrete latent variables, which are difficult to integrate in the VAE framework. This is illustrated by the many approaches listed in Section 2.1.5 of Chapter 2: relaxation, Gumbel-Softmax, or adversarial learning.

The problem stems from the non-differentiability of the sampling operation for discrete variables. In the usual stochastic variational formulation, expectations of random variables are computed through Monte Carlo sampling. A significant contribution of the Kingma & Welling (2014) seminal paper was to show how this obstacle can be circumvented for continuous random variables, through the so-called reparameterisation trick. The solution converts the sampling operation, from a parametric distribution, into a transformation of a parameter-free distribution, making it possible to compute gradients for the parameters, and thus to apply stochastic gradient descent (SGD) for learning.

For the discrete case, there is no equivalent of the reparameterisation trick. Thus the proper treatment of discrete variables for VAE is still an open research problem (Pineau & Lelarge, 2018; Rolfe, 2017).

One can sidestep the issue of an explicit discrete variable by using marginalization, a well-known technique used in mixture model estimation. The idea is to compute expectation using a sum over all possible discrete values rather than use Monte Carlo sampling. Such a sum is simple for a discrete variable that indexes mixture components. In Section 3.3, we will show how to derive the ELBO for a VAE with a discrete latent variable, using marginalization.

3.1.3 Long-term variability: sequence modelling

Modelling long-term variations implies that part of the representation changes slowly or stays constant within an utterance, for example. Such a long-term representation is useful to summarize a whole sequence with a single vector, similar to sentence embedding in NLP, or i-vectors in speaker or emotion recognition.

While a continuous prior is a good match for slowly changing continuous underlying factors, the difficulty lies in the pooling operation required to convert frame-level variables into sequence-level ones.

The many approaches that have been proposed to produce a single vector out of a sequence were summarized in Section 2.3. Usually, frame-level vectors are aggregated by averaging, max-pooling, or weighted combinations (attention). Another approach uses sequence-capable neural networks, such as RNN or LSTM, to produce history vectors that embed information about the whole sequence up to the current observation. A third solution is to consider sequence-level and frame-level variables as part of a hierarchical Bayesian model, which provides a principled way to pool sequence statistics.

3.1.4 Summary

In coming-up Section 3.2, we will revisit the conventional VAE model and provide a functional diagram view. We will present the mixture VAE, in Section 3.3 , in which a discrete latent distribution is inferred through conventional marginalization. We will then introduce two novel

approaches, which allow a multimodal prior: one based on expectation maximization (E-M) and one based on divergence sampling, in Section 3.4.1 and Section 3.4.2.

For sequence modelling, we will first extend the proposed multimodal models to a two-level hierarchical Bayesian model, using pooling and tying. We will generalize this approach to arbitrary time scales using filtering layers, and propose self-attention as a mechanism to model sequential dependencies.

The chapter will conclude with a preview of experimental results on simulated vowel signals, to illustrate the potential of the proposed model to capture frame and sequence level factors underlying a complicated, real world-like signal.

3.2 Conventional VAE

A description of the conventional VAE and its objective function was given in Chapter 2. Recall that the best approximate posterior $q_\phi(z|x)$ is found by minimizing its KL divergence with true posterior distribution $p_\theta(z|x)$, which gives rise to the variational objective, or expected lower bound objective (ELBO), written as (Bishop, 2006):

$$\mathcal{L}_{ELBO}(\theta, \phi; x) \;=\; \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \;-\; \mathrm{D}_{KL}[q_\phi(z|x) \parallel p(z)] \tag{3.1}$$

In Equation (3.1), the first term reflects how well the model can reconstruct an observation x given the latent variable z. The second term prevents the approximate posterior from moving arbitrarily away from the latent prior $p(z)$, and thus acts as a regularizer that discourages the trivial solution of memorizing each observation as the latent code.

3.2.1 Functional diagram

Let's revisit the conventional VAE model with a functional diagram rather than a graphical model. Such a diagram is shown in Figure 3.1. It highlights functional dependencies, rather than conditional probabilistic dependencies. Observation x is fed to a NN which outputs parameters of distribution $q_\phi(z|x)$, which is sampled to produce z. This sample is input to another NN which yields parameters of distribution $p_\theta(x|z)$. The KL divergence is computed from the parameters of $q_\phi(z|x)$ and $p(z)$, using a closed-form formula.

In a conventional VAE, the distributions are Gaussian with diagonal variances:

$$\begin{aligned} p_\theta(x|z) &= \mathcal{N}(\mu_\theta(z), \sigma^2_\theta(z)) \\ q_\phi(z|x) &= \mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x)) \end{aligned} \tag{3.2}$$

where μ_θ, σ_θ, μ_ϕ and σ_ϕ are computed by neural networks. Parameter sets ϕ and θ are estimated with SGD, by minimizing the negative of $\mathcal{L}_{ELBO}$ from Equation (3.1).

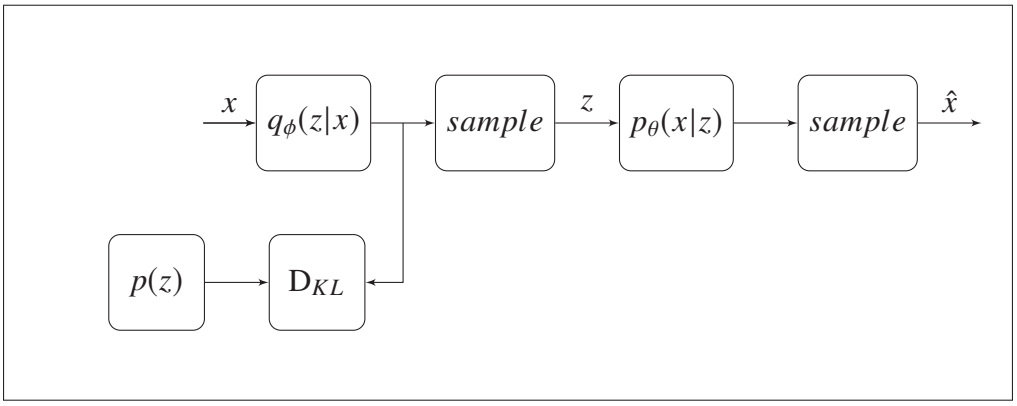


Figure 3.1 VAE functional diagram and closed-form divergence

3.3 Mixture VAE

In this section, we will describe a VAE in which the prior for the continuous z variable is an explicit mixture of Gaussian distributions with a discrete latent variable y. For each observation, y selects one of K mixture components, which then constitutes the prior for the z latent variable.

Several authors have followed this line of inquiry, for example Dilokthanakul *et al.* (2017). However none of these publications provide the full derivation, so here we will make explicit all crucial steps. Extending the VAE to the hierarchical model is helpful to understand it in more detail, and this knowledge will help to follow other derivations in this thesis.

Let's start by defining the generative model, where observation x depends on latent variable z, which itself depends on a mixture with discrete variable y:

$$p_{\beta,\theta}(x, z, y) = p_\theta(x|z)\, p_\beta(z|y)\, p(y) \tag{3.3}$$

The dependencies expressed in Equation (3.3) are illustrated by the graphical model of Figure 3.2, and the generative process can be described in terms of ancestral sampling:

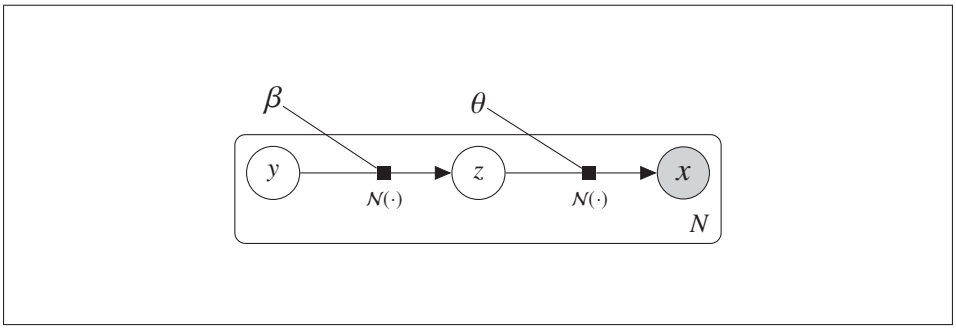


Figure 3.2 Generative model (decoder) for mixture VAE

- For each observation, pick a value for mixture component y among $k \in 1, \ldots, K$ according to a prior categorical distribution Cat$(\cdot)$.
- Given this component, select a mean $\mu_\beta(y = k)$ and a variance $\sigma_\beta(y = k)$ from a collection of components defined by parameters β.
- Sample $z \sim \mathcal{N}(\mu_\beta(y = k), \sigma_\beta(y = k))$.
- Sample $x \sim \mathcal{N}(\mu_\theta(z), \sigma_\theta(z))$. As for the VAE model, the observation x is generated from latent variable z through neural networks with parameters θ.

Following the VAE approach, we define approximate posterior distributions according to Equation (3.4):

$$q_{\phi_z,\phi_y}(z, y|x) = q_{\phi_z}(z|x, y)\, q_{\phi_y}(y|x) \tag{3.4}$$

The resulting graphical model is depicted in Figure 3.3. These dependencies were not chosen arbitrarily. They are constrained by the expression of the ELBO in terms of KL divergence: for every latent variable distribution defined in the generative model, we need a corresponding posterior distribution conditioned on x.

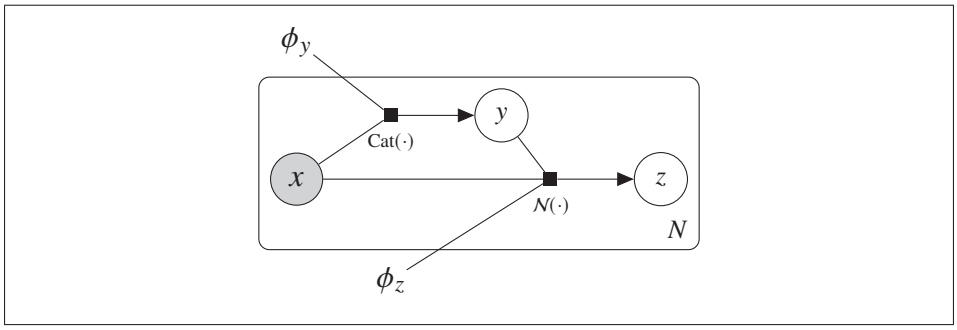


Figure 3.3 Posterior model (encoder) for mixture VAE

3.3.1 Expected lower bound

To make this clear, let's derive the ELBO using the generative and posterior models defined in Equations (3.3) and (3.4). The starting expression is (Dilokthanakul *et al.*, 2017):

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q_{\phi_z,\phi_y}(z,y|x)}[\log p_{\beta,\theta}(x,z,y) - \log q_{\phi_z,\phi_y}(z,y|x)] \tag{3.5}$$

This ELBO for mixture VAE is comparable to the ELBO for the conventional VAE in Equation (2.3). Expanding $p_{\beta,\theta}(x,z,y)$ and $q_{\phi_z,\phi_y}(z,y|x)$ using our generative and posterior models, Equation (3.3) and Equation (3.4), the ELBO can be rewritten:

$$\begin{aligned}\mathcal{L}_{ELBO} = \quad & \mathbb{E}_{q_{\phi_z,\phi_y}(z,y|x)}[\log p_\theta(x|z) + \log p_\beta(z|y) \\ & + \log p(y) - \log q_{\phi_z}(z|x,y) - \log q_{\phi_y}(y|x)]\end{aligned} \tag{3.6}$$

By grouping posteriors and their corresponding priors as in $-\mathrm{D}_{KL}(q \parallel p) = -\mathbb{E}_q(\log p - \log q)$ and applying $\mathbb{E}_{q_{\phi_z,\phi_y}(z,y|x)} = \mathbb{E}_{q_\phi(z)}\mathbb{E}_{q_{\phi_y}(y|x)}$, we obtain:

$$\begin{aligned}\mathcal{L}_{ELBO} = & \qquad\qquad (3.7)\\ & \mathbb{E}_{q_\phi(z)}\mathbb{E}_{q_{\phi_y}(y|x)} \log p_\theta(x|z) \qquad \text{(reconstruction term)} \\ & \quad - \mathbb{E}_{q_{\phi_y}(y|x)}\mathrm{D}_{KL}[q_{\phi_z}(z|x,y) \parallel p_\beta(z|y)] \qquad (z\text{-prior}) \\ & \quad - \mathbb{E}_{q_\phi(z)}\mathrm{D}_{KL}[q_{\phi_y}(y|x) \parallel p(y)] \qquad (y\text{-prior})\end{aligned}$$

The right-hand side contains three terms: a reconstruction term, a z-prior term and a y-prior term.

The y-prior term does not depend on $q_\phi(z)$ so it reduces to:

$$\mathbb{E}_{q_\phi(z)}\mathrm{D}_{KL}[q_{\phi_y}(y|x) \parallel p(y)] = \mathrm{D}_{KL}[q_{\phi_y}(y|x) \parallel p(y)] \tag{3.8}$$

The z-prior term is an expectation over the discrete distribution $q_{\phi_y}(y|x)$, which can be computed by marginalization, i.e., summing over all possible values of the discrete variable y:

$$\mathbb{E}_{q_{\phi_y}(y|x)}\mathrm{D}_{KL}[q_{\phi_z}(z|x,y) \parallel p_\beta(z|y)] = \sum_{k=1}^{K} q_{\phi_y}(y=k|x)\mathrm{D}_{KL}[q_{\phi_z}(z|x,y=k) \parallel p_\beta(z|y=k)] \tag{3.9}$$

Similarly, the reconstruction term is an expectation over the discrete variable and can be computed by summing. Inside the sum, the expectation of $p_\theta(x|z)$ is approximated by Monte Carlo sampling of z, as done in a conventional VAE:

$$\begin{aligned}\mathbb{E}_{q_{\phi_y}(y|x)}\mathbb{E}_{q_\phi(z)} \log p_\theta(x|z) &= \sum_{k=1}^{K} q_{\phi_y}(y=k|x)\, \mathbb{E}_{q_{\phi_z}(z|x,y=k)} \log p_\theta(x|z) \\ &\approx \sum_{k=1}^{K} q_{\phi_y}(y=k|x)\, \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(x \mid z_{(l,k)})\end{aligned} \tag{3.10}$$

with L samples $z_{(l,k)} \sim q_{\phi_z}(z|x,y=k)$.

Our final expression for the mixture VAE ELBO is thus:

$$\mathcal{L}_{ELBO} = -\mathrm{D}_{KL}[q_{\phi_y}(y|x) \parallel p(y)] + \sum_{k=1}^{K} q_{\phi_y}(y=k|x)\big[\mathcal{L}_{ELBO}(y=k)\big] \tag{3.11}$$

where

$$\mathcal{L}_{ELBO}(y=k) = \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x \mid z_{(l,k)}) - \mathrm{D}_{KL}[q_{\phi_z}(z|x, y=k) \parallel p_\beta(z|y=k)] \tag{3.12}$$

Note that all KL divergences are now expressed in terms of the parametric distributions $q_{\phi_y}(y|x)$, $p(y)$, $q_{\phi_z}(z|x, y)$ and $p_\beta(z|y)$ so they can be efficiently computed in closed form, i.e., directly from distribution parameters rather than from sampled values.

However, when we did preliminary experiments on pinwheel and MNIST datasets, we found this model was difficult to train, had trouble identifying modes, easily dropped components, etc. Similar problems are observed in general when estimating mixture models (Barber, 2016).

3.4 Multimodal VAE

In this section, we introduce two novel approaches which allow a multimodal prior to be estimated, one based on expectation maximization (E-M) and one based on divergence sampling.

Recall that in a conventional VAE, the KL term is computed from parameters of $q_\phi(z|x)$ and $p(z)$, generated by neural networks, as was illustrated in the functional diagram of Figure 3.1. In a conventional VAE, both $q_\phi(z|x)$ and $p(z)$ are assumed to be Gaussian distributions, and while $q_\phi(z|x)$ changes with each observation, the prior $p(z)$ is fixed.

To attain our objective of a multimodal prior distribution for z, instead of positing a mixture model with a discrete latent variable, as was done in the previous section, could we just represent $q_\phi(z|x)$ and $p(z)$ directly as multimodal distributions? For example, since a Gaussian mixture model is fully described by parameters $\Pi_k,\ \mu_k,\ \sigma_k$ for k in $[0, \ldots, K]$, we could have a NN produce these parameters for $q_\phi(z|x)$ and $p(z)$ and estimate these parameters with SGD?

Although there is no explicit discrete variable in this approach, there are two major problems. First, KL divergence cannot be computed in closed form between Gaussian mixtures, although approximations exist (Durrieu, Thiran & Kelly, 2012; Hershey & Olsen, 2007). More importantly,

there is a sampling step that produces z, and as sampling from a mixture is not differentiable, it prevents the gradient from being propagated back to update the ϕ parameters defining $q_\phi(z|x)$.

In the following sections, we introduce novel solutions to these two problems, which allow a multimodal prior in VAE without an explicit discrete variable and without recourse to marginalization. The solutions are based on two insights.

3.4.1 E-M reestimation

The first insight is that, even though $q_\phi(z|x)$ is a Gaussian, it is conditioned on x. When aggregated over all data, the aggregated posterior $\hat{q}(z)$ is:

$$\hat{q}(z) = \mathbb{E}_{p_{data}(x)}[q_\phi(z|x)] \tag{3.13}$$

where $p_{data}(x)$ denotes the empirical distribution. A different Gaussian is sampled for each observation; thus, in effect, the aggregated posterior $\hat{q}(z)$ is a mixture (Lopez, Regier, Jordan & Yosef, 2018).

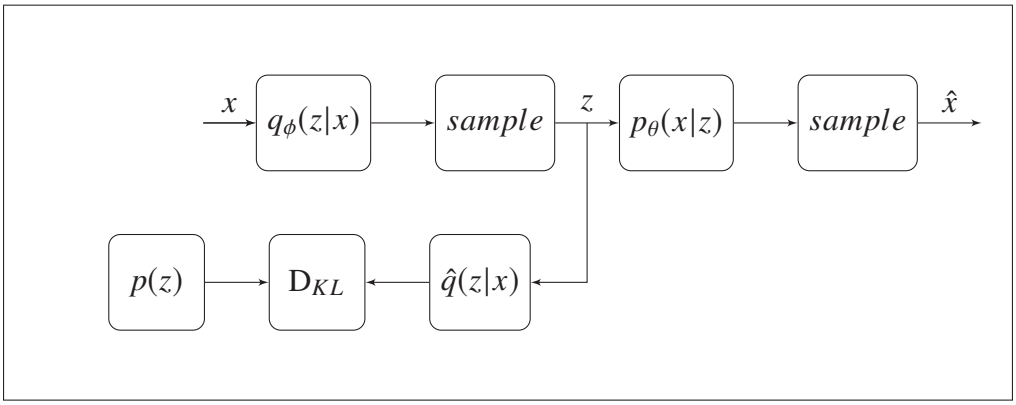


Figure 3.4 VAE with E-M reestimation

Figure 3.4 shows how this insight can be used to impose a multimodal prior. We keep $q_\phi(z|x)$ as a Gaussian (per observation), which allows us to compute the gradient when sampling z to reconstruct x. We set up the prior $p(z)$ as a mixture of Gaussians with fixed parameters.

We approximate the posterior distribution $q_\phi(z|x)$ over all the data with a Gaussian mixture, which we estimate as a parametric distribution $\hat{q}(z|x)$ using the sampled z values. More precisely, the mixture components and weights of $\hat{q}(z|x)$ are reestimated using one cycle of the E-M algorithm (Bishop, 2006), where component responsibilities are computed using current means and variances, then means and variances are updated using component responsibilities[1]. $\hat{q}(z|x)$ is initialized with $p(z)$ so that the components of $\hat{q}(z|x)$ and $p(z)$ are matched. Then the KL divergence is a sum of individual divergences between matched components, and the individual divergences can be computed in closed form from the distribution parameters, as illustrated by the two inputs of the D_{KL} box in Figure 3.4. We derive the expression for matched KL divergence in the following subsection.

3.4.1.1 Matched KL divergence

The matched component condition allows us to compute a KL divergence between two Gaussian mixtures, which otherwise is not possible in general.

Consider two Gaussian mixtures p and q, where individual components are single Gaussians $\mathcal{N}_q(\cdot; i)$ and $\mathcal{N}_p(\cdot; j)$:

$$q = \sum_{i=1}^{K} \pi_q(i)\mathcal{N}_q(\cdot; i) \tag{3.14}$$

$$p = \sum_{j=1}^{K} \pi_p(j)\mathcal{N}_p(\cdot; j) \tag{3.15}$$

Their KL divergence is:

$$\begin{aligned} D_{KL}[q \parallel p] &= \mathbb{E}_q[log(q) - log(p)] \\ &= \mathbb{E}_q[\log \sum_i \pi_q(i)\mathcal{N}_q(\cdot; i) - \log \sum_j \pi_p(j)\mathcal{N}_p(\cdot; j)] \end{aligned} \tag{3.16}$$

[1] In practice, for each minibatch, before computing the KL divergence, $\hat{q}(z|x)$ is reestimated with the z values sampled for the minibatch.

According to the generative process for mixture q, each observation is generated from a component i selected from prior distributions $i \sim \pi_q(i)$ and similarly, $j \sim \pi_p(j)$ for mixture p. Crucially, we further assume that i and j are not selected independently for each mixture, but that the observation is generated from the same index $i = j = k$ in both mixtures q and p. Ignoring all components except the matching ones, the sum over components is replaced by a single value for $i = j = k$:

$$\begin{aligned}
\mathrm{D}_{KL}[q \parallel p] &= \mathbb{E}_q[\log \sum_{i=k} \pi_q(i)\mathcal{N}_q(\cdot; i) - \log \sum_{j=k} \pi_p(j)\mathcal{N}_p(\cdot; j)] \\
&= \mathbb{E}_q[\mathbb{E}_k[\log \pi_q(k)\mathcal{N}_q(\cdot; k) - \log \pi_p(k)\mathcal{N}_p(\cdot; k)]] \qquad \text{(using } k \sim \pi_q\text{)} \\
&= \mathbb{E}_q[\sum_k \pi_q(k)[\log \mathcal{N}_q(\cdot; k) - \log \mathcal{N}_p(\cdot; k) + \log \pi_q(k) - \log \pi_p(k)]] \\
&= \sum_k \pi_q(k)\mathbb{E}_q[\log \mathcal{N}_q(\cdot; k) - \log \mathcal{N}_p(\cdot; k) + \log \pi_q(k)] + \log \pi_q(k) - \log \pi_p(k) \\
&= \sum_k \pi_q(k)[\mathrm{D}_{KL}(\mathcal{N}_q(\cdot; k) \parallel \mathcal{N}_p(\cdot; k)) + \log \pi_q(k) - \log \pi_p(k)] \qquad (3.17)
\end{aligned}$$

This final expression for matched component KL Equation (using $k \sim \pi_q$) is thus the expectation over k of KL divergences of individual mixture components. Applying this to our reestimation model in Figure 3.4, we obtain:

$$\mathrm{D}_{KL}[\hat{q}(z|x) \parallel p(z)] = \sum_{k=1}^{K} \pi_{\hat{q}}(k)[\mathrm{D}_{KL}(\mathcal{N}_{\hat{q}}(k) \parallel \mathcal{N}_p(k)) + \log \pi_{\hat{q}}(k) - \log \pi_p(k)] \qquad (3.18)$$

where k is the index of a mixture component and $\pi_{\hat{q}}(k)$ and $\pi_p(k)$ are the component weights for $\hat{q}(z|x)$ and $p(z)$ respectively, and $\mathcal{N}_{\hat{q}}(k)$ and $\mathcal{N}_p(k)$ are K individual Gaussian distributions. This expression is the same as the K-L approximation proposed in (Goldberger, Gordon & Greenspan, 2003) for Gaussian mixtures with matching components for the particular case where the matching function is an identity mapping.

3.4.2 Divergence sampling

The second insight uses sampling to estimate a KL divergence between two arbitrary distributions. More precisely, the KL divergence between $q_\phi(z|x)$ and $p(z)$ being an expectation (by definition), it can be estimated by sampling L values of z (Hammersley & Handscomb, 1964):

$$\begin{aligned} \mathrm{D}_{KL}[q_\phi(z|x_n) \parallel p(z)] &= \mathbb{E}_{z\sim q_\phi(z|x_n)}[\log q_\phi(z|x_n) - \log p(z)] \\ &\approx \frac{1}{L}\sum_{l=1}^{L}[\log q_\phi(z_l|x_n) - \log p(z_l)] \end{aligned} \tag{3.19}$$

with L samples $z_l \sim q_\phi(z|x_n)$ (we use all observations in a minibatch, i.e., L equal to the minibatch size). Note that here, we average values of KL divergence between two parametric distributions, and each distribution is evaluated at sample points defined by z_l.

This expression is valid for any arbitrary $q_\phi(z|x_n)$ and $p(z)$ (given some weak constraints about the support of each). If prior $p(z)$ is a Gaussian mixture, minimizing Equation (3.19) forces the *aggregated* posterior to match a Gaussian mixture.

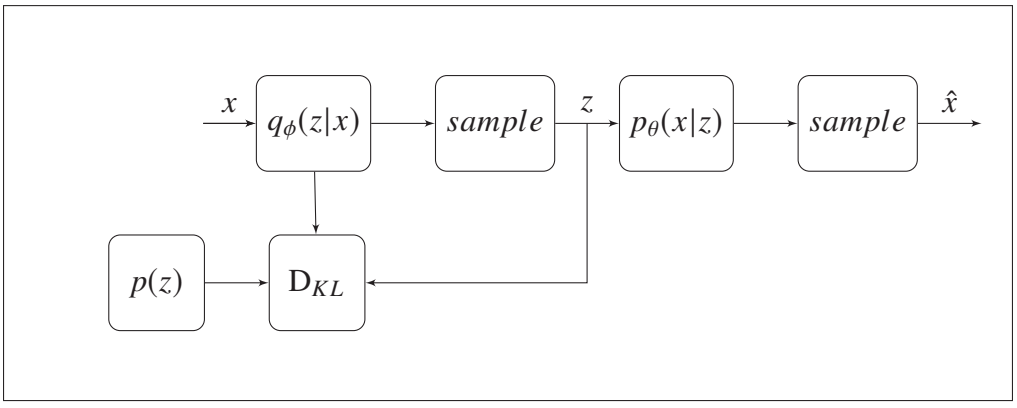


Figure 3.5 VAE with sampling divergence

The implementation follows the functional diagram of Figure 3.5. At the end of each minibatch, z values are sampled from $q_\phi(z|x)$ and fed into Equation (3.19) together with the parameters of $q_\phi(z|x)$ and $p(z)$. Note the three inputs to the D_{KL} box.

This approach can be related to Maximum Mean Discrepancy VAE (MMD-VAE), as proposed in (Zhao, Song & Ermon, 2017). MMD-VAE replaces the KL divergence in the conventional ELBO Equation (2.4) by a distance measure that compares a batch of samples z from encoded x observations, with a batch of samples from $p(z) = \mathcal{N}(0, 1)$:

$$\mathcal{L}_{\text{MMD-VAE}} = \text{MMD}(q_\phi(z)\|p(z)) + \mathbb{E}_{p_{data}(x)}\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \tag{3.20}$$

In contrast, here, we use a KL divergence, but estimate it using Equation (3.19):

$$\mathcal{L}_{\text{ELBO}_{data}} = \mathbb{E}_{p_{data}(x)}[-\text{KL}(q_\phi(z|x)\|p(z))] + \mathbb{E}_{p_{data}(x)}\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \tag{3.21}$$

The crucial difference here is that our KL estimate depends on two parametric distributions and should be less noisy than MMD, which relies on pairs of sampled values. To our knowledge, MMD-VAE is used only with simple Gaussian priors, and has never been applied to allow multimodal priors.

3.4.3 Summary for multimodal VAE

Three possible approaches were proposed: one with explicit discrete variables using marginalization, and two with implicit multimodal posteriors. Marginalization has been shown in the past to be difficult in the case of mixtures (Maaløe *et al.*, 2017a). The implicit posterior E-M approach might suffer from similar problems, since the E-M cycle is based on a similar marginalization of an implicit discrete indicator variable. The divergence sampling is elegant and less computationally expensive, but its variance as an estimator is not known.

3.5 Sequence and frame levels

Now we turn to the problem of distinguishing short-term and long-term underlying factors.

We split the latent variable in the VAE into a frame-level component, which changes for every observation, and a sequence-level component, which stays fixed for the duration of a sequence, to model longer term variations.

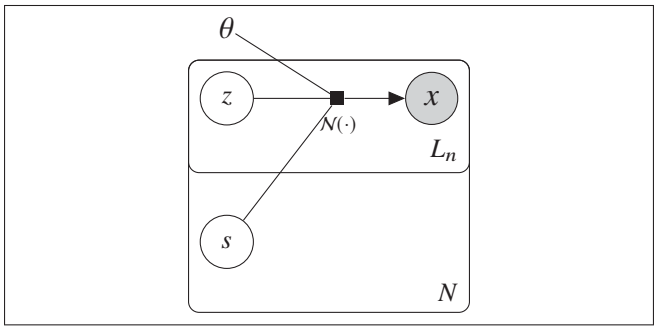


Figure 3.6 Generative model (decoder) for N sequences of length L_n. The frame-level variable is z, and the sequence-level variable is s

Given N sequences, $n = 1, \ldots, N$, each of length L_n with observations x_l, $l = 1, \ldots, L_n$, we can set up a two-level generative process as illustrated in the graphical model of Figure 3.6. The sequence-level variable s has a different value for each of the N sequences, while frame-level z changes with each of the L_n observations in the sequence.

In this graphical model, x depends on both z and s, which are independent, and we can write:

$$p_\theta(x, z, s) = p_\theta(x|z, s)p(z)p(s) \tag{3.22}$$

The corresponding posterior model is shown in Figure 3.7, and the posterior distribution is:

$$q_{\phi_z,\phi_s}(z, s|x) = q_{\phi_z}(z|x)q_{\phi_s}(s|x) \tag{3.23}$$

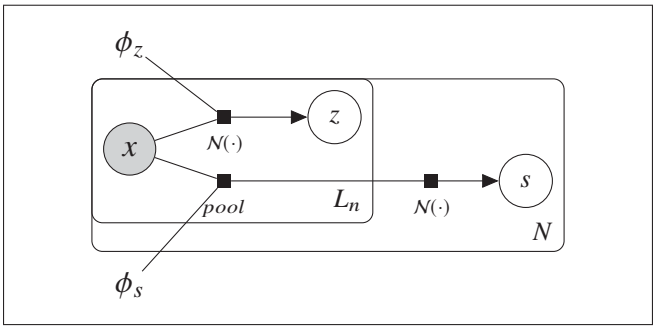


Figure 3.7 Posterior model (encoder) with N sequences of length L_n

Note the *pool* operation used on observations $x_1, \ldots, x_{L_n}$ to generate a single distribution for the sequence. This will be explained in Section 3.5.1.

Given these generative and posterior models, we can derive the ELBO in the same way as for to the mixture VAE Equation (3.5):

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q_{\phi_z,\phi_s}(z,s|x)}[\log p_\theta(x, z, s) - \log q_{\phi_z,\phi_s}(z, s|x)] \tag{3.24}$$

Following a derivation similar to the one in Section 3.3, we arrive at the following expression by grouping posteriors and their corresponding priors as in $-\mathrm{D}_{KL}(q \parallel p) = -\mathbb{E}_q(\log p - \log q)$ and applying $\mathbb{E}_{q_{\phi_z,\phi_s}(z,s|x)} = \mathbb{E}_{q_{\phi_z}(z|x)}\mathbb{E}_{q_{\phi_s}(s|x)}$ to obtain:

$$\begin{aligned} \mathcal{L}_{ELBO} = & \qquad (3.25) \\ & \mathbb{E}_{q_{\phi_z}(z|x)}\mathbb{E}_{q_{\phi_s}(s|x)} \log p_\theta(x|z, s) \qquad \text{(reconstruction term)} \\ & - \mathrm{D}_{KL}[q_{\phi_z}(z|x) \parallel p(z)] \qquad (z\text{-prior}) \\ & - \mathrm{D}_{KL}[q_{\phi_s}(s|x) \parallel p(s)] \qquad (s\text{-prior}) \end{aligned}$$

3.5.1 Tying and pooling

Note that in Figure 3.6, a sequence-level variable s (in the N plate) is combined with a frame-level variable z (in the L_n plate). Similarly, Figure 3.7, x in plate L_n enters a functional in plate N. How should we handle these transitions between levels?

To answer, we have to reintroduce observation and sequence indices on the observations and variables. Given N sequences of length $L_n,\ n = 1, \ldots, N$, the ELBO over the whole dataset X is:

$$\mathcal{L}_{ELBO}(\theta, \phi; X) = -\frac{1}{N} \sum_n \frac{1}{L_n} \sum_{l=1}^{L_n} \mathrm{D}_{KL}[q_{\phi_{z_{n,l}}}(z_{n,l}|x_{n,l}) \parallel p(z)] \tag{3.26}$$

$$+ \frac{1}{N} \sum_n \frac{1}{L_n} \sum_{l=1}^{L_n} \left(\log p_\theta(x_{n,l}|z_{n,l}, s_n) - \mathrm{D}_{KL}[q_{\phi_{s_n}}(s_n|x_{n,l}) \parallel p(s)] \right) \tag{3.27}$$

Now we have variables $z_{n,l}$ and $x_{n,l}$ indexed by both sequence number n and position l in the sequence, while s_n is only indexed by sequence number n.

The reconstruction cost appears as the leftmost term in Equation (3.27) and corresponds to the decoder of Figure 3.6. $x_{n,l}$ is conditioned on a single s_n for $l = 1, \ldots, L_n$, so this distribution is straightforwardly represented as neural network by *tying* its input s_n across L_n frames in a sequence:

$$s_{n,l} = s_n \quad \text{for } l = 1, \ldots, L_n \tag{3.28}$$

In other words, the L_n values of $s_{n,l}$ are *tied* to the single value s_n and the sums run over l and n. This provides a natural way to account for sequence-level contributions to the ELBO at the frame-level, with the proper scaling.

To handle the rightmost term in Equation (3.27) (encoder of Figure 3.7), we must condition a single sequence-level s_n on L_n frame-level variables $x_{n,l}$. This is problematic, as it requires for $q_{\phi_{s_n}}(s_n|x_{n,l})$ a neural network with an input that varies in length from sequence to sequence.

A frequently used approach to solve this problem is *pooling* by averaging, in which the different $x_{n,l}$ are approximated by a single variable which is their average over the sequence n:

$$x_n = \frac{1}{L_n} \sum_{l=1}^{L_n} x_{n,l} \tag{3.29}$$

This approach was used successfully in speaker recognition by Cai, Chen & Li (2018); Snyder *et al.* (2016b); Villalba *et al.* (2017).

3.5.2 Filtering

Together, the tying Equation (3.28) and pooling Equation (3.29) suggest a more general approach. Let us approximate s_n using a moving average over W frames:

$$s_n \approx \tilde{s}_{n,l} = \frac{1}{W} \sum_{w=-\frac{1}{2}W}^{w=+\frac{1}{2}W} s_{n,l+w}, \tag{3.30}$$

where the additional subscript l indicates a separate variable for each frame. We can use a frame-based $s_{n,l}$ variable everywhere in the model and replace s_n by its averaged version $\tilde{s}_{n,l}$ when computing the loss in Equation (3.27). We obtain a completely frame-based loss expression, where all variables are indexed by n, l. As we just did for $s_{n,l}$ and Equation (3.27), we can also replace $z_{n,l}$ by its averaged version $\tilde{z}_{n,l}$ in loss Equation (3.26).

Using averaged variables, the ELBO simplifies to:

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q_{\phi_z}(z|x)} \mathbb{E}_{q_{\phi_s}(s|x)} \log p_\theta(x|\tilde{z}, \tilde{s}) - \mathrm{D}_{KL}[q_{\phi_z}(\tilde{z}|x) \parallel p(z)] - \mathrm{D}_{KL}[q_{\phi_s}(\tilde{s}|x) \parallel p(s)] \tag{3.31}$$

where indices were dropped without loss of generality. With averaged variables, the ELBO includes a soft pooling/tying parametrized by the averaging filters' window lengths.

With Equation (3.31), we can use a different averaging window length for $\tilde{z}$ and $\tilde{s}$. For example, a short window captures short-term information in z and a long window captures long-term information in s. We are no longer limited to either frame-level or sequence-level variables, as we can choose a filter of any length from single frame to length of a whole sequence.

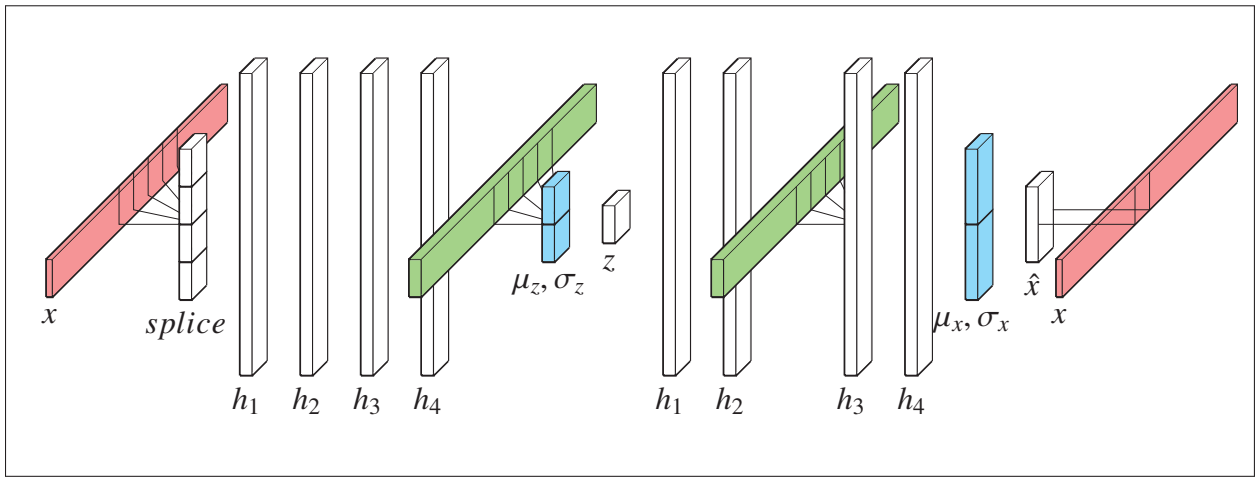


Figure 3.8 Proposed VAE model architecture with filtering layers.
Only latent variable z is illustrated for simplicity

In a neural network, a moving average filter such as Equation (3.30) is naturally implemented as an averaging layer. This is illustrated in Figure 3.8, which depicts how the frame-based ELBO of Equation (3.31) can be implemented with frame-based neural network layers, i.e., where each frame of input produces one frame of output.

In the figure, a frame is represented by one vertical column, with its height proportional to its dimensionality. Frames of the input sequence x appear in red on the left. A splicing layer combines several input frames into a single vector for the current frame. This is followed by hidden layers h_1, h_2, h_3, h_4 and the output layer of the encoder (in blue), which produces the mean and variance of z. Sampling produces z, which is input to the decoder consisting of hidden layers h_1, h_2, h_3, h_4 and an output layer, in blue, for the mean and variance of the reconstructed signal. Finally, sampling reconstructs one frame of output $\hat{x}$, that becomes part of the whole output sequence, in red.

The averaging layers are shown in green. They combine several adjacent frames with the current frame into a single vector for the current frame. Even though their original purpose is to provide a convenient way of computing the ELBO for latent variables with different time scales, averaging filters seen as simple layers can easily be moved to different locations along the depth of the network. Figure 3.8 illustrates two possible locations: one right after the last hidden layer of the encoder, and the other between two hidden layers of the decoder.

Filtering in the decoder can be motivated by asking how the model could generate smooth transitions in the observed signal. Even though the latent variable is transformed by a stack of non-linear layers, without a filter, it would have to exhibit smooth transitions itself. A low-pass filter inserted after the latent variable should allow it to have sharp transitions and still generate a smooth observed output. We can make here an analogy with the speech production model, where discrete underlying phoneme commands are executed by slow articulators, followed by the nonlinear vocal-tract to sound function. Smooth articulatory transitions can be converted to sharp acoustic changes at critical points (Stevens & Keyser, 2010).

Filtering in the encoder can be motivated by the need for a larger context than the current frame. A filter provides indirect access to the information left and right of the current frame, depending on its length, and is especially important for a sequence-level variable that seeks to capture information that depends on the whole sequence.

Since there are good motivations both for decoder and encoder locations, the question needs be explored experimentally.

The averaging filter was designed to induce, in the latent variable, a bias towards a particular time scale. It is tempting to replace the simple averaging filter by a more general filter, for example one with learnable weights, such as a one-dimensional convolution layer. However, relaxing the constraints put on the latent variable also removes the bias. Indeed, preliminary experiments showed that when a 1-D convolution is used in place of an averaging layer, it greatly improves reconstruction quality, but degrades the usefulness of the latent variable as a representation for a downstream phoneme classification task.

3.5.3 Attention

The model proposed so far makes use of time scale to bias latent variables towards useful representations. However the generative model assumes independence between latent variables (as in Figure 3.6). So it lacks a mechanism to learn long-term dependencies across a sequence of short-term variables, for example to learn probable phoneme sequences. Indeed, the most successful "unsupervised" model for learning phonemes (Chen, Tsai, Liu, Lee & Lee, 2019; Yeh, Chen, Yu & Yu, 2018) uses a phoneme language model as a weak supervision[2].

Approaches that have been used to model sequences include recurrent neural networks (RNN), in particular bidirectional long short-term memory (BLSTM) models, which have been used in the FHVAE model (Hsu *et al.*, 2017b). However, they are slow to train, and have a too limited effective memory (Al-Rfou *et al.*, 2019; Sperber *et al.*, 2018).

Recently, self-attention has been proposed as a different mechanism that can take into account large contexts and produce state-of-the-art results, but is much faster to train than recurrent networks (Chowdhury *et al.*, 2018; Sperber *et al.*, 2018). In the seminal "Transformer" paper by Vaswani *et al.* (2017), self-attention was used as a supervised learning method, given pairs of sentences in source and target language. Training is done by predicting future target words given past source words, with an encoder/decoder tandem. Training can also be framed as an unsupervised task, if input and output sentences are the same, for learning a language model, for example (Al-Rfou *et al.*, 2019).

Attention was historically introduced for supervised learning in text processing, and has been since applied to speech recognition (Chorowski *et al.*, 2015; Povey *et al.*, 2018; Sperber *et al.*, 2018) and speaker recognition (Chowdhury *et al.*, 2018; Povey *et al.*, 2018; Wang *et al.*, 2019; Zeinali *et al.*, 2019; Zhu *et al.*, 2018), but always in supervised settings. Here, we present the first use of self-attention for unsupervised learning in speech.

---

[2] The language model has to be trained on text phoneme sequences, but as these texts need not be related to the audio, the authors still consider their model as unsupervised.

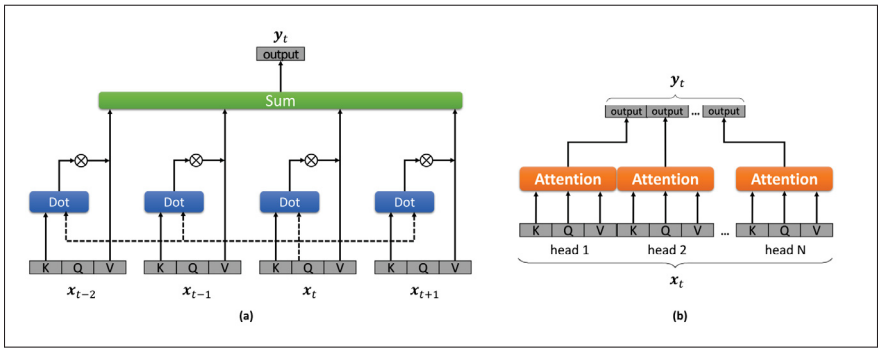


Figure 3.9 (a) Single-head attention block, for frame t.
(b) Multi-head attention using single-head attention blocks. K, Q, and V mean key, query, value, respectively
Taken from Povey *et al.* (2018, p. 5876)

Figure 3.9 (a) illustrates part of a single-head attention block. In our model, the full input sequence $t = 0, \cdots, T$ is supplied to the attention block. The key, query and value (K, Q, V) are learnable, linear projections of the input frame x. At time t, Q_t is computed from x_t, and its dot product with keys K_t for $t = 0, \cdots, T$ produces a set of weights $\alpha(t)$ (which are normalized to sum to one). The output y_t is formed by the weighted sum $\sum_{t=0}^{T} \alpha(t) V_t$. This configuration is called self-attention, since at each time, different parts of the input sequence can be emphasized to produce the current output, based on the current frame of the input sequence itself.

This kind of self-attention can be motivated again by considering the speech production viewpoint. Anticipatory coarticulation, for example, can happen when some articulator is not involved in the current phoneme and is free to anticipate its movement in preparation for a following phoneme which may be quite far in the future (Browman & Goldstein, 1992). For example, lips can anticipate rounding of a round vowel /u/ several phonemes in the future when producing an /s/, provided they are not involved until /u/ is reached, and this will strongly affect the /s/ sound produced. In this case, the identities of both the current phoneme and a phoneme far in the future are needed to correctly reconstruct the signal. Self-attention, when applied to the latent variable intended to capture phoneme identity, has access to the current and all future phoneme identities (as well as all past ones) in the sequence. In principle, a query derived from

/s/ could match a future key derived from /u/ to return a value that would add "roundness" in the concatenated output.

Multi-head attention is illustrated in Figure 3.9 (b) and simply consists of multiple single-head attention blocks operating in parallel. The multiple outputs are concatenated, and projected to a smaller dimension (this projection is not illustrated in the figure). Each head has the ability to capture a different relation between different parts of the input sequence.

A multi-head attention layer can be inserted in the model stack of network layers just as a filtering layer, and in fact would look just like the averaging layer in Figure 3.8, except that it has access to the whole utterance. However, since its role is to capture long-term dependencies between successive latent variables, we apply it to the encoder output i.e., μ_z and σ_z form the input of the attention layer.

3.6 Multiple filtered latent variables (MFL) VAE

To summarize the preceding sections, the ELBO for multiple latent with V variables can be generalized from Equation (3.31):

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q_{\phi_z},\phi_s} \log p_\theta(x|\tilde{z}_1, \cdots, \tilde{z}_V) - \sum_{v=1}^{V} \beta_v \, \mathrm{D}_{KL}[q_{\phi_v}(\tilde{z}_v|x) \parallel p_v(z)], \tag{3.32}$$

where variables $\tilde{z}_1, \cdots, \tilde{z}_v$ are filtered versions of the latent variables.

We call this model the MFL VAE. Note that for more generality, we introduce a β_v weighting term in the sum of KL divergences, as used with success to yield better disentangling in the β-VAE model (Burgess *et al.*, 2017; Higgins *et al.*, 2017).

The corresponding functional diagram appears in Figure 3.10, where priors and D_{KL} divergences have been omitted for simplicity. Each latent variable is sampled from its posterior distribution $q_{\phi_v}(z_v|x)$ and concatenated with the other latent variables before being fed to the decoder that estimates $p_\theta(x|z_1, \cdots, z_V)$.

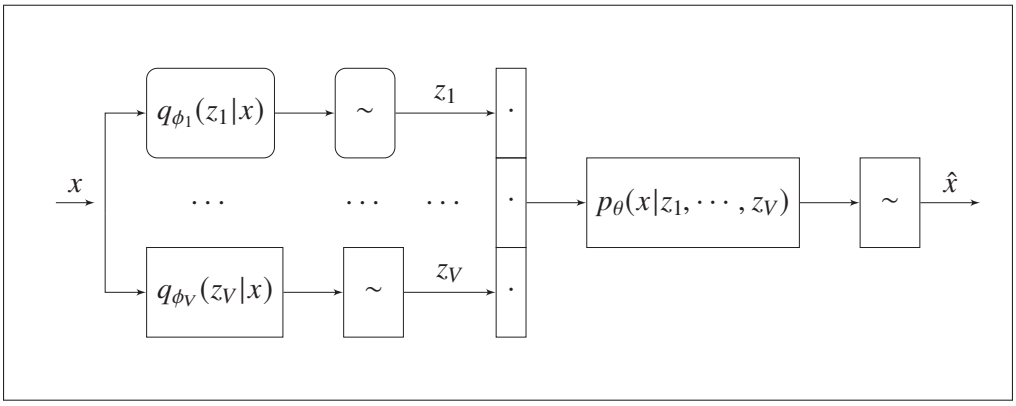


Figure 3.10 Multiple latent variables model. Filtering layers are not illustrated

Each latent variable z_v has its own pair of posterior and prior $p_v(z)$ distributions, so we can use a mix of discrete and continuous priors (using the adequate form of D_{KL} for each prior).

If we use only one latent variable ($V = 1$), a single Gaussian prior, and $\beta_1 = 1.0$, we recover the conventional VAE model of Kingma & Welling (2014). Then, relaxing β_1 reproduces the β-VAE model of Higgins *et al.* (2017). Using a mixture of Gaussians as a prior implements the multimodal VAE of Section 3.4. When $V = 2$, we can use a short and a long filter to implement the two-level, filtered model of Section 3.5.2.

3.7 Preliminary experiments

When proposing a new model, it is crucial to be able to verify the correctness of its implementation, and its ability to recover latent variables from the observed data.

Using real speech does not allow that, as it introduces an additional unknown, which is the match between assumptions and actual speech. With real speech, it is not possible to distinguish a model failing because of a wrong implementation or because of wrong modelling assumptions.

Before going into other modelling propositions, we need to be able to test modelling assumptions in a more controlled way than with real datasets. For example, while MNIST observations correspond to discrete underlying units (digits), continuous factors of variations such as slant

and boldness are not explicitly given and in fact, are often just visual judgments superimposed after the fact on classification results. Furthermore, the notions of sequence and observation levels are not represented in these databases.

A theoretical framework proposed by Eastwood & Williams (2018) quantitatively measures how well the representation learned by a model is disentangled and interpretable. To apply it, we need synthetic data for which the ground-truth latent structure is known. At the same time, the synthetic data must be realistic enough so that capturing underlying factors is as difficult as for actual data.

We generate random synthetic vowel data using a simplified speech production model[3]. We specify each of 5 vowels by three mean canonical formant frequencies. Speakers are represented as a single, continuous variable that determines vocal-tract length and fundamental frequency. We then generate sequences of signal frames by randomly sampling a vowel and a speaker, modify formants according to vocal-tract length, convert them to spectral envelope, convolve it with a source excitation dependent on fundamental frequency, and convert the source-filter output to frames of log Mel filterbank features. Thus each generated frame is a highly non-linear transformation of latent random vowel and speaker variables. For vowels, the latent variable resides at 5 discrete locations in 3-dimensional space, and for speakers, along a 1-dimensional continuum.

Figure 3.11 shows how well our model retrieves latent variables from the log Mel filterbank features of a development set, after training without any supervision. The top row is the model input. On the top left, Mel filterbank features are displayed as a spectrogram. 25 individual sequences of 4 frames per vowel are shown, forming 100 adjacent frames for each vowel. High variability within a vowel and overlaps in frequencies across different vowels are observed. The top middle and right plots display samples from the prior distributions of the two latent variables in the model, a short-term Gaussian mixture[4] and a long-term single Gaussian, respectively.

[3] Full details will be presented in Section 4.4

[4] With 7 means to show that the model has some freedom in the number of modes it finds.

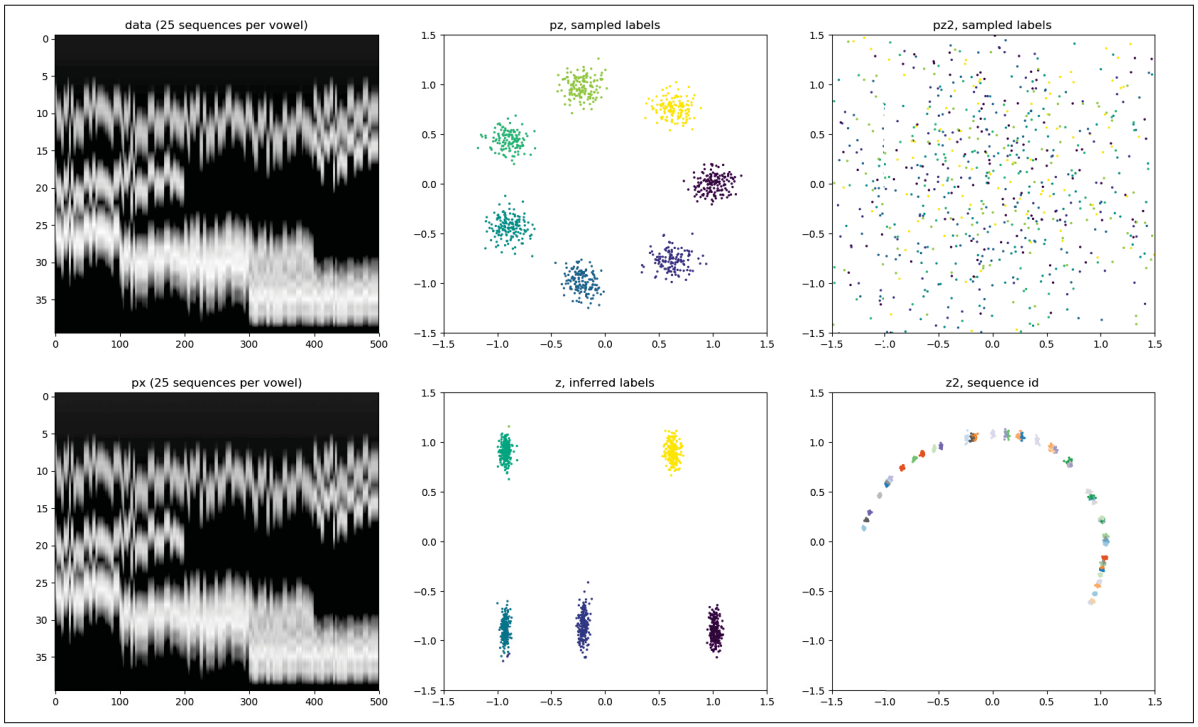


Figure 3.11 Modelling synthetic vowels with frame and sequence latent variables. Top row, from left to right: original filterbank features, frame-level prior, sequence-level prior. Bottom row: reconstructed signal, samples of frame-level variable, samples from sequence-level variable

The bottom row is the model output on the development set. On the left is the reconstructed signal sampled from the decoder output. The middle plot shows samples from the posterior distribution obtained for the first latent variable, coloured by vowel identity: the samples form well-defined clusters that match vowels. The right plot shows samples from the posterior distribution of the second latent variable, coloured by speaker identity. Samples form small clusters with a single speaker colour; clusters are arrayed along a well defined 1-dimensional manifold in the two-dimensional space.

So our model does indeed appear to correctly recover the latent variables used to generate the data. Full experimental results in Chapter 4 will show that the proposed time scale filtering is essential for capturing separately frame-level and sequence-level variables such as the vowel and speaker identity.

3.8 Summary

We relied on past empirical observations of speech properties to formulate variational autoencoding models with the specific objective of unsupervised representation learning in speech. We discussed the need to bias models towards meaningful representations based on time scale and type of prior for the latent variables. We proposed several extensions to current VAE models and a unified multiple filtered variables VAE model that encompasses all our propositions.

In closing this chapter, we gave a preview of preliminary results with synthetic vowels, generated according to the source-filter speech production model, that confirm our model's capacity to infer latent factors used to generate the data. From now on, for conciseness, we will refer to our proposed VAE model with multiple, filtered latent variables as the MFL VAE.

CHAPTER 4

EXPERIMENTAL RESULTS

The objective of our experiments is to verify that the proposed MFL VAE model yields representations that capture meaningful and disentangled underlying factors, as measured by the accuracy of a downstream unsupervised classification task. More specifically, for each representation that the model generates, we use unsupervised K-means clustering, and measure classification accuracy of this clustering against the target unit for the representation, for example phoneme accuracy, speaker accuracy, or accuracy of any other target unit.

4.1 Datasets

In this chapter, we will report results on two datasets.

Simulated vowels is a toy dataset that we created by sampling from a generative model that includes a simplified vocal-tract to sound nonlinearity. Since we know the exact underlying generating distributions, we can test the model capacity to recover these hidden generative factors given the realistic, observed signal.

TIMIT is a corpus of broadband read speech from 630 speakers from 8 major dialects of American English with time-aligned manual phonetic transcription and speaker labels (Lee & Hon, 1989). An often used benchmark, it allows comparison with large amounts of previously published results, for supervised and unsupervised phoneme and speaker recognition.

4.2 Model implementation

We implemented the most general MFL VAE model following the description in Section 3.6. With appropriate hyper-parameters to specify latent variables, prior type, number and location of filtering layers, this model can instanciate VAE models that include conventional, mixture, multimodal, multilevel and time-filtered variants described in previous sections.

Both encoder and decoder neural networks have one input layer followed by N hidden layers with nonlinearity, ending with a linear activation that generates mean and log variance parameters for the output distribution. Latent and observed variables are then sampled from the output distribution.

Hidden layers have all the same size for both the encoder and decoder. Nonlinear layers use leaky ReLU activation and residual connections from input to output, as well as batch normalization. Self-attention layers also use a residual connection. These elements were all found to be essential to convergence whenever more than one hidden layer was involved.

Filtering layers can be inserted between any two layers in the network, but only some locations were tested, in the encoder after the input layer and after each hidden layer, and after sampling the latent variable (before concatenation).

The model, training and evaluation scripts and analysis software were written in Python, with the help of PyTorch (Paszke, Gross, Chintala, Chanan, Yang, DeVito, Lin, Desmaison, Antiga & Lerer, 2017) and scikit-learn (Pedregosa, Varoquaux, Thirion, Grisel & Blondel, 2011) libraries.

Parametric distributions are diagonal multivariate Gaussian or Gaussian mixtures. They are implemented as Python classes with parameters for means, variances, and component weights in the mixture case. They provide methods to compute probability of an observation, to generate samples from the distribution using the reparameterisation trick (Kingma & Welling, 2014), and to evaluate gradients of these operations (when they exist).

4.3 Methodology

In our experiments, we evaluate informativeness (Eastwood & Williams, 2018) of the latent representations by using them in a downstream classification task. Classification accuracy measures the ability of the representation to cluster data into meaningful discrete groups. The better the representation maps to reference labels, the more interpretable it will be.

Classification accuracy can also be used to judge disentanglement, i.e., the degree to which a representation separates different types of information. In our case, a disentangled phoneme representation should have high phoneme classification accuracy but low speaker classification accuracy. Inversely, a disentangled speaker representation should have high speaker classification accuracy, but low phoneme classification accuracy.

Among many unsupervised classification algorithms that would be appropriate for the task, we use K-means as a simple unsupervised classifier, as it is well known and easy to reproduce and has been used in similar previously published work (Liu, Chen, Lee & Lee, 2018; Yeh *et al.*, 2018).

We report *framewise* phoneme accuracy rather than just phoneme accuracy. The latter is based on the number of correct and reference phoneme *segments*, and requires the frame label sequence to be converted into a sequence of phoneme segments, which is in itself a difficult problem. Some authors invoke a segmentation oracle (Yeh *et al.*, 2018), but since our objective is to evaluate representations rather than a whole phoneme recognition system, we choose not to include the segmentation problem in our evaluation, and report accuracy based on frames.

4.3.1 Accuracy computation

When using unsupervised K-means as the downstream task, classification results in anonymous cluster labels (modes). We do not know the relationship between cluster and reference labels, since cluster labels are subject to an arbitrary permutation. How then can we measure the classification accuracy?

The optimal solution examines all possible permutations (Dimitriadis & Fousek, 2017) and selects the one with the lowest Hamming distance. This solution has time complexity $O(n!)$ and quickly becomes impractical with more than a few classes. The Kuhn-Munkres algorithm (Munkres, 1957), also known as the Hungarian method, gives an optimal solution but has a time complexity $O(n^4)$. Here, we use a greedy algorithm to compute a suboptimal solution: we count all pairs of mode-vowel labels, and sort them by most frequent first; starting from the most frequent pair, we

associate the mode with the corresponding vowel, then use the second pair, etc. until all modes are associated. All unused pairs remaining count as errors. This suboptimal solution achieves results that agree closely with the optimal solution for practical values of n (Lewellen, 2017). Thus accuracy is computed from the results of this greedy cluster mapping as:

$$\text{Accuracy} = 1 - \frac{\text{Number of errors}}{\text{Total number of reference labels}} \tag{4.1}$$

4.3.1.1 Confidence interval estimation

For each result, a number of experiments are run to compute mean and 95% confidence interval for the mean, using Student's t-distribution, assuming samples are drawn independently from a normal distribution with unknown standard deviation. This confidence interval is unbiased for small samples (Young & Lewis, 1997).

4.3.2 Data preparation

We extract log Mel filterbank features with 40 channels from the audio utterances, with a frame advance of 10 ms. Another common practice is to use 40-dimensional MFCCs, but in the literature, not much difference is observed between these high resolution MFCCs and log Mel filterbank features, which is understandable since the first are a linear combination (through a discrete cosine transform) of the second. Log Mel filterbank features are also easy to display for visualizing reconstruction in the frequency domain.

"Spliced" features are obtained by concatenating the feature vector of a central frame and the vectors of the left N frames and right N frames. For example, filterbank features with 40 filters spliced with ± 2 frames result in a feature vector of dimension $5 \times 40 = 200$.

Sequences of similar lengths are collected into buckets and padded to one common length per bucket, to minimize the effect of padding and improve efficiency. Minibatches are created

on-the-fly, with random shuffling. Each minibatch is normalized to a zero mean and unit variance, as this was found to improve the results in preliminary experiments.

4.3.3 Training and evaluation

For each experiment, given a set of model hyper-parameters such as the number of latent variables, their dimensions, etc., we do SGD training with minibatches to find model parameters that will minimize the loss function, which is the negative of the ELBO as in Equation (3.32). Training is done for a fixed number of epochs, there is no early stopping. Each epoch represents going through the whole training dataset once.

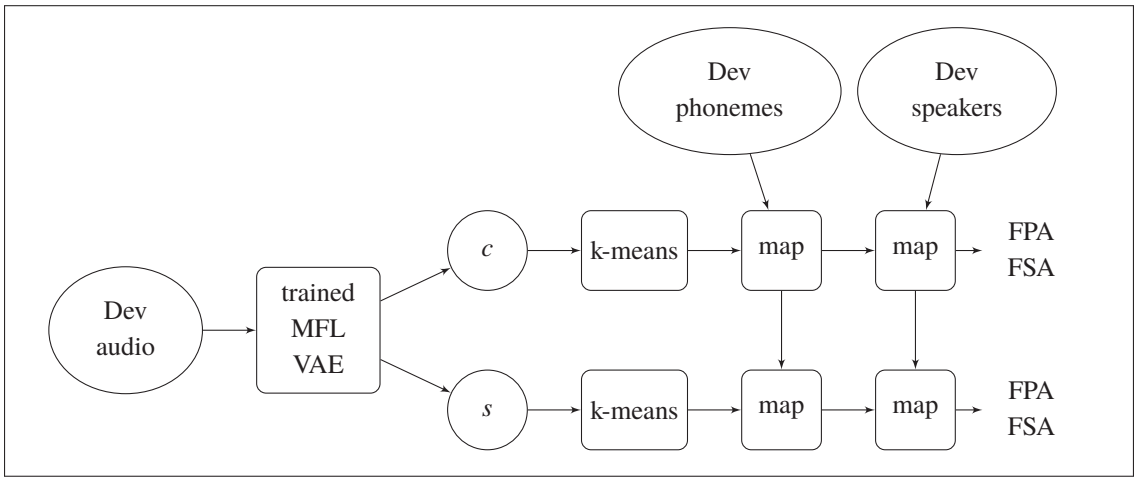


Figure 4.1 Evaluation of development FPA and FSA for two latent variables

Models are evaluated according to the process illustrated in Figure 4.1. An already trained model is applied on development audio to extract latent representations, such as the short-term c and long-term s variable shown in the figure. K-means clusters are trained on each representation[1]. Clusters are then mapped to phonemes for computing phoneme frame accuracy (FPA) or to speaker labels for computing speaker frame accuracy (FSA), using reference labels from the development set. Accuracy is the byproduct of mapping, as explained in Section 4.3.1.

[1] K-means is trained with a different number of clusters depending on which accuracy is being evaluated.

There are two experimental modes in which we use K-means to evaluate representations.

For **full evaluation**, each experiment consists of training with a fixed set of hyperparameters, on the full training set, with 10 different random initializations. The model with the lowest total loss on the training set is selected for evaluation. We compute *development frame accuracy* on the development set for this model, as illustrated in Figure 4.1. When reporting results, we use the average and confidence interval over 10 such experiments, all run with the same fixed set of hyperparameters. *Test frame accuracy* is obtained in the same way but using the test set.

For **hyperparameter tuning**, we hold out 5% of the training set as a validation set, and train a model with a given set of hyperparameters. We evaluate *batch frame accuracy* on the validation set after a fixed number of training epochs: a single validation minibatch is used to train K-means clusters and map them to the reference labels. Grid search is used to explore ranges of values for a few hyperparameters at a time. Hundreds of experiments can be run in this fashion to select architecture and hyperparameters. Only a small fraction of these are selected for full evaluation.

Selection of model architecture and hyperparameters is based on this validation set only. Although we agree with Locatello *et al.* (2018) that it is impossible to do truly unsupervised model selection, we believe that setting architecture and hyperparameters with such a small fraction of the training set, and transferring them "as is" to the development or test set for evaluation, does not constitute a significant supervision signal. The matter can only be resolved by testing the resulting architecture and hyperparameters on a variety of other datasets, which we are not yet in a position to do.

4.4 Experiments on simulated vowels

Our aim in these experiments is to simulate speech signals as realistically as possible, but in accordance with our main assumptions: discrete variables control short-term frame-level variations, while continuous variables define long-term, sequence-level variations. In addition, to be realistic, observations need to have a complicated distribution in a high-dimensional space, even if generated from discrete underlying units and low-dimensional continuous factors.

The source-excitation model of spoken vowel production provides us with a well-defined simulation path starting from a low-dimensional, discrete vowel description and ending with a complex, high-dimensional set of filterbank features.

4.4.1 Vowel signal generation

Human perception of vowel identity depends mainly on two properties of the speech signal called the F1 and F2 formants, which corresponds to frequencies of the first two vocal-tract resonances. Formant values do not directly correspond to peaks in the spectral envelope, as the envelope is a combination of the vocal-tract response with the glottal excitation, which produces peaks located at fundamental frequency harmonics. Measurements of formant values are made by visual examination of the spectrogram and auditory confirmation, since automatic identification is not entirely reliable (Labov *et al.*, 2006).

For a given vowel, formant values produced by a speaker are subject to some variation, but variability across speakers is even larger. Figure 4.2 illustrates how average values of F1 and F2 differ between individual speakers. Note that in the F1-F2 space, a vowel from a given speaker can sometimes overlap a different vowel from another speaker. A simple scaling of the frequency axes called log-mean normalization (Labov *et al.*, 2006) can be applied so that when plotted in this new two-dimensional space, vowels from different speakers are more concentrated in a single spot.

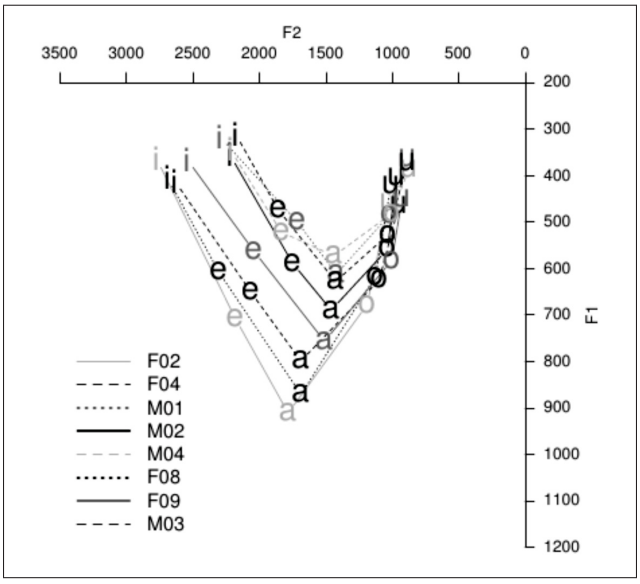


Figure 4.2 Observed vowel triangle for eight standard Indonesian speakers
http://drammock.github.io/phonR
(Retrieved, January, 2019)

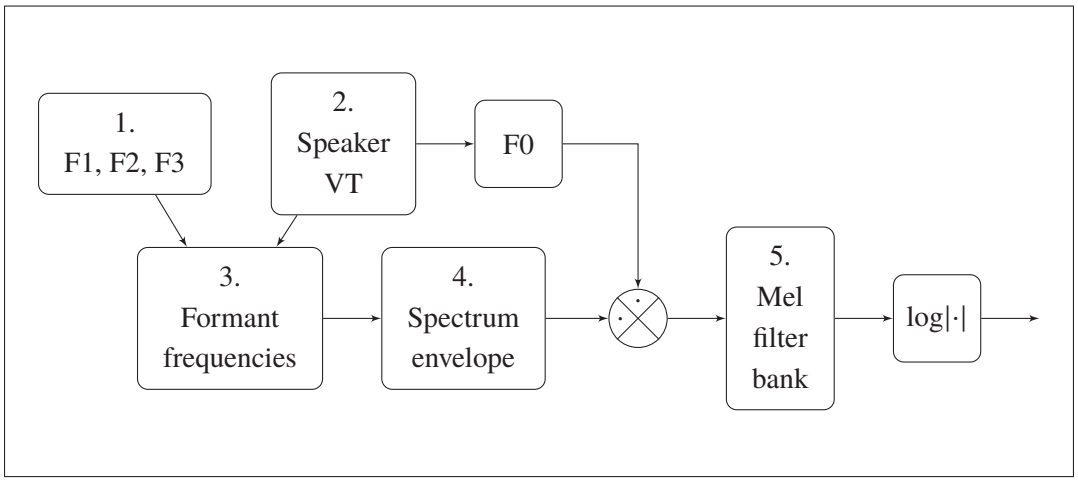


Figure 4.3 Steps of the vowel simulation process

We simulated speech data with natural-looking variations due to vowel identity and speaker characteristics using the processing flow graph depicted in Figure 4.3, with the following steps:

Step 1: Canonical formant frequencies are set for each of the 5 simulated vowels. Their values are listed[2] in Table 4.1. Formant bandwidths are fixed for all vowels at 100 Hz, 100 Hz, 200 Hz, and amplitudes at 1.0, 0.5, 0.2 for F1, F2 and F3, respectively.

Table 4.1 Formant frequencies of simulated vowels (Hz), step 1

Vowel	F1	F2	F3
/i/	240	2400	3500
/a/	850	1610	3500
/u/	250	595	3000
/ə/	585	1710	3200
/o/	500	700	3200

Step 2: For each speaker, a vocal-tract factor (VT) is generated with a value sampled from a uniform distribution in the interval [0.8, 1.2], which corresponds to a variation of ±20%, a range observed in actual data (Labov *et al.*, 2006).

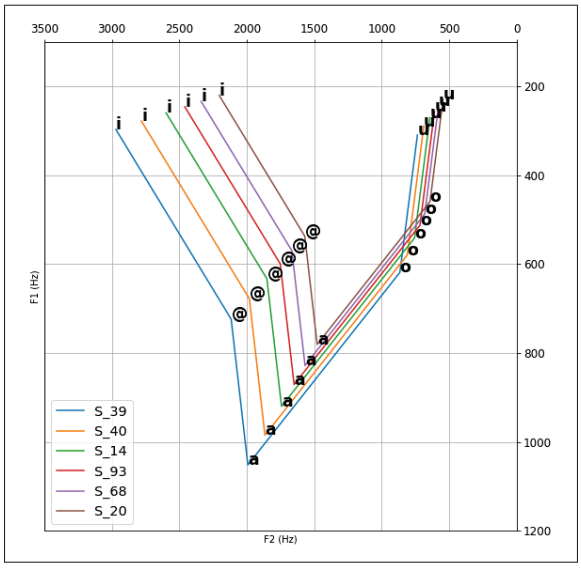


Figure 4.4 Simulated vowel triangle for 5 randomly chosen speakers, step 3

[2] From https://en.wikipedia.org/wiki/Formant

Step 3: For each vowel to be produced by a speaker, the canonical formant frequencies are scaled by the inverse of VT, simulating the inverse of log-mean formant normalization (Labov *et al.*, 2006). Resulting formant frequencies are shown in Figure 4.4 for 5 randomly chosen speakers. This figure can be compared to actual observed frequencies of Figure 4.2 (although for slightly different vowel set) and looks broadly similar.

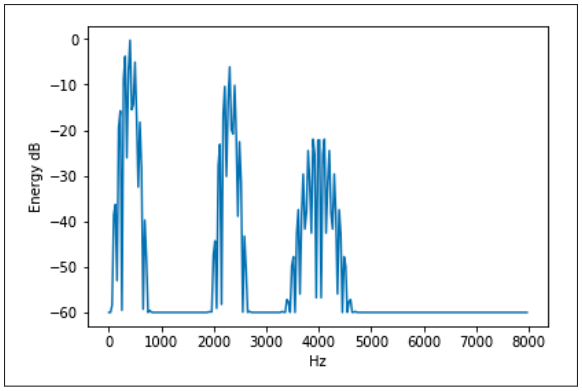


Figure 4.5 Modulated spectrum envelope over 512 FFT bins, step 4

Step 4: The formant bandwidths, amplitudes and frequencies for a given speaker and vowel are used to generate a power spectrum envelope over 512 FFT bins. We then multiply the spectrum envelope with the spectrum of a periodic excitation source. The fundamental frequency of this source is set to a value of 120 Hz scaled by the inverse of the VT factor, so that it is lower for male speakers and higher for female speakers. A typical result is shown in Figure 4.5. Notice how the periodic modulation due to the source creates a large number of narrow peaks within broad formant peaks.

Step 5: The 512 FFT bins are grouped into 40 filterbanks with Mel-spaced center frequencies, using standard speech signal processing software (Povey, et al., 2011). The end result is a 40-dimensional vector for each instance of a vowel by a particular speaker.

Step 6: The end result can be visualized as a spectrogram, arranging the 40 Mel-filterbank outputs in vertical columns and displaying each output as a gray-coloured square (from black for low energy to white for high energy). Such a display is shown in Figure 4.6, where the vertical axis represents filterbank index and horizontal axis represents time.

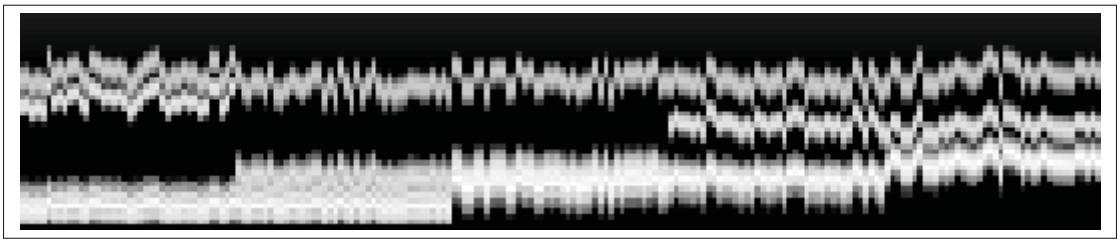


Figure 4.6 Simulated spectrograms of 40 mel-filterbank features. Five vowels ordered from left to right, with each vowel produced by 40 speakers, step 6

The left fifth of the display shows the first vowel uttered by 40 different speakers. Then the 2nd fifth shows the 2nd vowel, etc. until all 5 vowels of Figure 4.4 are produced.

4.4.2 Synthetic subsets

Synthetic vowels were generated for five vowels as described in the previous section. Table 4.2 shows the subsets generated while varying the sequence length L and number of speakers, and how each set was divided into training and development subsets. For each subset, a number of frames were synthesized as sequences of length L frames, each sequence uttered by a single speaker, in this case a single VT factor chosen at random in the $[0.8, 1.2]$ interval. Each frame in the sequence represents a different vowel selected at random among the five vowels. The reference label for the frame is the vowel used in generating the sample x for the frame.

Note that in these experiments, development sets were used solely as test sets, to measure accuracy after training. They were not used during training for early stopping or for selecting an initialization.

Table 4.2 Synthetic vowel datasets

Set	Train frames	Dev frames	N. sequences	L	N. speakers
vowels_1	30,000	1000	1000	1	1000
vowels_2	30,000	1000	100	10	100
vowels_3	30,000	1000	50	20	50
vowels_4	30,000	1000	20	50	20
vowels_5	30,000	1000	10	100	10

4.4.3 Model parameters

We experimented on simulated vowels with an early version of the full MFL model, with one or two latent variables, one discrete at the frame level, and a continuous one at the sequence level. For the frame-level variable, no filtering was applied (equivalent to an averaging filter of length one), and the sequence-level variable was averaged over the sequence (equivalent to pooling and tying for the whole sequence). This simplification is consistent with the assumptions underlying the vowel simulation: no speech dynamics are modelled, i.e., vowels parameters are switched abruptly between vowel instances, and the speaker factor is fixed for the sequence.

Table 4.3 summarizes hyperparameters and their default values used in experiments.

Table 4.3 Default parameters for experiments

Parameter	Description	Value
Model		
β	Frame-level KL weight	10.
β_2	Sequence-level KL weight	1
data dim	Input feature dimension	40
H	Hidden layer size in encoders and decoder	800
D	Latent variables dimension	2
K	N. of modes in multimodal prior	7
kl type	Type of KL used for multimodal prior	empirical
Training		
batch size	N. of sequences in minibatch	150
seq len	Length of sequences	20
n_{epochs}	N. times through the whole training set	1000
optimizer	Algorithm for gradient-descent optimization	Adam with $\beta_1 = 0.9, \beta_2 = 0.999$
other lr	General learning rate	1e-4
qz_2 *lr*	Learning rate for sequence level	1e-4
qz *lr*	Learning rate for frame level	2e-05
Prior distributions		
prior type	Prior distribution	GaussianMixture
init means	Geometry of means for multimodal prior	circular
spread	Spacing of means for multimodal prior	0.1

When used, the multimodal prior was a Gaussian mixture with K equal mixture weights, K means uniformly spaced in a circle of radius one, and K variances equal to the square of $spread$. Other possible arrangements for the multimodal mixture was to have means arrayed in a horizontal or vertical line, or along the vertices of a hypercube.

4.4.4 Results

The main results are reported in Table 4.4, in terms of mean accuracy and confidence interval, for the frame level variable. Accuracy is measured on the development set separately for each minibatch and averaged over one epoch[3]. For each result, 10 experiments are run and their individual results are averaged. The column ± reports the 95% confidence interval for the mean, using Student's t-distribution, assuming samples are drawn independently from a normal distribution with unknown standard deviation.

The first two rows of Table 4.4 correspond to the situation where only a frame level variable can be used, as frames are considered individually and not as part of a sequence. The batch framewise phoneme accuracy is much lower than for the following rows, where frames are part of a sequence and there is a sequence level variable.

When interpreting these results, we must keep in mind that the models are trained without supervision, and that latent variables have only two dimensions; in contrast, classification results in the VAE literature are often obtained with partial supervision and higher latent dimensions between 10 and 50 (Kim & Mnih, 2018; Kingma & Welling, 2014).

Our main findings from these preliminary experiments are:

- a sequence level variable is essential for disentangling speaker and vowel factors;
- a relatively large value of β is necessary, otherwise reconstruction loss dominates z KL divergence and we obtain good reconstruction but bad classification;
- a slower learning rate for z helps to slow convergence on suboptimal modes, so the rest of the model can settle first into a good reconstruction;

[3] This is different from the development FPA reported elsewhere in this thesis.

Table 4.4 Multimodal VAE accuracy with and without sequence level variable

Sequence length	β	Batch FPA
1	1.0	0.770 ±0.050
1	10.0	0.493 ±0.090
20	1.0	0.900 ±0.047
20	5.0	0.933 ±0.055
20	10.0	0.991 ±0.060
10	10.0	0.972 ±0.046
50	10.0	0.983 ±0.017

- there is no significant difference between pooling the hidden layer or the output layer of the neural net that computes $q_{\phi_s}(s|x)$.

Posterior distributions obtained with the model are illustrated in plots such as the one in Figure 4.7. The plot contains 6 sub-figures. The top row illustrates the model input. The upper-left is a spectrogram representation of a batch of test data, with five vowels arranged from left to right, each one represented by many examples spoken by different speakers. The upper-middle shows samples from the 2-dimensional multimodal prior $p(z)$[4], and the upper-right shows samples from the 2-d normal prior $p(z_2)$. The colours represent true vowel identities.

The bottom row shows the model output. The bottom left is a spectrogram reconstructed by sampling from the model posterior distribution $p(x|z)$. Reconstruction accuracy can be assessed by comparing the top and bottom spectrograms. The bottom middle shows samples of the frame-level latent variable z taken from the estimated posterior distribution $q(z|x)$. Each sample is coloured by the ground-truth vowel label so we can see how the modes of $q(z|x)$ correspond to underlying vowels. Finally, the bottom right is a plot of z_2 sequence-level latent variable samples from $q(z_2|x)$, coloured by the sequence id, or equivalently, speaker label since each sequence is generated by a different speaker.

[4] The number of mixture components must be set at minimum to the number of modes expected to be found, but can be set slightly higher without affecting the results, as shown in this example.

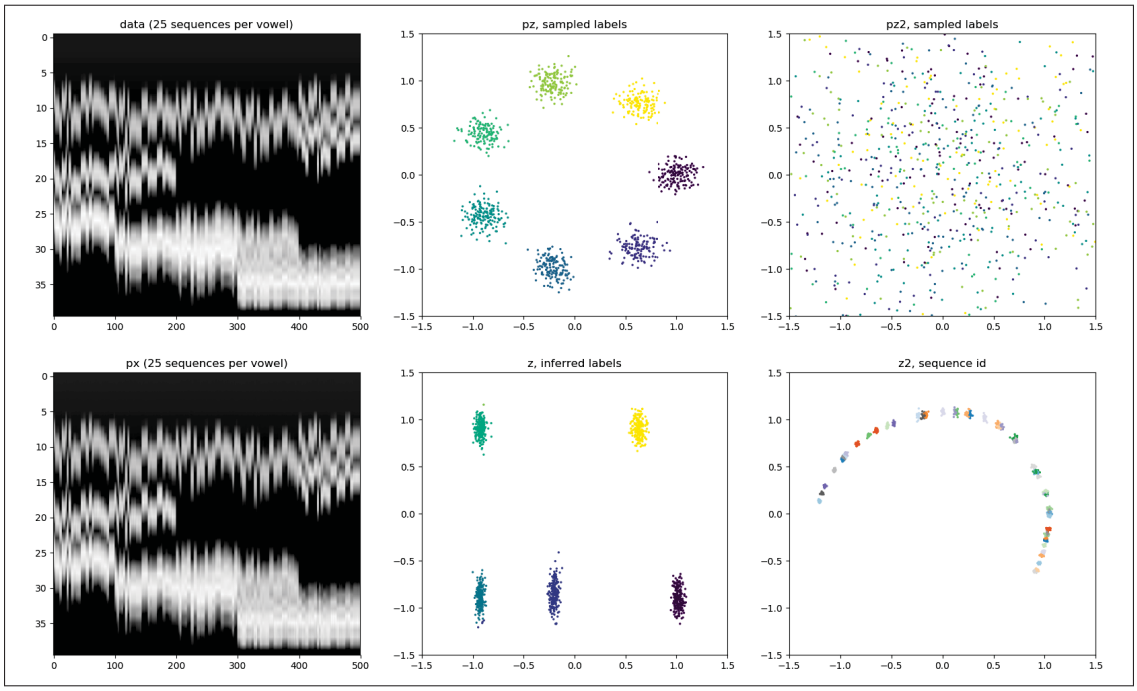
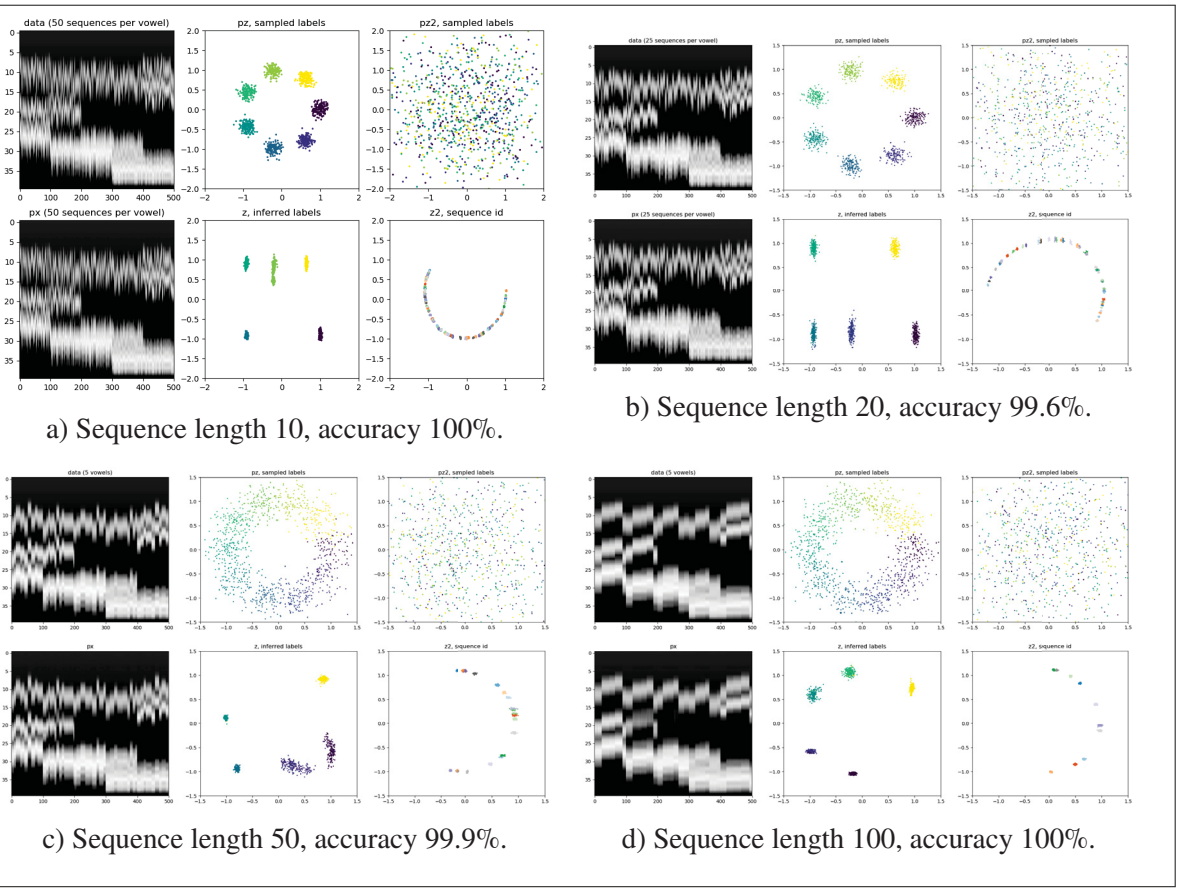


Figure 4.7 Sequential VAE on vowels_4 test set, sequence length 20, accuracy 99.6%

This figure shows how frame-level variable z does indeed capture vowel identity, as the accuracy of 99.6% demonstrates. This accuracy is reported for the displayed development minibatch. The sequence-level variable z_2 is distributed along a continuous, one-dimensional manifold, as we would expect: in our simulation, speaker identity depends on VT, a single, continuous, one-dimensional factor. A more magnified view would show that different vowels from the same speaker are clustered together along the manifold.

Figure 4.8 shows various outcomes of the model with various spreads in the prior modes, and different sequence lengths. In each case, the z_2 variable was able to capture the single-dimension, continuous factor that represents speaker variation. Frame-level z clustered along five distinct modes, even though some are elongated where speaker variation crept in.

Multimodal VAE model without sequential modelling ($\beta_2 = 0$ and fixed z_2) plots appear in Figure 4.9 and show that z does not cluster into well-delimited modes, as it tries to account for



a) Sequence length 10, accuracy 100%.

b) Sequence length 20, accuracy 99.6%.

c) Sequence length 50, accuracy 99.9%.

d) Sequence length 100, accuracy 100%.

Figure 4.8 Sequential VAE on vowels_2, 3, 4, 5 test sets

both speaker and vowel factors. The accuracy suffers accordingly, and reconstruction is also badly affected.

Finally, conventional VAE plots (with $\beta = 0$ and fixed z) are shown in Figure 4.10. As for the non-sequential multimodal model, both sequence and frame-level factors are entangled within the representation offered by the single latent variable. Note here that accuracy is meaningless since classes are derived from the z variable which is not trained in this case.

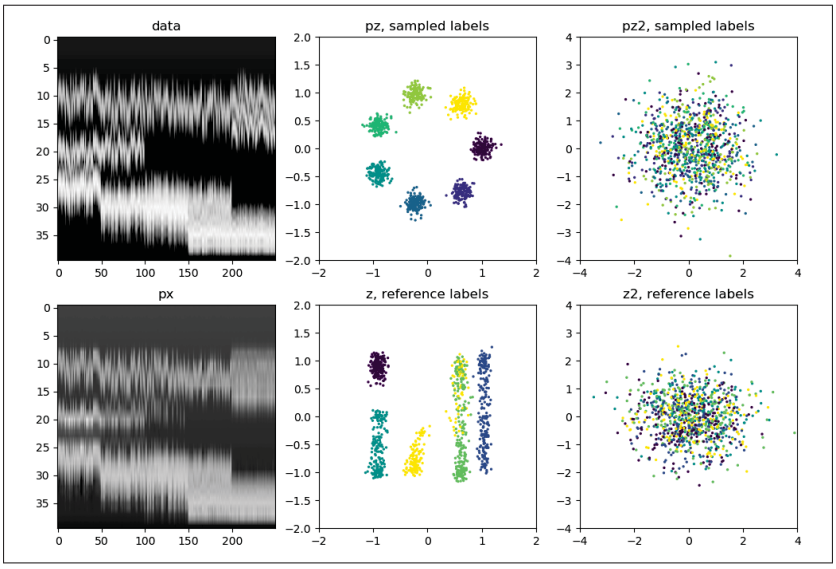


Figure 4.9 Multimodal VAE on vowels_4 test set, no sequence modelling, accuracy 76.8%

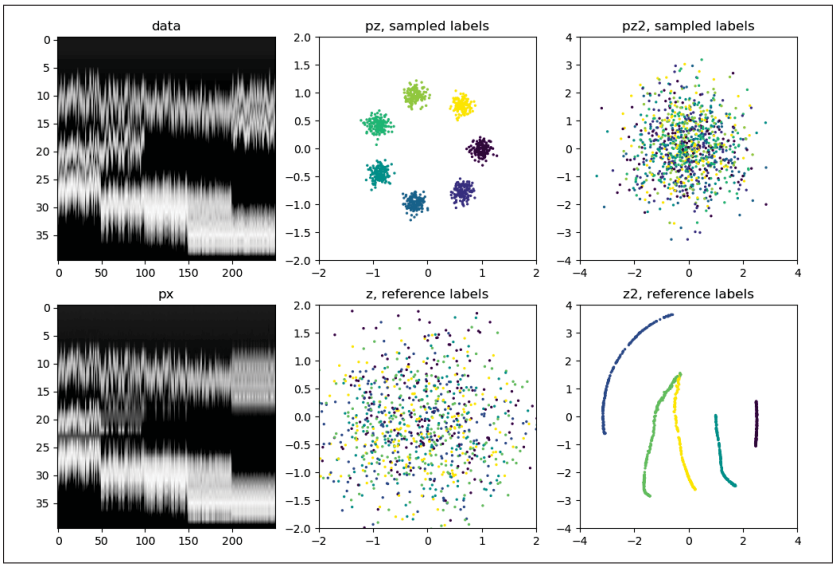


Figure 4.10 Conventional VAE on vowels_4 test set, no sequence modelling, accuracy 28.5%

4.5 Experiments on TIMIT

We use TIMIT as the main corpus for our next series of experiments, since it provides utterances labelled with phonemes and speakers. Because it is read speech recorded at 16 kHz in clean conditions, it represents a small step towards more realistic speech, without introducing additional problems due to acoustic conditions (noise, reverberation), spontaneous speech (hesitations, repairs) or conversation (overlaps, background channel). Manual time-aligned phonetic transcriptions and speaker labels allow us to compute accurate framewise measures. In addition, there are large amounts of published results on TIMIT, for supervised and unsupervised phoneme and speaker recognition.

4.5.1 Data preparation

The dataset is split following the standard TIMIT partition (Lee & Hon, 1989) into a training set of 3696 utterances and 168 speakers, a full test set of 1344 utterances and 168 speakers, and a core test set of 192 utterances and 24 speakers. As is usually done, a development set of 400 utterances is also derived from the full test set, according to the Kaldi TIMIT recipe[5]. Speakers do not overlap between the training, test and development sets. We further split the training set by holding out 5% of the training utterances to use as a validation set, for small experiments. All the results we report here were obtained by training on the full training set and evaluating on the development set; any hyperparameter tuning was performed using only the small held-out validation set.

We extracted log Mel filterbank features with a standard configuration in Kaldi recipes: no energy, 8 filters/octave which yield 40 filters (for a 16 kHz sampling frequency), and splicing with ±2 or ±5 frames on the left and right.

[5] https://github.com/kaldi-asr/kaldi/tree/master/egs/timit/s5

Each utterance in TIMIT varies in duration from 0.91 to 7.8 seconds, and corresponds to a sequence having from 91 frames to 780 frames. Using bucket boundaries of (99, 210, 260, 290, 310, 340, 400) resulted in 6 buckets with roughly same number of utterances.

As is customary for TIMIT since Lee & Hon (1989), modelling is done using 48 phoneme classes, but accuracy is measured after mapping these 48 classes to a smaller 39 phoneme inventory.

To compute framewise phoneme classification accuracy, we obtain the phoneme label for each frame from manual phoneme transcriptions included in TIMIT, mapped to the 39 phoneme set. For speaker accuracy, we use utterance speaker labels to assign the same speaker label to all frames in an utterance.

For tuning, we use "automatic" framewise phoneme labels from a GMM-HMM recognizer trained following the standard Kaldi recipe for TIMIT[6], up to the "tri3" models (LDA + MLLT + SAT). These models yield a phoneme error rate of 20.7% on the dev set and 21.6% on the test set. They were used in a forced alignment using only the phoneme labels from the TIMIT transcriptions provided.

4.5.2 Baseline results

Table 4.5 Frame-wise accuracy for baseline phoneme and speaker classifiers on TIMIT development and test sets, with filterbank features spliced ±2

Classifier	Dev FPA	Dev FSA	Test FPA	Test FSA
Random	0.121 ±0.003	0.021 ±0.004	0.118 ±0.004	0.041 ±0.004
Single	0.121 ±0.003	0.020 ±0.005	0.118 ±0.005	0.038 ±0.008
K-means	0.362 ±0.002	0.122 ±0.002	0.339 ±0.003	0.151 ±0.004
Random (480, 500)	0.121 ±0.003	0.023 ±0.003	0.118 ±0.004	0.050 ±0.004
Random (4800, 5000)	0.122 ±0.003	0.040 ±0.002	0.123 ±0.004	0.079 ±0.003
Random (48000, 50000)	0.158 ±0.003	0.103 ±0.004	0.199 ±0.009	0.187 ±0.011
Random (480000, 500000)	0.414 ±0.026	0.393 ±0.029	0.611 ±0.037	0.618 ±0.037
Random (4800000, 5000000)	0.888 ±0.008	0.889 ±0.008	0.945 ±0.007	0.946 ±0.006

[6] See footnote 5.

Table 4.5 gives results obtained with trivial or simple unsupervised classifiers. Even though the development set is larger, and was not used in tuning, we report results for both development and test sets for easier comparison with other published results.

The **random classifier** produces a random label for each frame. For phonemes, there are 48 possible labels, which are mapped to the 39 phoneme set using the greedy mapping described in Section 4.3.1. There are 50 distinct speakers in the development set, 24 in the test set, and each speaker contributed 8 recordings in both sets. For K-means, we used 100 clusters for development and 48 clusters for test.

The **single** classifier predicts the same label for all frames, this label being chosen as the most frequent in the development or test set.

Note that naively, with 39 phonemes to predict, one would expect a random guess accuracy of $\frac{1}{39} = 0.026$ and similarly, with 50 distinct speakers to predict, one would expect a random guess accuracy of $\frac{1}{50} = 0.02$ on the development set. However, recall that cluster labels are mapped to actual labels using the greedy algorithm, Section 4.3.1; so by mapping all random guesses to the most frequent reference phoneme, it is always possible to achieve at least the same accuracy as the single classifier. In contrast, the number of speakers is balanced so speakers are equally likely. So both random and single best classifier achieve very close batch and development accuracies, around 0.121 for phonemes and 0.021 for speakers.

The **K-means** classifier is trained on Mel filterbank features, with a splicing context of ±2. These features of dimension 200 are projected with LDA to a dimension of 20, which provides the best accuracy on the development set.

The **best unsupervised** framewise phoneme accuracy for TIMIT is reported by Yeh *et al.* (2018), who describes a fully unsupervised model evaluated on the same TIMIT test set, using the same 39 phoneme inventory. The model is obtained in a fully unsupervised fashion, using no oracle phoneme boundaries, but uses a language model. With the weaker "non-matching LM" derived from an held-out third of the training sentences, the reported FPA is 0.484.

So the K-means classifier with filterbank features is not so trivial a baseline, as it achieves an accuracy of 0.339 while the best reported mostly unsupervised model gets 0.484.

To put these baselines into perspective, some **supervised** framewise phoneme accuracies have also been published on the same TIMIT dataset. For example Shulby, Ferreira, de Mello & Aluisio (2019) gives 0.570 FPA for an hybrid CNN-MLP model, and 0.580 for an hybrid MLP-HMM model (both using a 61 phoneme set).

The second part of Table 4.5 shows random classifiers with 480 to 4.8M phoneme classes, and 500 to 5M speaker classes, to illustrate how error rate can be reduced arbitrarily as the number of classes is increased. In the limit, each of the development set 1.38 M frames is assigned a separate class that can be mapped to the correct reference. This is in contrast to Chorowski *et al.* (2019), who uses a similar mapping to compute frame accuracy, and claims that increasing the number of classes does not trivially improve accuracy. Here we have empirical support to show that increasing the number of classes does artificially improve accuracy, and makes all errors vanish in the limit. However, the effect is small until the number of classes reaches about half the number of frames in the dataset.

4.5.3 Experimental results

In this section, we evaluate the impact of our proposed improvements over previous conventional and β-VAE models. In particular, we will see how well time-filtering and multiple latent variables allow representations to disentangle phoneme and speaker information. Finally, we will report the effect of self-attention in both encoder and decoder.

Figure 4.11 provides an example of representations obtained on the TIMIT developement set by an MFL VAE model with short-term c and long-term s continuous latent variables (only the first two dimensions are plotted).

Recall that in the tables that will follow, each entry is generated by running 10 experiments with a given set of fixed hyperparameters, in order to compute a confidence interval. In each of these

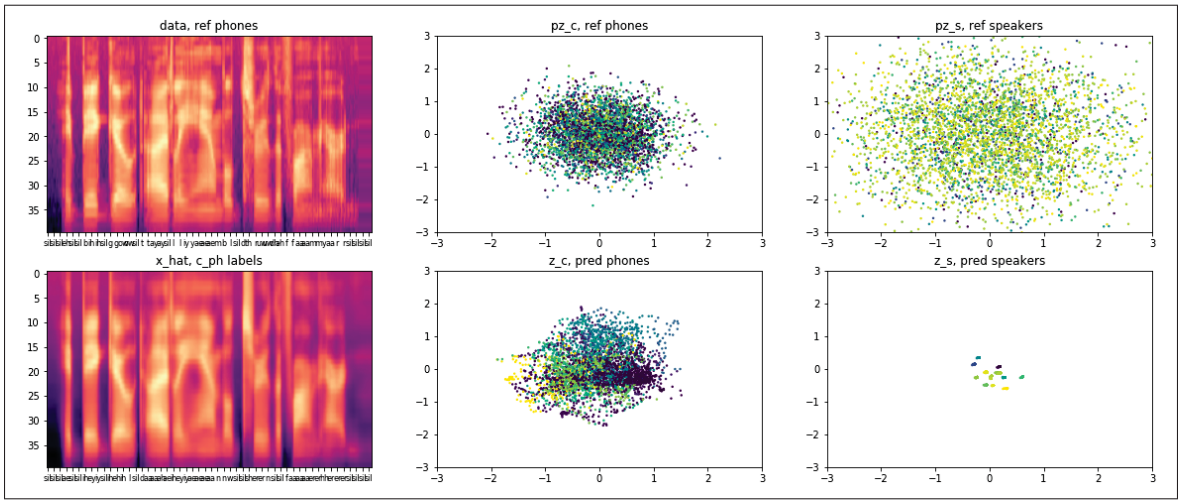


Figure 4.11 Modelling TIMIT dev set with frame and sequence variables c and s. Top row, from left to right: Mel filterbank features, c prior and s prior. Bottom row, from left to right: reconstructed signal, c samples colored by phoneme, s samples colored by speaker

experiments, 10 models are trained from different initializations, and the model with lowest total *training loss* is selected for evaluation. Evaluation consists of using the model to generate representation vectors for the whole development or test set. We then train a K-means classifier on the set of representations, unsupervised, and use the corresponding clusters to label the batch or development set. We compute the framewise phoneme and speaker accuracy of the cluster labels against the reference labels with the greedy algorithm mentioned in Section 4.3.1.

4.5.3.1 Time filtering

The first series of experiments compares conventional and β VAE, which both have a single latent variable with a Gaussian prior, and the effect of filtering as a bias to induce phoneme and speaker representation.

Table 4.6 reports the development and test framewise phoneme and speaker accuracies. In the first row are the K-means baseline results from Table 4.5 for reference. Column "Dim" shows the latent variable dimensions that yields the best FPA in the tuning phase, expressed as a tuple where each element is the dimension of one latent variable. Column "Filter" shows the filter

Table 4.6 Plain VAE and β-VAE representations and the effect of filtering on TIMIT development set, filterbank features spliced ±2

Model	Dim	Filter	Dev FPA	Dev FSA	Test FPA	Test FSA
K-means	-	-	0.362 ±0.002	0.122 ±0.002	0.339 ±0.003	0.151 ±0.004
VAE	15	0	0.178 ±.002	0.037 ±.000	0.178 ±.001	0.067 ±.000
β-VAE	15	0	**0.387 ±.004**	0.148 ±.002	**0.371 ±.003**	0.209 ±.003
VAE+f	15	4	0.381 ±.003	0.167 ±.003	0.355 ±.004	0.250 ±.008
β-VAE+f	15	2	**0.387 ±.004**	0.153 ±.003	**0.369 ±.004**	0.219 ±.003
VAE+f	40	500	0.166 ±.004	0.579 ±.011	0.182 ±.002	0.691 ±.016
β-VAE+f	40	500	0.173 ±.003	**0.694 ±.020**	0.191 ±.003	**0.823 ±.016**

used in each experiment, also as a tuple where each element is the filter length, in frames, for one latent variable. Only hyperparameters of interest are listed here; the other hyperparameters are from the default[7] configuration. In each column, bold characters emphasize the best value. Note that there may be several equivalent, best values, when they fall within the confidence interval of each other.

Conventional vs β-VAE: In Table 4.6, VAE with filter length 0 corresponds to the conventional VAE model of Kingma & Welling (2014), with a Normal prior, $\beta = 1.0$, and no filtering. Its FPA and FSA values do not even reach the K-means baseline, so as a representation for phoneme or speaker, it is not even as good as filterbank features used in baseline.

β-VAE with filter length 0 has $\beta = 0.1$ and corresponds to a β-VAE model (Burgess *et al.*, 2017; Higgins *et al.*, 2017). It yields better FPA than the K-means baseline in Table 4.5 but with a Dev FSA of less than 0.2, it still fails to capture significant speaker information.

Time-scale filtering: The following rows VAE+f and β-VAE+f in Table 4.6 show the impact of inserting an averaging filter of 4 and 2 frames after the latent variable sampling stage. For VAE+f and filter length 4, phoneme accuracy surpasses the baseline K-means FPA, while for β-VAE+f with filter length 2, FPA is also improved over its non-filtered version β-VAE.

[7] For VAE with no filter, only 3 hidden layers were used, since convergence was not possible with 4.

The last two rows of Table 4.6 show that a filter of 500 frames dramatically improves speaker representation, as shown by FSA values. However, a long filter improves speaker accuracy at the expense of phoneme accuracy: FPA is dramatically reduced compared to the short filter FPA for both VAE+f and β-VAE+f. Clearly, with a single latent variable, there is a trade-off between phoneme and speaker accuracy.

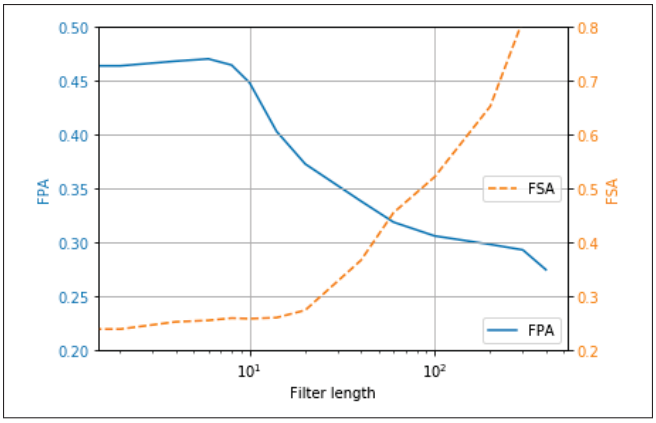


Figure 4.12 Batch FPA and FSA on the TIMIT validation set with varying filter lengths

This trade-off is illustrated more clearly in Figure 4.12, which plots FPA and FSA on a validation batch, as a function of filter length. A length of 1 (no filtering) is not the best for FPA, which peaks at intermediate lengths between 6 and 8. The best FSA is obtained with a length of 500, which is longest than the average TIMIT utterance. There is no value of filter length that is good simultaneously for FPA and FSA.

Location of filtering layer: The filtering layer can be located just after the input layer, after each of the hidden layers, or right after sampling the latent variable (see Section 3.5.2). This can have a large impact on the performance, especially for speaker accuracy. For experiments reported in Table 4.6, the best location was found to be after sampling. However, this was also true for most other model variants and conditions that we tested. As the filtering layer is moved closer to the input (from after sampling to the 2nd hidden layer), accuracy is slightly degraded, similarly for both latent variables.

In summary, we confirm, as reported in previous studies, that β-VAE produces better representations than its conventional VAE counterpart. The new finding here is that time-scale filtering improves representations even more, especially for speakers. A *short* filter makes the VAE latent variable a better low-dimensional representation for phonemes compared to filterbank features. A *long* filter makes the VAE latent variable a better low-dimensional representation for speakers. However, there is a trade-off which prevents the VAE latent variable from being good at representing both phoneme and speaker information at the same time.

4.5.3.2 Multiple latent variables

Table 4.7 examines the effect of using multiple, filtered latent variables, as in our proposed MFL-VAE model. In this table, for MFL-VAE, FPA is reported only for the short-term latent variable, and FSA only for the long-term variable, for easy comparison with previous β-VAE+f results shown in the first two rows, where both FPA and FSA are reported for the single variable[8].

Table 4.7 Multiple variables modelling on TIMIT development set, filterbank features spliced ±2

Model	Dim	Filter	Dev FPA	Dev FSA	Test FPA	Test FSA
β-VAE+f	15	2	0.387 ±.004	0.153 ±.003	0.369 ±.004	0.219 ±.003
β-VAE+f	40	500	0.173 ±.003	0.694 ±.020	0.191 ±.003	0.823 ±.016
MFL-VAE	10, 40	6, 500	**0.392 ±.003**	**0.950 ±.009**	**0.375 ±.004**	**0.986 ±.004**

For MFL-VAE, multiple latent variables are modelled jointly (as described in Section 3.6). A dimension of 10, with a filter length of 6 frames is used for the "phoneme variable", and a dimension of 40, with a filter length of 500, for the "speaker" variable, as shown in the Dim and Filter columns. Note that from the modelling point of view, the two latent variables are treated exactly the same, except for their dimension and filter length, so the names "phoneme" and "speaker" variables are used only for convenience.

[8] Dev FPA is 0.161 for the long-term variable and Test FSA 0.132 for the short-term variable.

Each accuracy in MFL-VAE can be compared to the best accuracy in the same column, in the first and second rows of the table, which report the best single latent variable models from Table 4.6. The accuracy of each variable in MFL-VAE is better than the best accuracy of any individual model, showing that both representations benefit from the joint modelling. This is especially marked for speaker classification: Dev FSA is improved from 0.694 to 0.950 by MFL-VAE, and Test FSA from 0.823 to 0.986. FPA slightly improves both for development and testing.

Table 4.8 Multiple variables modelling on TIMIT development set, filterbank features spliced ±5

Model	Dim	Filter	Dev FPA	Dev FSA	Test FPA	Test FSA
β-VAE+f	15	6	**0.397 ±.003**	0.159 ±.002	0.375 ±.003	0.230 ±.007
MFL-VAE	10, 40	4, 500	**0.398 ±.004**	**0.948 ±.008**	**0.388 ±.004**	**0.972 ±.012**
MFL-VAE+d	8, 40	8, 500	0.382 ±.011	**0.941 ±.032**	**0.382 ±.017**	**0.972 ±.039**

Size of receptive field: Table 4.8 presents results for a larger input receptive field. Input features were the concatenation of the central frame plus the preceding and following 5 frames, resulting in a receptive field of 11 frames. The second row confirms the MFL-VAE results observed with ±2 splicing (Table 4.7) but with higher phoneme and comparable speaker accuracy.

Target of reconstruction: when splicing with ±2 or ±5 frames, one may wonder if the reconstruction loss should be based only on the central frame or on the whole receptive field. We found that a 3-frame window, including the central frame plus the left and right frame, worked best. We did not obtain better results with a target reconstruction field offset in the future relative to the input field, even with overlap.

Multimodal prior: the last line of Table 4.8 shows the best result we could obtain using a multimodal prior on the "phoneme" variable, despite a lot of experimentation with hyperparameters. The best FPA result was obtained with a Gaussian mixture of 64 components, arranged in an 8-dimensional hypercube. Surprisingly, phoneme accuracy is degraded relative to a continuous variable, but speaker accuracy remains as good as the model with a unimodal short-term prior.

4.5.3.3 Self-attention

Table 4.9 explores the use of attention layers, which we introduced with the goal of modelling sequences of short-term representations. Column "Heads" shows the number of heads used in the self-attention layer, expressed as a tuple where each element is the number of heads for one latent variable. The first two rows (1 and 2) are taken from the best β-VAE+f and MFL-VAE

Table 4.9 MFL-VAE with self-attention, on TIMIT development set, filterbank features spliced ±5

	Model	Heads	Dev FPA	Dev FSA	Test FPA	Test FSA
1	β-VAE+f	0	0.397 ±.003	0.159 ±.002	0.375 ±.003	0.230 ±.007
2	MFL-VAE	0, 0	0.398 ±.004	**0.948 ±.008**	0.388 ±.004	0.972 ±.012
3	β-VAE+f	1	0.393 ±.005	0.162 ±.003	0.370 ±.003	0.247 ±.004
4	MFL-VAE	1, 0	**0.409 ±.003**	0.927 ±.015	0.383 ±.003	0.976 ±.010
5	MFL-VAE	6, 0	0.405 ±.005	0.926 ±.009	0.386 ±.005	0.978 ±.007
6	MFL-VAE	6, 6	0.406 ±.004	0.934 ±.013	0.387 ±.005	0.983 ±.007
7	MFL-VAE	6, 12	**0.408 ±.003**	0.933 ±.010	**0.392 ±.005**	**0.987 ±.003**

models in Table 4.8, for reference. Adding one attention head to the single variable of β-VAE+f does not improve the results (row 3 compared to row 1). Adding 1 or 6 heads to MFL-VAE (rows 4 and 5) improves FPA, but slightly degrades FSA. The degradation in FSA is somewhat compensated by also adding attention to the speaker variable, with 6 or 12 heads (rows 6 and 7).

Even though self-attention obtains the best Dev FPA results reported in this thesis, its performance is not consistent across development and test. It seems that adding attention to both variables is important to avoid degrading FSA. Positional sine/cosine encoding, in the form proposed by Vaswani *et al.* (2017), was tested both with sum and concatenation but was found to degrade the results. Restricting attention to a local context around the current frame (Povey *et al.*, 2018) was also tried but did not improve the results.

4.5.4 Visualizations

Figures 4.13 and 4.14 shows 2-dimensional projections. On the left, raw filterbank features, and right, one of the latent representations from the model from row 7 of Table 4.9. The projections are obtained with t-SNE (van der Maaten & Hinton, 2008). A random subset 10% of TIMIT development is plotted; each point is one frame. For clarity silence frames are not plotted.

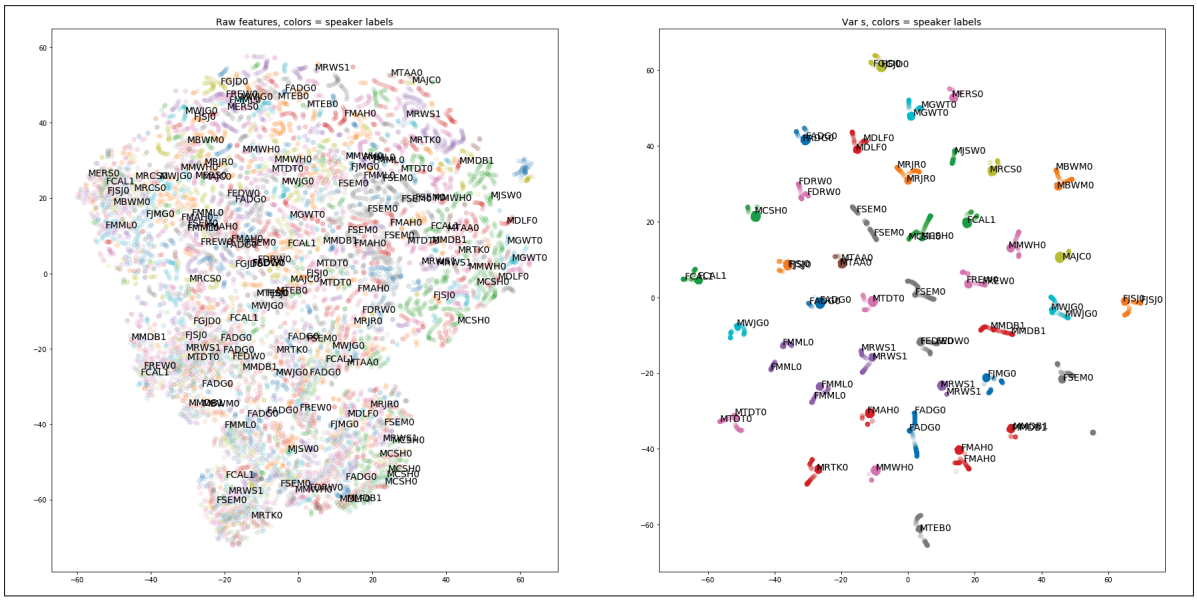


Figure 4.13 t-SNE projection of raw features (left) and *s* representation. Each point is one frame of 40 utterances from TIMIT development set. Colors represent speakers

Figure 4.13 shows projected filterbank features on the left and *s* representation on the right, both colored by reference speaker label. Speakers in the *s* representation are located in well separated clusters, within which micro-clusters corresponding to individual sessions are visible. With raw features, each speaker is largely distributed over all the representation space. This is not surprising, given that a K-means classifier only obtains a development FSA of 0.122 with filterbank features, compared with 0.933 for the *s* representation.

Figure 4.14 compares the filterbank features (left) with the *c* representation (right), this time colored by reference phoneme class[9], either vowel, stop, fricative, approximant or nasal. The

[9] Following TIMIT documentation, except jh,ch were considered as fricatives rather than affricates.

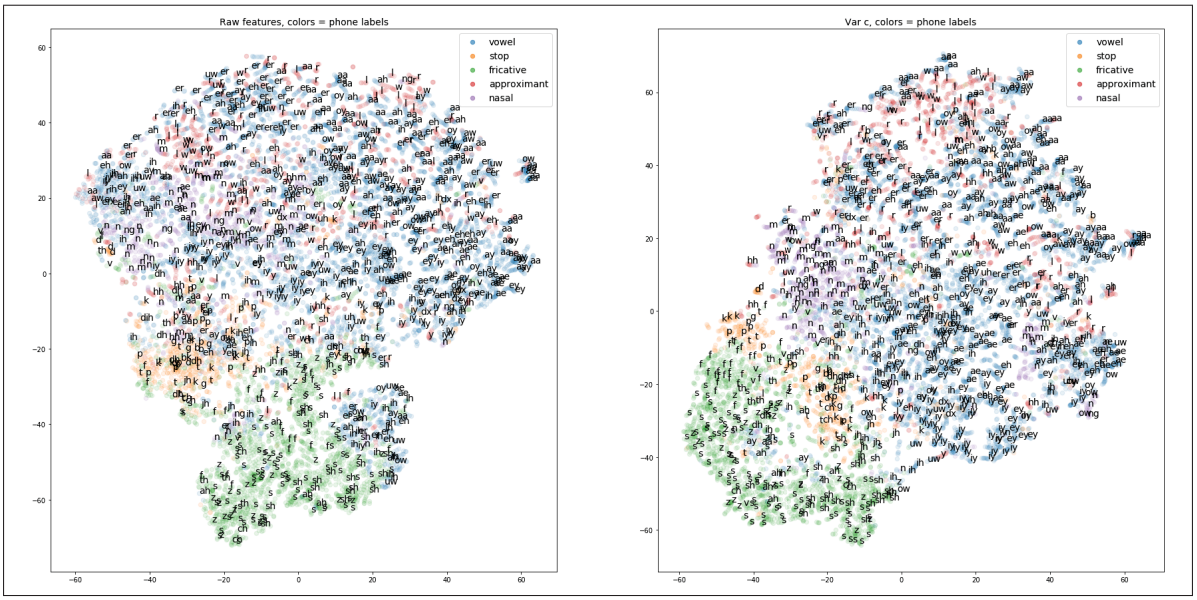


Figure 4.14 t-SNE projection of raw features (left) and c representation. Each point is one frame of 40 utterances from TIMIT development set. Colors represent phonemes

difference is not expected to be as large as for the s representation: Dev FPA is 0.362 for filterbank features and 0.408 for c. Indeed, fricatives (green) and stops (orange) form distinct clusters in both representations. However, in c, nasals (magenta) are better grouped; approximants (red) and vowels (blue) are more separated, while in the filterbank representation they are mixed together.

A phoneme confusion matrix obtained on one minibatch of the TIMIT development set is shown in Table 4.10. Each row is a phoneme obtained from the short-term representation of the model, each column is a reference phoneme from manual transcription. Each entry represents how many times the model phoneme was correctly mapped to the reference phoneme, as a fraction of all occurrences of the model phoneme (rows sum to 1). Correct mappings are highlighted in blue. Only phonemes with entries > 0.18 are shown individually, the others are aggregated in the last column, under the name "rest".

Most frequent confusions can be found by identifying the largest non-highlighted number in a given row. Vowels are more frequently confused with other vowels from a similar class, for

example /ae/ and /eh/, or /iy/ and /ih/. Among the more frequent confusions, we find /k/ with /f,sil,t/, /m/ with /n/, /s/ with /z/, and /sh/ with /s/. This is anectodal evidence but it appears that even when the representation is mapped to the wrong reference by K-means, it is still close to a phonetically related neighbor.

Table 4.10 Phoneme confusion matrix, TIMIT dev set, model 7 from Table 4.9

	aa	ae	eh	er	f	ih	iy	k	l	m	n	r	s	sh	sil	t	w	z	rest
aa	0.29	0.06	0.03	0.02	0.00	0.02	0.00	0.00	0.09	0.01	0.01	0.03	0.00	0.00	0.02	0.00	0.02	0.00	0.41
ae	0.04	0.26	0.23	0.01	0.00	0.08	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36
er	0.05	0.03	0.04	0.39	0.00	0.05	0.01	0.00	0.00	0.02	0.02	0.23	0.00	0.00	0.02	0.00	0.01	0.00	0.14
f	0.00	0.00	0.00	0.01	0.54	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.09	0.02	0.06	0.03	0.00	0.04	0.17
ih	0.01	0.06	0.09	0.04	0.01	0.25	0.05	0.00	0.02	0.01	0.06	0.03	0.02	0.00	0.02	0.01	0.00	0.02	0.30
iy	0.00	0.02	0.02	0.01	0.00	0.15	0.40	0.00	0.01	0.02	0.05	0.01	0.00	0.00	0.04	0.00	0.00	0.00	0.27
k	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.23	0.01	0.01	0.00	0.01	0.02	0.00	0.12	0.06	0.01	0.00	0.37
l	0.05	0.00	0.01	0.01	0.00	0.06	0.01	0.00	0.35	0.05	0.05	0.00	0.00	0.00	0.04	0.00	0.05	0.00	0.31
m	0.01	0.01	0.01	0.01	0.00	0.04	0.01	0.00	0.01	0.33	0.26	0.02	0.00	0.00	0.09	0.00	0.01	0.00	0.19
n	0.00	0.00	0.01	0.01	0.00	0.07	0.03	0.00	0.01	0.10	0.57	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.16
s	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.61	0.02	0.02	0.01	0.00	0.26	0.04
sh	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.27	0.41	0.01	0.01	0.00	0.11	0.15
sil	0.00	0.00	0.00	0.00	0.01	0.02	0.00	0.03	0.00	0.01	0.04	0.00	0.02	0.00	0.65	0.05	0.00	0.02	0.13
t	0.00	0.00	0.01	0.00	0.05	0.02	0.00	0.13	0.00	0.00	0.00	0.00	0.11	0.08	0.02	0.23	0.00	0.04	0.29
w	0.13	0.00	0.01	0.02	0.01	0.00	0.00	0.01	0.18	0.02	0.01	0.01	0.00	0.00	0.06	0.00	0.31	0.00	0.23
z	0.05	0.00	0.00	0.08	0.00	0.01	0.00	0.04	0.00	0.00	0.00	0.00	0.14	0.00	0.01	0.19	0.00	0.36	0.13

4.6 Summary

In this chapter we verified that our proposed multiple filtered variables model could produce low-dimensional representations that capture meaningful information such as phoneme and speaker.

In a first series of experiments, we measured directly how well the model could infer posterior distributions of known underlying vowel and speaker factors that were used for generating a synthetic vowel dataset. We found that, given only a realistic high-dimensional log filterbank signal, our model was able to fully recover the generating factors. In addition, we showed that having both frame and sequence level latent variables was essential to this recovery.

The second series of experiments was done with TIMIT, a corpus of recorded speech from 630 speakers. We evaluated representations through a downstream classification task: representations were clustered with K-means into discrete labels, which were mapped to reference labels. Phoneme and speaker framewise classification accuracies were used to measure how much phoneme or speaker information was captured in the representation.

We first established some baseline results with random and raw filterbank representations, which highlighted that K-means with filterbank features is a non-trivial baseline for unsupervised phoneme classification. We also set additional baselines with plain VAE and β-VAE representations. Although some of these were somewhat better than filterbank features for capturing phoneme information, they all failed at representing speaker information.

We then showed how time filtering was acting as a bias to induce meaningful representations, by improving phoneme accuracy with short filters and speaker accuracy with long filters. However separate models were needed for phoneme and speaker representations, since a single latent variable was not able to capture both at the same time. A discrete inductive bias was tested, but could not be made to work as expected on TIMIT.

By introducing multiple latent variables in a single model, we obtained representations that were good at capturing both phoneme and speaker identity. Jointly modelling two latent variables was shown to improve over separately modelling them. We were able to improve this model further with a larger receptive field, a selective target for reconstruction, and self-attention. Our best MFL VAE model achieves a test framewise phoneme accuracy of 0.392 and speaker accuracy of 0.987, on the TIMIT development set, using self-attention with 6 heads in both encoder and decoder, with completely unsupervised training.

CHAPTER 5

DISCUSSION

Although many generative models similar to VAE have been introduced for unsupervised learning, most notably GANs and AAE, as described in Section 1.4, these models optimize good reconstruction but not interpretability or disentangling of underlying factors. Some previous works use the sequence vs. observation level distinction, such as FHVAE and the sequential autoencoder mentioned in Section 2.2. These two approaches assume one underlying factor that varies independently from frame to frame, and one that is constant across the utterance. Note that Chorowski *et al.* (2019) also has a sequence-level variable, but trains it with supervision. In contrast, we proposed here a model where each latent variable has its own filter that controls the time scale of captured information independently, without being limited to a binary opposition. Separating underlying factors in the representation is obtained by assigning a separate latent variable to each factor, yielding a global representation where each factor occupies a strict subset of the dimensions.

Time-scale biases have been succesful in the separation of phoneme and speaker factors, but the separation is still incomplete. As mentioned in Section 4.5.3.2, for the MFL-VAE model results, the short-term variable has a phoneme accuracy of 0.392, but its speaker accuracy is still 0.132, better than the MFCC baseline of 0.122. So the short-term variable does captures a little of the speaker information. A possible explanation would be a failure of the long-term encoder to fully capture speaker-dependent speech dynamics, even though it has access to an input spanning several frames. Similarly, the long-term variable has a speaker accuracy of 0.950, yet a relatively high phoneme accuracy of 0.161, so it still contains phoneme information. This may be due to speaker characteristics that are phoneme-dependent, which forces the long-term variable to encode this information. In the model, the latent variables are assumed independent from one another. This result suggests that a speaker variable should perhaps be conditioned on a phoneme variable; in other words, knowledge of phonetic content would yield a better disentangled speaker representation.

The dimension of the phoneme latent variable is small and acts as a strong bottleneck; it is another inductive bias. Notably, it seems to be relatively invariant across the databases that were explored in this work. The optimal was 5 for simulated vowels, and approximately 10 to 15 for TIMIT. This is consistent with the range generally posited to generate phonetic content in speech production models such as Stevens & Keyser (2010).

To put these results into perspective, the best "unsupervised" framewise phoneme accuracy reported for TIMIT with the same TIMIT development set and 39 phoneme classes, is 0.484 (Yeh *et al.*, 2018) with the provision that Yeh *et al.* (2018) used a language model trained from manual transcriptions (so not a completely unsupervised setting). This particular result is reported as 51.6% FER in Yeh *et al.* (2018) and was obtained for the second iteration of their proposed model without oracle phoneme boundaries, using the weaker "nonmatching" 5-gram language model derived from an held-out third of the training sentences.

Self-attention seems promising, but much more work is needed. Using attention both in the encoder and decoder, as is done in NLP transformers (Vaswani *et al.*, 2017), did yield small improvements, but that was rather disappointing. We would expect that giving sequence modelling power to the model, which has none otherwise, would result in larger improvements. Note that we did not have success with positional encoding, while it seems that positional information should play an important role in predicting coarticulation, for example. So positional encoding needs more investigation, as we did not try some of the several positional encoding schemes that have been proposed for speech (Povey *et al.*, 2018; Sperber *et al.*, 2018). Similarly, as implemented here, attention spans the whole utterance. In practice, to handle very long utterances, attention would need to be restricted to a reasonable context, and this constraint might even have some benefits, as results from Povey *et al.* (2018) seem to suggest. Also intriguing is the recent proposal by Dai, Yang, Yang & Carbonell (2019), where a recurrent state propagates information from the previous segment, a mechanism that would provide a way to handle very long utterances as sequences of segments.

An important negative result was obtained in the TIMIT experiments: multimodal priors were not helpful in inducing phoneme representations. This is surprising, given the underlying discreteness of phonemes. Furthermore, recent results from Chorowski *et al.* (2019) support the idea that a discreteness constraint is helpful in deriving a phoneme representation, although that result was obtained with supervision about the speaker variable. It must be noted, however, that even for simulated vowels, convergence to a good solution was difficult, for example it required careful tuning of relative learning rates of the latent variables and of the means spacing for the multimodal prior. So there are two plausible explanations here. One is that difficult convergence and the existence of many local minimas due to the multiple modes (see Section 2.1.5) could have led to suboptimal results for TIMIT, while for simulated data these problems were easier to overcome. The second one is that the learned representation did cluster around the prior modes, which indeed appears to be the case visually, but that these clusters *did not correspond well to phonemes*, so K-means classification results reflect this less than perfect correspondence. More work would be needed to clarify this point.

Silence removal at beginning and end of utterances (endpointing) has a big effect on overall framewise phoneme accuracy. Before endpointing, silence intervals represent 34% of all frames, and in this situation silence is easily detected (a typical MLF model would have a silence frame accuracy of 80% or more). After removal, silence frames represent a more reasonable 12% and silence frame accuracy is approximately 60%, in a range similar to the accuracy for /s/ or /f/. Unfortunately, papers that report framewise phoneme accuracy seldom mention whether endpointing was applied.

In TIMIT, each recording session is a single utterance from one speaker, and sessions from the same speaker are just randomly shuffled in the data. Thus in TIMIT, there is no actual distinction between session and speaker, so we expect the long-term latent variable to represent speaker/session pairs rather than just the speaker. However, qualitative results illustrated in Figure 4.13 show that all utterances from the same speaker do cluster together in the long-term representation. A quantitative evaluation of the speaker and session disentangling would require a database where each recording session includes several speakers taking turns.

CONCLUSION AND RECOMMENDATIONS

In previous work, unsupervised low-dimensional representations for speech have been applied to various tasks such as speaker verification and emotion detection from speech. But these representations capture simultaneously different sources of variability, and that is why i-vectors are also called "total variability" vectors. The consequence is that the same representation can be used for different applications, and i-vectors have been used for speaker verification, emotion detection, personality classification, etc. But it also means that for any particular application, a downstream classifier must be trained to extract the information of interest, and in general, this must be done with supervised training.

In the current work, we demonstrated that imposing biases such as time scales and multimodal priors can induce representations that separate different information into distinct latent variables, without requiring any supervision.

6.1 Summary of contributions

In Chapter 3, we proposed a multiple filtered latent variables VAE model, in which we introduced time filtering as a bias to induce representations at a different time scale for each latent variable. We showed how to impose a multimodal prior for discrete latent variables, and derive two tractable variants of the VAE loss function: E-M reestimation with matched KL divergence, and KL divergence sampling. We also described how self-attention layers can be used to provide for sequence modelling.

In Chapter 4, using simulated vowels, we examined how well the proposed time-scale and discreteness biases encouraged posterior distributions to recover underlying vowel and speaker factors that were used to generate audio features. We found that, given only a realistic high-dimensional log filterbank signal, the model accurately recovered the generating factors. In

addition, we showed that both frame- and sequence-level variables were essential for accurate reconstruction as well as well-disentangled representation.

With TIMIT, a corpus of recorded speech from 630 speakers, we evaluated unsupervised representations through a downstream classification task: representations were clustered with K-means into discrete labels, which were mapped to reference labels. Phoneme and speaker framewise classification accuracies measured how much phoneme or speaker information was captured in each representation. We showed that plain VAE and β-VAE representations are somewhat better than filterbank features for capturing phoneme information, but completely fail at representing speaker information. Time filtering as a inductive bias to induce a phoneme representation had its largest impact on plain VAE, bringing it to the same level of performance as β-VAE while having a negligible effect on β-VAE. In contrast, time filtering on a long-term scale was critical for inducing speaker representation. Jointly optimizing for multiple latent variables with a distinct bias for each one made it possible to disentangle underlying factors that a single latent variable cannot capture simultaneously. Finally, self-attention did provide a small improvement but not as much as could be expected from adding sequence modelling.

The proposed extensions to VAE and experimental results obtained with this model are the subject of a manuscript submitted to the IEEE/ACM Transactions on Audio, Speech and Language Processing, currently requiring minor revisions for English usage before acceptance.

6.2 Limitations and recommendations

In this work, we did not concentrate on the design of the low-level, first layers of the neural networks. For example, we did not try complex convolutional architectures, as is usually done for speech processing. Here, the focus was on exploring inductive biases to obtain interpretable representations, rather than obtaining high performance but uninterpretable representations. The results obtained here could possibly be improved by adding more complex signal preprocessing

layers; however, it is also possible that these approaches are only effective when training is supervised.

Multimodal priors did not seem to improve phoneme representations. This result suggests that more work is needed in that direction, a conclusion that is also partly supported by recent supervised results from Chorowski *et al.* (2019).

Finally, extending this work to a more challenging dataset with multiple speakers in each recording would be of special interest, as there was only one speaker per recording in TIMIT. Trying to separate speakers and recording would verify the model's capacity to include a third factor at a different time scale, with potential applications to domain adaptation and speaker diarization.

APPENDIX I

HYPERPARAMETERS

Table-A I-1 Model hyperparameters

	Typical values	Description
attention	(6, 0)	Number of attention heads for each latent variable
attention_layer	((3, 3), (-1, -1))	Hidden layers where attention is applied for each latent variable
batch_norm	True	Batch normalisation on hidden layers (true or false)
batch_norm_affine	True	Train affine transform for batch norm (true or false)
beta	(0.1, 0.1)	Beta value for each latent variable
code_dim	(12, 48)	Dimension of each latent variable
data_dim	440	Encoder input size (depends on source_slice)
data_dim_out	120	Decoder output size (dependes on target_slice)
dataset	Timit39M2spl	Timit39M2spl
dataset_batch_size	15	Minibatch size (n. of sequences in minibatch)
dataset_buckets	(71,...,350)	Max. sequence length for each dataset bucket
dataset_padded	True	Use padding in dataset
dataset_root	Experiments/data	Root directory of dataset
dataset_suffix	fbank5-nosil_train	Detailed specification for dataset
device	cuda	Device where to run model (cpu or gpu)
drop_out_p	0	Dropout probability for hidden layers
em_iters	0	Estimate matched KL with this number of iterations
front_end	none	Front-end feature processing: cnn or tdnn or none
h_dim	(200, 200)	Hidden layer size for each latent variable
init_means	hypercube	Type of initial mixture prior: random, circular, hypercube, etc. for each latent variable
kernel_len	(6, 500)	Filter length for each latent variable
kl_type	(,)	Type of KL estimation for each latent variable
kmeans_nclusters	(48, -1)	N. of k-means clusters for accuracy, for each label type (-1 uses n. of classes in reference)
lat_types	(c, s)	Names of latent variables
mnorm	True	Use batch mean normalization of input features (true or false)
model	S_VAE	Model class (S_VAE = multiple filtered latent variational autoencoder
n_epochs	101	N. of training epochs
n_labels	350	(unused)
n_layers	3	N. of hidden layers
num_components	64	N. of components in Gaussian mixture priors
optimizer	Adam	Optimizer used in training (Adam, RMSprop, etc.)
optimizer_eps	1e-08	Optimizer epsilon parameter
other_lr	0.001	Learning rate for nonencoder variables
output_dir	Experiments/out	Output directory for intermediate results
pooled	(c, s)	List of latent variables to be filtered
pooled_layer	(-1, -1)	Layers where filtering is applied, maybe several, for each latent variable
pooling	(avg, avg)	Type of pooling for each filtered variable (average or convolutional)
positional_encoding	none	Use positional encoding in attention (true of false)
prior_type	(Normal, Normal)	Prior type for each latent variable (Normal, Gaussian mixture, etc.)
qz_lr	(0.0001, 0.0001)	Learning rate for encoder variables
rep_layer	-1	Hidden layer from which representation is extracted
scheduler_gamma	1	Gamma parameter of scheduler (1. means no effect)
scheduler_milestones	(20, 40, 60, 80)	Apply scheduler gamma after each of these number of epochs
source_slice	(0, 440)	Begin, end of feature slice for encoder input
spread	(0.25, 0.25)	Spread of mean values for mixture priors
superv_indices	(-1, -1)	For supervised experiments, indices of supervision labels to use
target_slice	(160, 280)	Begin, end of feature slice for decoder output
vnorm	True	Use batch variance normalization of input features (true or false)
xavier_init	False	Use xavier initialization (true or false)

LIST OF REFERENCES

Airoldi, E. M., Blei, D. M., Erosheva, E. A. & Fienberg, S. E. (2014). Introduction to mixed membership models and methods. In *Handbook of Mixed Membership Models and Their Applications* (ch. 1, pp. 3–14). Chapman & Hall/CRC.

Aitchison, L., Adam, V. & Turaga, S. C. (2018). *Discrete flow posteriors for variational inference in discrete dynamical systems*. arXiv e-print. Consulted at http://arxiv.org/abs/1805.10958.

Al-Rfou, R., Choe, D., Constant, N., Guo, M. & Jones, L. (2019). Character-Level Language Modeling with Deeper Self-Attention. *Proc. AAAI Conference on Artificial Intelligence*, pp. 3159–3166.

Anderson, E. C. (1999). *Monte Carlo Methods and Importance Sampling*. Lecture Notes for Statistical Genetics 578C.

Astrahan, M. M. (1970). Speech Analysis by Clustering, or the Hyperphoneme Method. *Stanford Artificial Intelligence Project Memorandum*, AIM-124, 22.

Barber, D. (2016). *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. (2006). Greedy Layer-Wise Training of Deep Networks. *Proc. NIPS*, pp. 153–160.

Bengio, Y., Courville, A. & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.

Bhattacharya, G., Alam, J. & Kenny, P. (2017). Deep Speaker Embeddings for Short-Duration Speaker Verification. *Proc. Interspeech*, pp. 1517–1521.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer.

Blaauw, M. & Bonada, J. (2016). Modeling and transforming speech using variational autoencoders. *Proc. Interspeech*, pp. 1770–1774.

Blei, D. M., Kucukelbir, A. & McAuliffe, J. D. (2016). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518), 859–877.

Blei, D. M. & Lafferty, J. (2009). Topic models. In *Text mining: classification, clustering, and applications* (pp. 101–124). Chapman & Hall/CRC.

Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5, 135–146.

Bouchacourt, D. (2017). *Task-Oriented Learning of Structured Probability Distributions*. (Ph.D. thesis, University of Oxford).

Boulianne, G. (2015). Language-independent voice passphrase verification. *Proc. ICASSP*, pp. 4490–4494.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R. & Bengio, S. (2016). Generating Sentences from a Continuous Space. *CoNLL*, pp. 10–21.

Bredin, H. (2017). TristouNet: Triplet loss for speaker turn embedding. *Proc. ICASSP*, pp. 5430–5434.

Browman, C. P. & Goldstein, L. (1992). Articulatory Phonology: An Overview. *Phonetica*, 49(3-4), 155–180.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G. & Lerchner, A. (2017). Understanding disentangling in β-VAE. *Proc. NIPS*, pp. 1–11.

Cai, W., Chen, J. & Li, M. (2018). Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System. *Proc. Odyssey*, pp. 74–81.

Celebi, M. E., Kingravi, H. A. & Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1), 200–210.

Chen, K.-Y., Tsai, C.-P., Liu, D.-R., Lee, H.-Y. & Lee, L.-s. (2019). Completely Unsupervised Phoneme Recognition By A Generative Adversarial Network Harmonized With Iteratively Refined Hidden Markov Models. *Proc. Interspeech*, pp. 1–5.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. & Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *Proc. NIPS*, pp. 2172–2180.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I. & Abbeel, P. (2017). Variational Lossy Autoencoder. *Proc. ICLR*, pp. 1–17.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K. & Bengio, Y. (2015). Attention-based models for speech recognition. *Proc. NIPS*, pp. 577–585.

Chorowski, J., Weiss, R. J., Bengio, S. & van den Oord, A. (2019). *Unsupervised speech representation learning using WaveNet autoencoders*. arXiv e-print. Consulted at http://arxiv.org/abs/1901.08810.

Chou, J.-C., Yeh, C.-C., Lee, H.-Y. & Lee, L.-S. (2018). Multi-target Voice Conversion without Parallel Data by Adversarially Learning Disentangled Audio Representations. *Proc. Interspeech*, pp. 501–505.

Chowdhury, F. A. R., Wang, Q., Moreno, I. L. & Wan, L. (2018). Attention-Based Models for Text-Dependent Speaker Verification. *Proc. ICASSP*, pp. 5359–5363.

Clark, C., Lee, K., Zettlemoyer, L., Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018). Deep contextualized word representations. *Proc. NAACL-HLT*, pp. 2227–2237.

Cremer, C., Li, X. & Duvenaud, D. (2018). Inference Suboptimality in Variational Autoencoders. *Proc. ICML*, pp. 1–12.

Cummins, N., Epps, J., Sethu, V. & Krajewski, J. (2014). Variability compensation in small data: Oversampled extraction of i-vectors for the classification of depressed speech. *Proc. ICASSP*, pp. 970–974.

Cyrta, P., Trzciński, T. & Stokowiec, W. (2018). Speaker diarization using deep recurrent convolutional neural networks for speaker embeddings. *Advances in Intelligent Systems and Computing*, 655, 107–117.

Dai, A. M. & Le, Q. V. (2015). Semi-supervised Sequence Learning. *Proc. NIPS*, pp. 3079–3087.

Dai, Z., Yang, Z., Yang, Y. & Carbonell, J. (2019). *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. arXiv preprint. Consulted at https://arxiv.org/pdf/1901.02860.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T. & Tomczak, J. M. (2018). *Hyperspherical Variational Auto-Encoders*. arXiv preprint. Consulted at http://arxiv.org/abs/1804.00891.

Deena, S., Hasan, M., Doulaty, M., Saz, O. & Hain, T. (2016). Combining feature and model-based adaptation of RNNLMs for multi-genre broadcast speech recognition. *Proc. Interspeech*, pp. 2343–2347.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P. & Ouellet, P. (2011). Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4), 788–798.

Delpech, E., Laignelet, M., Pimm, C., Raynal, C., Trzos, M., Arnold, A. & Pronto, D. (2018). A Real-life , French-accented Corpus of Air Traffic Control Communications. *Language Resources and Evaluation Conference (LREC)*, pp. 1–5.

Deng, J., Xu, X., Zhang, Z., Fruhholz, S. & Schuller, B. (2018). Semisupervised Autoencoders for Speech Emotion Recognition. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26(1), 31–43.

Deng, L. & Jaitly, N. (2016). Deep discriminative and generative models for speech pattern recognition. In *Handbook of Pattern Recognition and Computer Vision* (ch. 1.2, pp. 25–72). World Scientific.

Dhillon, I., Guan, Y. & Kulis, B. (2005). A Unified View of Kernel k-means, Spectral Clustering and Graph Cuts. *Computational Complexity*, 25(5), 1–20.

Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K. & Shanahan, M. (2017). Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *Proc. ICLR*, pp. 1–12.

Dimitriadis, D. & Fousek, P. (2017). Developing on-line speaker diarization system. *Proc. Interspeech*, pp. 2739–2743.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. & Darrell, T. (2013). DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *Proc. ICML*, pp. 647–655.

Durrieu, J. L., Thiran, J. P. & Kelly, F. (2012). Lower and upper bounds for approximation of the Kullback-Leibler divergence between Gaussian mixture models. *Proc. ICASSP*, pp. 4833–4836.

Eastwood, C. & Williams, C. K. (2018). A framework for the quantitative evaluation of disentangled representations. *Proc. ICLR*, pp. 1–15.

Edwards, H. & Storkey, A. (2017). Towards a Neural Statistician. *Proc. ICLR*, pp. 1–13.

Esmaeili, B., Wu, H., Jain, S., Bozkurt, A., Siddharth, N., Paige, B., Brooks, D. H., Dy, J. & van de Meent, J.-W. (2018). *Structured Disentangled Representations*. arXiv e-print. Consulted at http://arxiv.org/abs/1804.02086.

Fitch, W. T. & Giedd, J. (1999). Morphology and development of the human vocal tract: A study using magnetic resonance imaging. *The Journal of the Acoustical Society of America*, 106(3), 1511–1522.

Foulds, J. & Frank, E. (2010). A review of multi-instance learning assumptions. *Knowledge Engineering Review*, 25(1), 1–25.

Gales, M. J. F. (1998). Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12(2), 75–98.

Garcia-Romero, D., Snyder, D., Sell, G., Povey, D. & McCree, A. (2017). Speaker Diarization Using Deep Neural Network Embeddings. *Proc. ICASSP*, pp. 4930 – 4934.

Goldberger, J., Gordon, S. & Greenspan, H. (2003). An efficient image similarity measure based on approximations of KL-divergence between two gaussian mixtures. *Proc. International Conference on Computer Vision*, pp. 487–493.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27*, 2672–2680.

Gundogdu, B. & Saraclar, M. (2017). Similarity learning based query modeling for keyword search. *Proc. Interspeech*, pp. 3617–3621.

Gupta, V., Kenny, P., Ouellet, P. & Stafylakis, T. (2014). I-vector-based speaker adaptation of deep neural networks for French broadcast audio transcription. *Proc. ICASSP*, pp. 6334–6338.

Hammersley, J. G. & Handscomb, D. C. (1964). *Monte Carlo Methods*. London, UK: Fletcher & Son Ltd, Norwich.

Hazen, T. J., Shen, W. & White, C. (2009). Query-by-example spoken term detection using phonetic posteriorgram templates. *IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 421–426.

Henter, G. E., Wang, X. & Yamagishi, J. (2018). *Deep Encoder-Decoder Models for Unsupervised Learning of Controllable Speech Synthesis*. arXiv e-print. Consulted at http://arxiv.org/abs/1807.11470.

Hershey, J. R. & Olsen, P. A. (2007). Approximating the Kullback Leibler divergence between Gaussian mixture models. *Proc. ICASSP*, pp. 317–320.

Hershey, J. R., Chen, Z., Le Roux, J. & Watanabe, S. (2016). Deep clustering: Discriminative embeddings for segmentation and separation. *Proc. ICASSP*, pp. 31–35.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S. & Lerchner, A. (2017). β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *Proc. ICLR*, pp. 1–22.

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507.

Hoffman, M. D., Blei, D. M., Wang, C. & Paisley, J. W. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1), 1303–1347.

Hsu, C. C., Hwang, H. T., Wu, Y. C., Tsao, Y. & Wang, H. M. (2017a). Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. *Proc. Interspeech*, pp. 3364–3368.

Hsu, W.-N. & Glass, J. (2018a). Extracting Domain Invariant Features by Unsupervised Learning for Robust Automatic Speech Recognition. *Proc. ICASSP*, pp. 5614–5618.

Hsu, W.-N. & Glass, J. (2018b). Scalable Factorized Hierarchical Variational Autoencoder Training. *Proc. Interspeech*, pp. 1462–1466.

Hsu, W.-N., Zhang, Y. & Glass, J. (2017b). Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data. *Proc. NIPS*, pp. 1876–1887.

Hsu, W. N., Zhang, Y. & Glass, J. (2017c). Learning latent representations for speech generation and transformation. *Proc. Interspeech*, pp. 1273–1277.

Hsu, W.-N., Zhang, Y. & Glass, J. (2017d). Unsupervised Domain Adaptation for Robust Speech Recognition via Variational Autoencoder-Based Data Augmentation. *Proc. ASRU*, pp. 16–23.

Hsu, W.-N., Tang, H. & Glass, J. (2018). Unsupervised Adaptation with Interpretable Disentangled Representations for Distant Conversational Speech Recognition. *Proc. Interspeech*, pp. 1576–1580.

Huszár, F. (2017). *Variational Inference using Implicit Distributions*. arXiv preprint. Consulted at http://arxiv.org/abs/1702.08235.

IARPA. (2012). *The IARPA Babel Program*. Web page. Consulted at https://www.iarpa.gov/index.php/research-programs/babel.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8), 651–666.

Jang, E., Gu, S. & Poole, B. (2017). Categorical Reparameterization with Gumbel-Softmax. *Proc. CLR*, pp. 1–13.

Jiang, Z., Zheng, Y., Tan, H., Tang, B. & Zhou, H. (2017). Variational deep embedding: An unsupervised generative approach to Clustering. *Proc. IJCAI*, pp. 1965–1972.

Johnson, M. J., Duvenaud, D., Wiltschko, A. B., Datta, S. R. & Adams, R. P. (2016). Composing graphical models with neural networks for structured representations and fast inference. *Proc. NIPS*, pp. 2946–2954.

Kajarekar, S. S., Malayath, N. & Hermansky, H. (1999). Analysis of sources of variability in speech. *Proc. Eurospeech*, pp. 343–346.

Karaletsos, T., Belongie, S. & Rätsch, G. (2016). Bayesian representation learning with oracle constraints. *Proc. ICLR*, pp. 1–16.

Kenny, P., Boulianne, G., Ouellet, P. & Dumouchel, P. (2007a). Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 15(4), 1435–1447.

Kenny, P., Boulianne, G., Ouellet, P. & Dumouchel, P. (2007b). Speaker and Session Variability in GMM-Based Speaker Verification. *IEEE Trans. Audio Speech and Language Processing*, 15(4), 1448–1462.

Kilcher, Y., Lucchi, A. & Hofmann, T. (2017). *Flexible Prior Distributions for Deep Generative Models*. arXiv e-print. Consulted at http://arxiv.org/abs/1710.11383.

Kim, H. & Mnih, A. (2018). *Disentangling by Factorising*. arXiv preprint. Consulted at http://arxiv.org/abs/1802.05983.

Kingma, D. P. & Welling, M. (2014). *Auto-Encoding Variational Bayes*. arXiv e-print. Consulted at http://arxiv.org/abs/1312.6114.

Kingma, D. P., Rezende, D. J., Mohamed, S. & Welling, M. (2014). Semi-Supervised Learning with Deep Generative Models. *Proc. NIPS*, pp. 3581–3589.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I. & Welling, M. (2016). Improved Variational Inference with Inverse Autoregressive Flow. *Proc. NIPS*, pp. 4743–4751.

Knill, K. M., Gales, M. J., Ragni, A. & Rath, S. P. (2014). Language independent and unsupervised acoustic models for speech recognition and keyword spotting. *Proc. Interspeech*, pp. 16–20.

Kulis, B. & Jordan, M. I. (2012). Revisiting k-means: New Algorithms via Bayesian Nonparametrics. *Proc. ICML*, pp. 291–299.

Labov, W., Ash, S. & Boberg, C. (2006). *The Atlas of North American English*. De Gruyter Mouton.

Lacoste, A., Oreshkin, B., Chung, W., Boquet, T., Rostamzadeh, N. & Krueger, D. (2018). *Uncertainty in Multitask Transfer Learning*. arXiv e-print. Consulted at http://arxiv.org/abs/1806.07528.

Latif, S., Rana, R., Qadir, J. & Epps, J. (2017). *Variational Autoencoders for Learning Latent Representations of Speech Emotion*. arXiv e-print. Consulted at http://arxiv.org/abs/1712.08708.

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Dean, J., Corrado, G. S. & Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. *Proc. ICML*, pp. 8595–8598.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

Lee, A., Zhang, Y. & Glass, J. R. (2013). Mispronunciation detection via dynamic time warping on deep belief network-based posteriorgrams. *Proc. ICASSP*, pp. 8227–8231.

Lee, K. F. & Hon, H. W. (1989). Speaker-Independent Phone Recognition Using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11), 1641–1648.

Lewellen, G. (2017). *A Greedy Approximation Algorithm for the Linear Assignment Problem*. The Antimatroid. Consulted at https://antimatroid.wordpress.com/2017/03/21/a-greedy-approximation-algorithm-for-the-linear-assignment-problem/.

Li, L., Chen, Y., Shi, Y., Tang, Z. & Wang, D. (2017). Deep Speaker Feature Learning for Text-independent Speaker Verification. *Proc. Interspeech*, pp. 1542–1546.

Li, Y. & Mandt, S. (2018). Disentangled Sequential Autoencoder. *Proc. ICML*, pp. 5656–5665.

Liberman, A. M., Harris, K. S., Howard, S. H. & Griffith, B. C. (1957). The Discrimination of Speech Sounds within and Across Phoneme Boundaries. *Journal of Experimental Psychology*, 54(5), 358–368.

Lippmann, R. (1997). Speech perception by machines and humans. *Speech Communication*, 22(1), 1–15.

Liu, D. R., Chen, K. Y., Lee, H. Y. & Lee, L. S. (2018). Completely unsupervised phoneme recognition by adversarially learning mapping relationships from audio embeddings. *Proc.*

Interspeech, pp. 3748–3752.

Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B. & Bachem, O. (2018). *Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations*. arXiv e-print. Consulted at http://arxiv.org/abs/1811.12359.

Lopez, R., Regier, J., Jordan, M. I. & Yosef, N. (2018). Information Constraints on Auto-Encoding Variational Bayes. *Proc. NeurIPS*, pp. 6114–6125.

Maaløe, L., Fraccaro, M. & Winther, O. (2017a). *Semi-Supervised Generation with Cluster-aware Generative Models*. arXiv e-print. Consulted at http://arxiv.org/abs/1704.00637.

Maaløe, L., Sønderby, C. K., Sønderby, S. K. & Winther, O. (2017b). Auxiliary Deep Generative Models. *Proc. ICLR*, pp. 1–9.

Maddison, C. J., Mnih, A. & Teh, Y. W. (2017). The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *Proc. ICLR*, pp. 1–20.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluriixuan, M., Li, Y., Bharambe, A. & van der Maaten, L. (2018). Exploring the Limits of Weakly Supervised Pretraining. *Proc. ECCV*, pp. 181–196.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. & Frey, B. (2015). *Adversarial Autoencoders*. arXiv e-print. Consulted at http://arxiv.org/abs/1511.05644.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Proc. NIPS*, pp. 3111–3119.

Muhlenbach, F., Lallich, S. & Zighed, D. A. (2004). Identifying and Handling Mislabelled Instances. *Journal of Intelligent Information Systems*, 22(1), 89–109.

Munkres, J. (1957). On the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1), 32–38.

Murray, W. & Ng, K.-M. (2010). An algorithm for nonlinear optimization problems with binary variables. *Computational Optimization and Applications*, 47(2), 257–288.

Muscariello, A., Gravier, G. & Al, E. (2011). Zero-resource audio-only spoken term detection based on a combination of template matching techniques. *Proc. Interspeech*, pp. 921–924.

Nalisnick, E. & Smyth, P. (2017a). Stick-Breaking Variational Autoencoders. *Proc. ICLR*, pp. 1–12.

Nalisnick, E. & Smyth, P. (2017b). Variational Reference Priors. *ICLR Workshop*, pp. 1–4.

Nalisnick, E., Hertel, L. & Smyth, P. (2016). Approximate Inference for Deep Latent Gaussian Mixtures. *Proc. NIPS Workshop on Bayesian Deep Learning*, pp. 1–4.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. (2017). Automatic differentiation in PyTorch. *Proc. NIPS*, pp. 1–4.

Pedregosa, F., Varoquaux, G., Thirion, B., Grisel, O. & Blondel, M. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning R*, 12, 2826–2830.

Pelleg, D. & Moore, A. (2000). X-means: Extending K-means with efficient estimation of the number of clusters. *Proc. ICML*, pp. 727–734.

Pennington, J., Socher, R. & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proc. EMNLP*, pp. 1532–1543.

Phuong, M., Welling, M., Kushman, N., Tomioka, R. & Nowozin, S. (2018). The Mutual Autoencoder: Controlling Information in Latent Code Representations. *Proc. ICLR*, pp. 1–12.

Pineau, E. & Lelarge, M. (2018). *InfoCatVAE: Representation Learning with Categorical Variational Autoencoders*. arXiv preprint. Consulted at http://arxiv.org/abs/1806.08240.

Povey, D., et al. (2011). The Kaldi speech recognition toolkit. *Proc. ASRU*.

Povey, D., Hadian, H., Ghahremani, P., Li, K. & Khudanpur, S. (2018). A time-restricted self-attention layer for ASR. *Proc. ICASSP*, pp. 5874–5878.

Rezende, D. J. & Mohamed, S. (2015). Variational Inference with Normalizing Flows. *Proc. ICML*, pp. 1–10.

Rezende, D. J., Mohamed, S. & Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *Proc. ICML*, pp. 1278–1286.

Ridgeway, K. (2016). *A Survey of Inductive Biases for Factorial Representation-Learning*. arXiv preprint. Consulted at http://arxiv.org/abs/1612.05299.

Rolfe, J. T. (2017). Discrete Variational Autoencoders. *Proc. ICLR*, pp. 1–33.

Rouvier, M., Bousquet, P.-M. & Favre, B. (2015). Speaker Diarization through Speaker Embeddings. *Proc. EUSIPCO*, pp. 2127–2131.

Rudolph, M., Ruiz, F., Athey, S. & Blei, D. (2017). Structured Embedding Models for Grouped Data. *Proc. NIPS*, pp. 250–260.

Rudolph, M. R., Ruiz, F. J. R., Mandt, S. & Blei, D. M. (2016). Exponential Family Embeddings. *Proc. NIPS*, pp. 478–486.

Scholkopf, B., Smola, A. & Muller, K. R. (1996). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5), 1299–1319.

Seifart, F., Evans, N., Hammarström, H. & Levinson, S. (2018). Language documentation twenty-five years on. *Language, Journal of the Linguistic Society of America*, 94, e324–e345.

Shulby, C. D., Ferreira, M. D., de Mello, R. F. & Aluisio, S. M. (2019). Theoretical learning guarantees applied to acoustic modeling. *Journal of the Brazilian Computer Society*, 25(1), 1–12.

Siddharth, N., Paige, B., van de Meent, J.-W., Desmaison, A., Goodman, N. D., Kohli, P., Wood, F. & Torr, P. H. S. (2017). Learning Disentangled Representations with Semi-Supervised Deep Generative Models. *Proc. NIPS*, pp. 5925–5935.

Snyder, D., Garcia-Romero, D. & Povey, D. (2016a). Time delay deep neural network-based universal background models for speaker recognition. *Proc. ASRU*, pp. 92–97.

Snyder, D., Garcia-Romero, D., Povey, D. & Khudanpur, S. (2016b). Deep Neural Network-Based Speaker Embeddings For End-To-End Speaker Verification. *Proc. IEEE Spoken Language Technology Workshop (SLT)*, pp. 165–170.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K. & Winther, O. (2016). How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks. *Proc. ICML*, pp. 1–10.

Sperber, M., Niehues, J., Neubig, G., Stüker, S. & Waibel, A. (2018). Self-attentional acoustic models. *Proc. Interspeech*, pp. 3723–3727.

Stevens, K. N. & Keyser, S. J. (2010). Quantal theory, enhancement and overlap. *Journal of Phonetics*, 38(1), 10–19.

Tan, S. & Sim, K. C. (2017). Learning utterance-level normalisation using variational autoencoders for robust automatic speech recognition. *IEEE Workshop on Spoken Language Technology (SLT)*, pp. 43–49.

Tanha, J., van Someren, M. & Afsarmanesh, H. (2017). Semi-supervised self-training for decision tree classifiers. *Int. J. Mach. Learn. & Cyber.*, 8, 355–370.

Tomczak, J. M. & Welling, M. (2018). VAE with a VampPrior. *Proc. AISTATS*, pp. 1–16.

van der Maaten, L. & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention Is All You Need. *Proc. NIPS*, pp. 5998–6008.

Villalba, J., Brümmer, N. & Dehak, N. (2017). Tied variational autoencoder backends for i-vector speaker recognition. *Proc. Interspeech*, pp. 1004–1008.

Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proc. ICML*, pp. 1096–1103.

Wang, D., Li, L., Tang, Z. & Zheng, T. F. (2017). Deep Speaker Verification: Do We Need End to End? *Proc. APSIPA ASC*, pp. 177–181.

Wang, Q., Okabe, K., Lee, K. A., Yamamoto, H. & Koshinaka, T. (2019). Attention Mechanism in Speaker Recognition: What Does it Learn in Deep Speaker Embedding? *Proc. SLT Workshop*, pp. 1052–1059.

Weiss, R. J., Chorowski, J., Jaitly, N., Wu, Y. & Chen, Z. (2017). Sequence-to-sequence models can directly translate foreign speech. *Proc. Interspeech*, pp. 2625–2629.

Welling, M. (2009). Bayesian K-Means as a "Maximization-Expectation" Algorithm. *Neural Computation*, 21(4), 1145–1172.

Yang, Z., Hu, Z., Salakhutdinov, R. & Berg-Kirkpatrick, T. (2017). Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. *Proc. ICML*, pp. 3881–3890.

Yeh, C.-K., Chen, J., Yu, C. & Yu, D. (2018). Unsupervised Speech Recognition via Segmental Empirical Output Distribution Matching. *Proc. ICLR*, pp. 1–14.

Young, K. D. & Lewis, R. J. (1997). What is confidence? Part 2: Detailed definition and determination of confidence intervals. *Annals of Emergency Medicine*, 30(3), 311–318.

Zeghidour, N., Synnaeve, G., Usunier, N., Dupoux, E. & Etudes, H. (2016). Joint Learning of Speaker and Phonetic Similarities with Siamese Networks. *Proc. Interspeech*, pp. 1295–1299.

Zeinali, H., Burget, L., Rohdin, J., Stafylakis, T. & Cernocky, J. H. (2019). How to Improve Your Speaker Embeddings Extractor in Generic Toolkits. *Proc. ICASSP*, pp. 6141–6145.

Zha, H., He, X., Ding, C. & Simon, H. (2001). Spectral Relaxation for K-means Clustering. *Proc. NIPS*, pp. 1057–1064.

Zhang, Q. & Hansen, J. H. (2018). Language/Dialect recognition based on unsupervised deep learning. *IEEE Transactions on Audio Speech and Language Processing*, 26(5), 873–882.

Zhang, Y. & Glass, J. R. (2009). Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. *Proc. ASRU*, pp. 398–403.

Zhao, S., Song, J. & Ermon, S. (2017). *InfoVAE: Information Maximizing Variational Autoencoders*. arXiv preprint. Consulted at http://arxiv.org/abs/1706.02262.

Zheng, H., Yao, J., Zhang, Y. & Tsang, I. W. (2018). *Degeneration in VAE: in the Light of Fisher Information Loss*. arXiv e-print. Consulted at http://arxiv.org/abs/1802.06677.

Zhou, Z. H. (2018). A brief introduction to weakly supervised learning. *National Science Review*, 5(1), 44–53.

Zhu, Y., Ko, T., Snyder, D., Mak, B. & Povey, D. (2018). Self-Attentive Speaker Embeddings for Text-Independent Speaker Verification. *Proc. Interspeech*, pp. 3573–3577.