SPEECH ANALYSIS USING VERY LOW-DIMENSIONAL BOTTLENECK FEATURES AND PHONE-CLASS DEPENDENT NEURAL NETWORKS

by

LINXUE BAI

A thesis submitted to the University of Birmingham for the degree of DOCTOR OF PHILOSOPHY

Department of Electronic, Electrical and Systems Engineering

School of Engineering

College of Engineering and Physical Sciences

University of Birmingham

September 2017

Abstract

The first part of this thesis focuses on very low-dimensional bottleneck features (BNFs), extracted from deep neural networks (DNNs) for speech analysis and recognition. Very low-dimensional BNFs are analysed in terms of their capability of representing speech and their suitability for modelling speech dynamics. Nine-dimensional BNFs obtained from a phone discrimination DNN are shown to give comparable phone recognition accuracy to 39-dimensional MFCCs, and an average of 34% higher phone recognition accuracy than formant-based features of the same dimensions. They also preserve the trajectory continuity well and thus hold promise for modelling speech dynamics. Visualisations and interpretations of the BNFs are presented, with phonetically motivated studies of the strategies that DNNs employ to create these features. The relationships between BNF representations resulting from different initialisations of DNNs are explored.

The second part of this thesis considers BNFs from the perspective of feature extraction. It is motivated by the observation that different types of speech sounds lend themselves to different acoustic analysis, and that the mapping from spectra-in-context to phone posterior probabilities implemented by the DNN is a continuous approximation to a discontinuous function. This suggests that it may be advantageous to replace the single DNN with a set of phone class dependent DNNs. In this case, the appropriate mathematical structure is a manifold. It is shown that this approach leads to significant improvements in frame level phone classification accuracy.

Acknowledgements

I would like to express my deepest gratitude to my wonderful supervisors, professor Martin Russell and Dr Peter Jančovič, for their professional guidance, invaluable moral support, and considerable care. Also a big thank you to Dr Philip Weber, who I regard as another supervisor to me, for every big and small help he ever offered. Thank you also to Dr Steve Houghton for his inspiring ideas.

I would like to thank my lovely friends and colleagues in the University of Birmingham, for their accompany, encouragement and help. Evangelia Fringi, Dr Hao Fu, Dr Emilie Jean-Baptiste, Mao Li, Dr Roozbeh Nabiei, Dr Maryam Najafian, Yikai Peng, Mengjian Qian, Alp Sayin, Chloe Seivwright, Dr Zhongbei Tian, Xizi Wei and Xinyu Yu, with special thanks to those from the speech group. It has been a great pleasure working with them.

Finally, thank you to all my families. A special thank you goes to my parents Mr Xianyu Bai and Mrs Ruixia Tian who give me the most selfless love and always believe in me, and my husband Yongjing who encourages and supports me all the time. Thank you all for seeing me though this.

Contents

List of Figures

List of Tables

List of Abbreviations

ASR automatic speech recognition

BNF bottleneck feature

BP back-propagation

BPC broad phone class

CD contrastive divergence

CNN convolutional neural network

CS-HMM continuous-state hidden Markov model

CTC connectionist temporal classification

DNN deep neuarl network

GMM Gaussian mixtrue model

GRBM Gaussian-binary restricted Boltzmann machine

HMM hidden Markov Model

HTK hidden Markov model toolkit

LDA linear discriminant analysis

LLE locally linear embedding

logFBEs logarithm filter-bank energies

LSTM long-short term memory

MFCC Mel-frequency cepstral coefficient

MLP multi-layer perceptron

NN neural network

PCA principal component analysis

RBM restricted Boltzmann machine

RNN recurrent neural network

SGD stochastic gradient descent

t-SNE t-distributed stochastic neighbour embedding

Chapter 1

Introduction

1.1 Research background

A conventional hidden Markov model (HMM) models the speech signal as a sequence of piece-wise constant segments. The information about dynamics of speech is typically incorporated into the feature representation by concatenating features describing the current signal frame ("static") with their time derivatives ("delta"). Over the years, there has been considerable interest in alternative models which aim to model speech dynamics more accurately (Deng, 2006). In these models, which we refer to as segmental, the states are associated with sequences of acoustic vectors, or segments, rather than with individual acoustic feature vectors as in conventional HMMs. A recent addition to these models is the continuous-state HMM (Weber et al., 2014; Champion and Houghton, 2015), which can realize segmental models of speech such as the Holmes-Mattingly-Shearme dwell-transition model (Holmes et al., 1964) as computationally viable models for speech recognition. Conventional HMM models, segmental models and continuous-state HMM are briefly reviewed in Section 2.1.

Mel-frequency cepstral coefficients (MFCCs) have been shown to perform well for speech recognition, however they are less suitable as feature representation for segmental models of speech dynamics. This is due to the fact that in representations of

speech derived through a linear transformation of short-term spectra the articulator dynamics of speech are manifested indirectly, often as movement between, rather than within, frequency bands. Representation of speech in terms of formant parameters (frequencies and amplitudes) or articulatory features, directly describes the process of speech production and preserves speech dynamics. However, formant features are notoriously difficult to estimate reliably. Moreover, they are not well defined for some speech sounds. Therefore, there is a need for compact representations of speech that can be reliably estimated for all speech sounds. Such low-dimensional representations reflect the fact that the mechanisms of speech production involve movement of a small number of speech articulators.

Over the last few years, there has been an intensive research interest on employing (deep) neural networks (NN/DNN) for speech recognition. As a NN performs a non-linear mapping, they seem a natural method to employ for our problem of representing speech. There have been a variety of ways of using NNs investigated, including NNs as a non-linear feature extractor. For instance, features can be derived from the output layer (Hermansky et al., 2000), or various intermediate hidden layers of the NNs (Grézl et al., 2007). Comparisons between different deep NN hidden and output layer features as well as their concatenations were studied in (Deng and Chen, 2014). Bottleneck features (BNF) are a particular form of NN features which are extracted from NNs with a compression layer. The bottleneck structure provides a way to reduce dimensionality. However, most BN layers are still relatively high-dimensional, for example, 40-80 neurons in (Doddipatla, 2016; Petridis and Pantic, 2016), failing to exploit the fact that speech is believed to be inherently of much lower dimensionality. It would be interesting to investigate using very small bottlenecks to extract very low-dimensional BNF and see if they can well represent speech and preserve speech dynamics. Although recent years have witnessed remarkable improvements in automatic speech recognition (ASR) with the use of (deep) neural networks (Hinton et al., 2012; Deng et al., 2013), neural networks have always been a “black box” and our understanding of how (deep) neural

networks work has been limited. There has been some progress on visualising and interpreting networks in image recognition (Karpathy et al., 2015; Zeiler and Fergus, 2014), but analyses of the structures and representations learned for speech recognition are more scarce (Nagamine et al., 2015; Tan et al., 2015). The work by (Nagamine et al., 2015) suggests that DNNs learn phonetic structures in acoustic features and treat different broad phone classes differently, whereas (Tan et al., 2015) argue the contrary that DNNs have to be stimulated to learn proper phonetic structures.

Furthermore, as one would expect, both network parameters and BNFs would be different when a different initialisation is applied, and thus it is less convincing to explore and interpret only one network and one set of BNFs. However, experiments using neural networks seem to suggest that NN initialisations do not affect the final ASR results much (Weber et al., 2016). It is interesting to ask if there are any relationships between the differently initialised NNs, and specifically in our case, relationships between BNFs extracted from NNs with different initialisations. If the BNFs can be mapped to one another, the interpretation would be much more meaningful and applicable, and the mappings may give us more insight into the neural networks learning mechanism.

Most state-of-the-art automatic speech recognition (ASR) systems use a single deep neural network (DNN) to map the acoustic space to the decision space. However, different phonetic classes employ different production mechanisms and are best described by different types of features. Hence it may be advantageous to replace this single DNN with several phone class dependent DNNs.

The non-linear function T realized by the DNN maps the "acoustic space" A (for example, the space of short sequences of vectors of log filter-bank energies) to the BNF space B (*en route* to the space of vectors of phone posterior probabilities). Although this is a single continuous mapping, in practice the DNN is trained to approximate a discontinuous function whose outputs jump between 0 and 1 across triphone state boundaries. Therefore, assuming that acoustic space A is connected (which we don't

actually know), it may be advantageous to think of T as a set of continuous functions $\{T_1, ..., T_N\}$, where each T_i is defined on a subset $A_i \subseteq A$ and $A = \bigcup_i A_i$. In this case the appropriate mathematical structure is a non-linear topological manifold. Intuitively, one might hope that the subsets A_i correspond to broad phone classes (BPCs), in which case the mappings T_i implement phone-class dependent feature extraction. The idea of phone-dependent feature extraction is well-established. For example, while vocal tract resonance frequencies provide a natural description of vowels, unvoiced consonants are better described in terms of duration and mean energies in key frequency bands (Li et al., 2010, 2012; Stevens and Blumstein, 1978; Heinz and Stevens, 1961; Raphael, 1972; Wilde, 1995; Khasanova et al., 2014). There are also a number of studies that use BPC-dependent classifiers to focus on subtle differences between phones within a BPC (Scanlon et al., 2007).

A two-level linear computational model that is motivated by these considerations is presented in (Huang et al., 2016). The first level comprises a set of discriminative linear transforms W_j^T, one for each of a set of overlapping BPCs Q_j, $j = 1, ..., N$, that are used for feature extraction. The transforms W_j^T are obtained using variants of linear discriminant analysis (LDA). An acoustic feature vector t is transformed using each W_j^T to obtain $t_j = W_j^T t$ and k-nearest neighbour methods are used to estimate $p(Q_j|t_j)$ and $p(c|Q_j, t_j)$ for each specific phone class c. These probabilities are combined in the second level to estimate the posterior probabilities $p(c|t_j)$ and hence to classify t. In acoustic feature vector phone classification experiments on TIMIT (Garofolo et al., 1993), the two-level linear classifier obtained slightly better results when BPC-specific linear transforms were learned, compared to a single transform. The authors of (Huang et al., 2016) speculate that better performance would be achieved using non-linear DNN-based transformations.

The above background leads to our four research questions, which are stated in the following section.

1.2 Main research questions and contributions

1.2.1 Research question 1

Is a bottleneck neural network capable of extracting very low-dimensional speech features that are good for speech recognition and preserve speech dynamics?

Bottleneck neural networks are built and bottleneck features (BNFs) are extracted and analysed in terms of their capability of representing speech and their suitability for modelling speech dynamics. Experimental evaluations are performed on the TIMIT speech corpus (Garofolo et al., 1993) which is later described in chapter 3.1.

It is demonstrated that the low-dimensional BNFs obtained from phone classification networks give on average 33.7% improvement in phone accuracies compared with formant-based features of the same dimensionality, and 9-dimensional BNFs have been shown to perform similarly to 39-dimensional conventional Mel frequency cepstral coefficients (MFCCs), both in a conventional HMM-based ASR system. It is also observed that the BNFs preserve better the trajectory continuity and fit better the CS-HMM modelling assumptions than formants. The BNFs provide a compact speech representation in terms of the number of model parameters, can be consistently obtained for all speech sounds and they seem to be in overall well suited to be employed for segmental models of speech dynamics.

1.2.2 Research question 2

Can we interpret the very low-dimensional BNFs and bottleneck neural network learning behaviours from the perspective of phonetics?

Bottleneck activations are visualised and analysed. A new use of back-propagation is proposed to obtain "cardinal" neural activations under a trained neural network. Z-scores are estimated and plotted to visualise neuron responses. Linear discriminant

analysis (LDA) and t-distributed stochastic neighbour embedding (t-SNE) have been applied to visualise BNFs. Compressing the bottleneck layer to as small as 2 units is also attempted to achieve dimensionality reduction. The experimental results show that the networks and BNFs can be interpreted in a phonically meaningful way, which may correspond to phone production mechanisms.

1.2.3 Research question 3

Are there any relationships between BNFs obtained from neural networks trained with different initialisations?

The relationships between these sets of BNFs are explored. It is shown that the relationship between them is approximately piecewise linear. The results of experiments in which hierarchical clustering is applied to phone-dependent linear transformations between BNF sets are presented. The combined linear transforms that emerge correspond, in general, to phonetically meaningful phone classes. In addition, it is observed that the biggest decreases in phone recognition accuracy occur when transforms corresponding to categories that differ significantly in their phonetic properties are combined. This result suggests that the type of feature extraction applied by the network to extract BNFs is different for different phone categories.

1.2.4 Research question 4

Is it better to use several phone class dependent DNNs instead of a single DNN for phone classification tasks?

A manifold-inspired approach to phone classification using bottleneck neural networks is applied by using a set of phone class dependent DNNs. Various ways of designing and training the DNNs are assessed, and the results show a small but significant improvement compared with a single DNN, when a non-linear manifold structure incorporating multiple BPC-dependent DNNs is used for phone classification. A new way to decide

whether one system is significantly better than another is proposed, with the effect of NN initialisations taken into consideration. By visualising the BNFs from local BPC-dependent DNNs and making comparisons to the BNFs from a single global DNN, we show that local BPC-dependent DNNs learn clearer structures of local broad phone classes.

1.3 Thesis outline

The outline of this thesis is listed in this section, with brief introductions to each of the chapters.

Background chapter

Chapter 2 is the background chapter, which reviews theories and algorithms that are relevant to this thesis.

- Section 2.1 reviews main concepts and techniques used in automatic speech recognition (ASR). The hidden Markov model (HMM) commonly used in ASR is specifically described, including acoustic modelling with Gaussian mixture model (GMM)-based HMM and deep neural network (DNN)-based HMM. The Continuous-State HMM (CS-HMM), being a recent speech model modelling speech dynamics, is also described in this section. The GMM-HMM and CS-HMM models are used in the experiments covered in Chapter 4 in this thesis.

- Section 2.2 reviews neural networks and their developments in ASR. In particular, multi-layer perceptrons (MLP) trained by standard back-propagation and deep belief networks (DBN) trained with restricted Boltzmann machine (RBM) pre-training are described. The MLP, DBN and their relevant training techniques are used throughout the work in this thesis.

- Section 2.3 first reviews speech production mechanisms of phones and phonetic

categories, then several types of speech representations (i.e. speech features). These are used as frameworks for studying dynamic properties and phonetic interpretation of BNFs in this thesis.

- Section 2.4 reviews techniques for dimensionality reduction and high-dimensional data visualisation. In particular, linear discriminant analysis (LDA) and t-distributed stochastic neighbour embedding (t-SNE) are described. They are used in the visualisation tasks covered in Chapter 5.

- Section 2.5 describes the mathematical formalism of manifolds. The experiments in Chapter 6 are inspired by the idea of a manifold.

Experimental chapters

- Chapter 3 describes the speech corpus used in the experiments in this thesis, namely the TIMIT corpus.

- Chapter 4 addresses research question 1. This chapter reports how the very low-dimensional BNFs are extracted using bottleneck neural networks, their ASR performance in conventional GMM-HMM systems, and whether they fit the assumption of the CS-HMM speech model. The material contained in this chapter also appears in (Bai et al., 2015).

- Chapter 5 addresses research question 2. This chapter presents the 2-dimensional visualisations and interpretations of 9-dimensional BNFs using LDA and t-SNE, and 2-dimensional BNFs extracted from 2-node bottleneck layer, and explores how the bottleneck neural networks learn for the phone discrimination task. Some of the material contained in this chapter also appears in (Weber et al., 2016).

- Chapter 6 addresses research question 3. This chapter explores relationships between two BNF sets obtained from neural networks with different training initialisations.

- Chapter 7 addresses research question 4. This chapter presents phone classification experiments using a set of broad phone class (BPC) dependent neural networks. This BPC-dependent neural network structure, inspired by the idea of a manifold, is compared to the global single network, from the perspective of phone classification accuracy and LDA visualisations of BNFs. The material contained in this chapter also appears in (Bai et al., 2017).

- Chapter 8 concludes the thesis by summarising the major contributions and suggesting possible future works.

1.4 Publications

Some of the ideas and results in this thesis have appeared in other articles, published at various stages during the period of study for this thesis. They are as listed below:

- L. Bai, P. Jančovič, M.J. Russell and P. Weber, Analysis of a low-dimensional bottleneck neural network representation of speech for modelling speech dynamics, in Interspeech, Dresden, Germany, 2015, pp. 583-587

- P. Weber, L. Bai, M. Russell, P. Jančovič, and S. M. Houghton, Interpretation of low dimensional neural network bottleneck features in terms of human perception and production, in Interspeech, San Francisco, CA, USA, 2016, pp. 3384-3388

- L. Bai, P. Jančovič, M.J. Russell, P. Weber, and S. M. Houghton, Phone Classification using a Non-Linear Manifold with Broad Phone Class Dependent DNNs, in Interspeech, Stockholm, Sweden, 2017, pp. 319-323

Chapter 2

Literature Reviews

2.1 Automatic speech recognition (ASR)

2.1.1 Review on ASR

The goal of Automatic Speech Recognition (ASR) is to transform the speech (i.e. spoken words) to text automatically using machines. It has been a goal of research for more than six decades and has always been considered as an important task towards better machine-involved communications and artificial intelligence.

A general ASR process is depicted in Figure 2.1. There are mainly three components: *signal processing/feature extraction*, *model generation* and *pattern matching/decoding*. Firstly, acoustic feature vectors X are extracted from the speech signal – these are also referred to in this thesis as speech representations. A brief review on speech representations is given in Section 2.3.2. The following model generation stage models speech statistically at the acoustic and syntactic levels based on training data, and creates acoustic models and language models (sometimes also lexical models). The decoding stage can be considered as a mapping that is determined by the models and maps observed features X to a sequence of words W. It aims to return the most

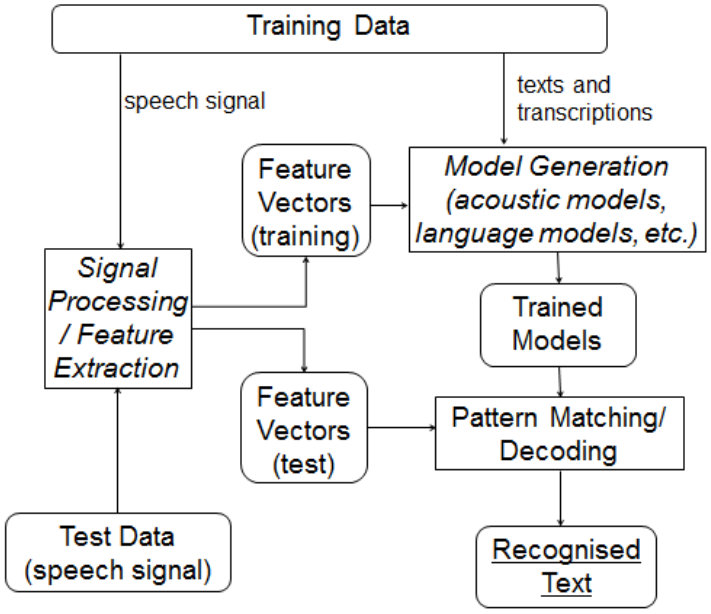


Figure 2.1: *General principles of ASR.*

probable word sequence $\hat{W}$ corresponding to X, i.e.

$$\hat{W} = \arg\max_{W} P(W|X). \tag{2.1}$$

In this thesis we use P for probabilities and p for probability densities. Using Bayes rule, we have

$$P(W|X) = \frac{p(X|W)P(W)}{p(X)}, \tag{2.2}$$

where $p(X)$, the probability of the observation vectors is a constant; $p(X|W)$ is achieved by acoustic models; $P(W)$ is determined by some linguistic criteria and modelled by language models. Therefore, a good ASR system requires $p(X|W)P(W)$ maximised, namely a good acoustic model score and a good language model score.

For decades, the hidden Markov model (HMM) has played a dominating role in ASR acoustic modelling. The HMM was first developed in the 1960s by (Baum and Eagon, 1967) and has been applied in speech recognition since the mid-1970s (Baker, Baker; Bahl and Jelinek, 1975). Gaussian mixture model (GMM)-based HMMs were the mainstream ASR approach for over four decades, before deep learning and deep neu-

ral networks (DNN)-based HMMs showed breakthrough improvements in ASR in the 2010s (Deng et al., 2013; Hinton et al., 2012) and became popular. The GMM-HMM ASR systems are also referred to as conventional ASR systems. GMM-HMMs and DNN-HMMs are to be described later in this section. In very recent research (for example Lu et al., 2015; Zhang et al., 2017; Pundak and Sainath, 2017; Hori et al., 2017), ASR systems based on convolutional neural networks (CNN, LeCun et al., 2015)[1], recurrent neural networks (RNN) with long-short term memory (LSTM, Gers et al., 2000; Hochreiter and Schmidhuber, 1997; Sak et al., 2014)[2] and connectionist temporal classification (CTC, Graves et al., 2006)[3] have shown even better performance, some of which do not include the use of HMMs[4].

Another important topic in acoustic modelling is modelling speech dynamics. Generally a dynamic model refers to a dynamic characteristic and something dynamic means it changes according to some function over time. Considerable research effort has been devoted to developing models of speech dynamics which more faithfully reflect the properties of speech structure than conventional HMMs. Such dynamic models of speech aim in various ways to reduce the assumptions that speech is a piece-wise stationary process and that the observations are temporally independent, as well as improve duration modelling. A comprehensive survey of many different types of statistical models of speech dynamics is given in (Deng, 2006). This includes segmental HMMs (for example, Ostendorf et al., 1996), trajectory models (for example, Deng and Ma, 2000; Gales and Young, 1993; Richards and Bridle, 1999), intermediate state models (for example, Henter and Kleijn, 2011), Gaussian process dynamical models (for example, Henter et al., 2012) and more recently Continuous-State HMMs (Weber et al., 2014; Champion and Houghton, 2015). In this thesis, the Continuous-State HMM (CS-HMM) is used to perform an initial analysis of the temporal dynamic of bot-

[1] A CNN is a feed-forward DNN with layers applying convolution operation to the input.

[2] A RNN is a neural network with feedback connections. LSTM is an algorithm that provides an efficient way to train RNNs. A good review of RNN and LSTM can be found in (Schmidhuber, 2017)

[3] CTC is a type of neural network output and associated scoring function that concerns label sequences without learning boundaries and timings.

[4] for example, A CTC-fitted neural network could be an alternative approach to HMM.

tleneck features. A further review on segmental models and continuous-state models is presented in Section 2.1.3.

Language models define the probability of a word sequence. An N-gram language model is a widely used language model, which defines the probability of each word given the $N-1$ preceding words. A bi-gram model is used in this thesis.

Decoding is the process that finds the most likely word sequence according to the given acoustic and language models. In HMM based ASR, it is achieved by the Viterbi algorithm. A dictionary is needed for word ASR, to express words in the vocabulary in terms of the phone-level units that are modelled. After the most likely sequence is decided, a scoring process is usually applied to evaluate recognition performance. Three types of errors are considered: deletions (D), insertions (I) and substitutions (S). The formulas to calculate recognition rates are (Young et al., 1997)

$$\%Corr = \frac{N - D - S}{N} \times 100\%, \tag{2.3}$$

and

$$\%Acc = \frac{N - D - S - I}{N} \times 100\%. \tag{2.4}$$

The word error rate (WER) is defined as

$$WER = 100\% - \%Acc. \tag{2.5}$$

2.1.2 Hidden Markov models (HMM) based ASR

A more general definition of HMM used in ASR

A hidden Markov model statistically models an assumed Markov process with unobserved (hidden) states. In an HMM it is assumed that the occurrence of a state is only dependent of the previous state and that the state transition probabilities are independent of actual time. An N state HMM model is usually defined by state probabilities,

state transition probabilities and an initial state distribution (Rabiner, 1989).

A traditional way to describe HMMs is presented in (Rabiner, 1989). We now describe the HMM framework in a way that enables to cover both the GMM-HMM and DNN-HMM systems. Let us consider an HMM λ, in which the underlying N state Markov process is supplemented by a mapping $f : X \rightarrow R^N$, where X is the set of observation vectors, such that

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_N(x) \end{bmatrix}, x \in X \tag{2.6}$$

The $f_i(x)$ denotes the probability of observing x when state i $(1 \leq i \leq N)$ is entered,

$$f_i(x) = p(x|i). \tag{2.7}$$

The decoding process aims to find the most probable sequence of hidden state given the model λ. An observation sequence $x = (x_1, x_2, ..., x_T)$ can only be generated via a path $q = (q_1, q_2, ..., q_T)$, where $q_t \in S = \{1, 2..., N\}$. The joint probability $p(x, q|\lambda)$ is then given by

$$p(x, q|\lambda) = p(x|\lambda, q)P(q|\lambda), \tag{2.8}$$

where $p(x|\lambda, q)$ is the state conditional probability and $P(q|\lambda)$ the transition probability. Equation 2.8 becomes

$$p(x, q|\lambda) = \prod_{t=1}^{T} f_{q_t}(x_t) \prod_{t=1}^{T} a_{q_t q_{t+1}}, \tag{2.9}$$

where $f_i(x)$ is the probability of observing x when state i is entered, a_{ij} is the transition probability from state i to state j.

We want to find the optimum path $\hat{q}$ such that

$$\hat{q} = \arg\max_{q} p(x, q|\lambda). \tag{2.10}$$

The above process of searching for the optimum path is achieved using a Viterbi decoder. The Viterbi algorithm is the decoding technique used in HMM based ASR systems.

Acoustic modelling with GMM-HMM

A GMM is a universal approximator of density distributions (Bilmes et al., 1998). It is a weighted sum of its component Gaussian probability density functions

$$p(x) = \sum_{m=1}^{M} w_m p_m(x), \tag{2.11}$$

where M is the number of Gaussian mixture components and $\sum_{m=1}^{M} w_m = 1(w_m > 0)$.

In the case of a standard GMM-HMM system, HMM states are represented by Gaussian mixture models

$$f_i(x) \sim g_i(x), \tag{2.12}$$

where g_i is GMM probability density function for the ith state whose parameters are optimised by the Baum-Welch algorithm which looks for a local maximum of $p(x_{tr}|\lambda)$, where x_{tr} denotes the training set.

A monophone GMM-HMM model uses a GMM to model a monophone state. For example, when modelling N monophones using 3-state HMMs, there would be $3N$ GMMs. In contrast to monophones, triphones are defined with coarticulatory effects taken into consideration. For example, the /k/ in "ticket" would be the triphone "ih-k+ax" (/k/ as it occurs when preceded by /ih/ and followed by /ax/). A triphone GMM-HMM models triphone states.

Acoustic modelling with DNN-HMM

In the case of a DNN-HMM system, HMM states correspond to the output of neural network nodes. Classification neural networks are usually used and thus the output of the network is a set of posterior probabilities. The output of network is

$$o(x) = \begin{bmatrix} o_2(x) \\ o_2(x) \\ \vdots \\ o_N(x) \end{bmatrix}, \tag{2.13}$$

where the state probability $o_i(x)$ is the output of the node i:

$$o_i(x) = P(i|x). \tag{2.14}$$

As the HMM requires $p(x|i)$, Bayes rule is applied

$$p(x|i) = \frac{P(i|x)p(x)}{P(i)} = \frac{o_i(x)p(x)}{P(i)}, \tag{2.15}$$

where $P(i)$ is the prior probability of each state estimated from the training set (calculated by $P(i) = \frac{\text{number of frames labeled as state i}}{\text{total number of frames}}$); $p(x)$ is independent of the word sequence and can be ignored. Thus we have

$$f_i(x) \sim \frac{o_i(x)}{P(i)}. \tag{2.16}$$

A single DNN is needed to estimate the conditional state posterior probabilities for all states. It is different from the GMMs discussed before where a different GMM is needed for each different state.

In a standard DNN-HMM ASR system (Yu and Deng, 2014), we need to train a GMM-HMM system first to prepare the DNN training labels. DNNs are usually applied to a triphone system after decision tree tying of the states. The resulting tied

states are called senones, and every HMM state is identified with a senone. The output targets for the DNN correspond to posterior probability distributions over the senones.

A more detailed review of DNN and its training techniques are covered in Section 2.2.1 and 2.2.3.

2.1.3 Continuous state HMMs

Conventional HMMs model speech as a sequence of constant segments, associating states with individual acoustic feature vectors which are assumed independent. Segmental models (Ostendorf et al., 1996) try to overcome the independence assumption with a segment by assuming that a segment is a noisy realisation of an underlying trajectory. However, segmental HMMs cannot enforce continuity across segment boundaries. A continuous state HMM, as proposed by (Champion and Houghton, 2015), is able to enforce continuity by including the (continuous) trajectory value in its state and using a more complex decoder. Such models aim to be more faithful to the dynamics of the speech signal arising from slow, continuous movement of a few human articulators between target positions for the various speech sounds.

The CS-HMM algorithm considers the speech to fit a dwell-transition timing model, where dwells represent phone targets and transitions the smooth migration from one dwell to the next. d-dimensional input features are assumed to be noisily distributed around underlying dwell "realisations" for a phone instance, or around transitions. Dwell realisations are also assumed to be noisily distributed around reference targets for each phone.

A sequential branching algorithm is used to recover a sequence of alternating dwells and transitions, the times of changes between them, and the sequence of phones which could have generated them. We assume dwell start and initialise one hypothesis per phone, and then update hypothesis to take account of observations.

At time t, the hypothesis can branch, either it remains in a dwell/transition or a new dwell/transition begins. A hypothesis contains the probability information of a

set of states, where discrete components of the states store the current phone identity, time since previous phone, and phone history. In a parametric form,

$$\alpha_t(\vec{x}) = K_t \, n\Big(\vec{x} - \vec{\mu}_t, \vec{P}_t\Big) \,, \tag{2.17}$$

where K_t is the sum of probabilities of paths consistent with the hypothesis, $\vec{x}$ is the realised dwell/transition target, n is a Gaussian distribution with parameters $\vec{\mu}_t$, and $\vec{P}_t$ being mean and precision of distribution over state.

Suppose in a dwell state at time $t-1$, observation $\vec{y}_t$ is made. Assuming Gaussian measurement error, $\vec{y}_t$ is drawn from the distribution $n(\vec{x}, E)$ where E is the measurement precision. We update hypotheses to account for $\vec{y}_t$, by

$$\begin{aligned} \alpha_t(\vec{x}) &= K_{t-1} \, n\Big(\vec{x} - \vec{\mu}_{t-1}, \vec{P}_{t-1}\Big) \, n(\vec{y}_t - \vec{x}, E) \\ &= K_t \, n(\vec{x} - \vec{\mu}_t, P_t) \,, \end{aligned} \tag{2.18}$$

where

$$P_t = P_{t-1} + E, \tag{2.19}$$

$$\vec{\mu}_t = P_t^{-1} \left(P_{t-1}\vec{\mu}_{t-1} + E\vec{y}_t\right), \tag{2.20}$$

$$K_t = K_{t-1} \, n\Big(\vec{y}_t - \vec{\mu}_{t-1}, \left(P_{t-1}^{-1} + E^{-1}\right)^{-1}\Big) \,. \tag{2.21}$$

Discrete components of the state store the current phone identity, time since previous phone, and phone history. Following each observation all hypotheses are split, to model the alternatives of continuing the current dwell or transition, or changing from dwell to transition or vice versa. For every new dwell, a hypothesis is created for every phone in the inventory, while for a new transition a d-dimensional vector of slope values is appended to $\vec{x}$, and marginalised out at the end of the transition. Low probability hypotheses are "pruned" to maintain computational efficiency.

2.2 Deep neural networks

2.2.1 Neural networks review

An artificial neural network (or neural network, NN for short) is an information processing paradigm that is inspired by the way biological nervous systems (such as a brain) process information. It was first proposed as a mathematical model by (McCulloch and Pitts, 1943).

One early NN is Rosenblatt's perceptron (Rosenblatt, 1962) where the input units are connected directly to the output units (single-layer perceptron, SLP). SLP is the simplest form of neural network, and is limited to classifying linearly separable data (Bishop et al., 2006).

A multi-Layer Perceptron (MLP) is a feed-forward neural network with hidden layers between the input layer and the output layer. With the addition of the hidden layers, MLPs can learn arbitrarily complex decision boundaries between different classes. MLPs were popularised by (Rumelhart et al., 1985), who introduced the use of error back-propagation to adjust NN parameters. MLP and error back-propagation are described in Section 2.2.2.

With more applications of neural networks in different fields, researchers found that back-propagation has some limitations that make NN training difficult. For example, it takes a long time to train and there tends to be a vanishing gradient problem when the network has many hidden layers. The next development milestone of neural networks is when Hinton made a breakthrough with his deep learning techniques (Hinton, 2007). The deep learning approach provides a better way to initialise deep neural networks (DNN) to provide a better starting point for error back-propagation. This is achieved by treating each adjacent pair of DNN layers as a restricted Boltzmann machine (RBM) that is trained using contrastive divergence. More details about DNNs are provided in Section 2.2.3.

Neural networks have been applied in speech recognition and made progress since

the early 1990s (Bourlard and Morgan, 1993). The recent decade has witnessed successes of DNNs in the context of speech recognition, where ASR performance has been largely improved (Hinton et al., 2012; Deng et al., 2013). Meanwhile different types of DNNs (for example, RNNs, LSTM-RNNs, CTCs) are developed and used in speech recognition. It is also noticeable that apart from the introduction of machine learning algorithms, computing capability also plays a significant role in the history of artificial neural networks.

2.2.2 Multi-layer perceptron (MLP) and error back-propagation

General structures of MLPs and the forward propagation process

The multi-layer perceptron (MLP) is one of the most important structures used in this thesis. As defined in (Bishop, 1995), multi-layered feed-forward networks having either threshold or sigmoid-like activation functions[5] are generally called MLPs. They are feed-forward neural network that map a set of inputs onto a set of outputs. The structure of an MLP with two hidden layers is shown in Figure 2.2. Layers between the input layer and the output layer are called hidden layers and units at hidden layers are called hidden units. The n^{th} layer is fully connected to the $(n+1)^{th}$ layer.

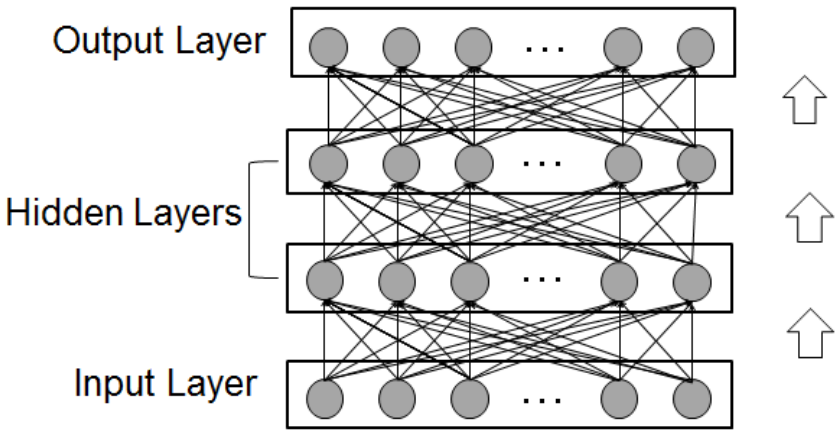


Figure 2.2: *The structure of an MLP with two hidden layers.*

[5]For example, sigmoid, tanh.

Assume a l-layer MLP with $l-2$ hidden layers between the input and output layers. Let L_m denote the layer m after the input ($m \geq 0$), such that L_0 is the input layer, L_{l-1} is the output layer. Let W_m and b_m be the weight matrix and bias vector respectively, between layer L_m and L_{m+1}. Let a_m denote the neuron activations of layer L_m, and a_0 is the neural network input.

For a hidden layer $L_m(0 < m < l-1)$, the linear output of this layer o_m is given by

$$\mathrm{o}_m = \mathrm{a}_{m-1} W_{m-1} + b_{m-1}, \tag{2.22}$$

to which a logistic sigmoid function is applied element-wise:

$$\mathrm{a}_m = \frac{1}{1 + e^{-\mathrm{o}_m}}. \tag{2.23}$$

For the output layer L_{l-1}, we also calculate the linear output first:

$$\mathrm{o}_{l-1} = \mathrm{a}_{l-2} W_{l-2} + b_{l-2}. \tag{2.24}$$

The activation function varies according to actual cases. In particular, for classification networks in which the NN outputs are posterior probabilities, the softmax function is usually used after the linear regression:

$$\mathrm{a}_{l-1} = Softmax(\mathrm{o}_{l-1}), \tag{2.25}$$

where the softmax function is defined as

$$Softmax(z) : \sigma(z)_j \equiv \frac{exp(z_j)}{\sum_{k=1}^{K} exp(z_k)} \qquad for\ j = 1, ..., K \tag{2.26}$$

The softmax function makes the maximum output more distinguishing and ensures that the outputs $\sigma(z)_j$ are within the range (0, 1) and that the summation of all outputs are always be equal to 1.0. This leads to a form of discriminative training, as probability of

one node can only be big if the other probabilities are small. In addition, the softmax function is differentiable to train by gradient descent. These characteristics make it suitable to use for classification or discrimination neural networks.

Error back-propagation (BP)

Training an MLP is a supervised learning process which needs both input samples and their corresponding required output. It is based on an error function whose value is minimised with respect to the weights and biases (Bishop, 1995). As long as the activation functions are differentiable, evaluating the derivatives of the error with respect to the weights and biases is possible. These derivatives can then be used to find the optimal weights and biases that locally minimise the error function. It should be noted that this optimum is a local optimum, and at least for now there is no way for us to find the global optimum.

The algorithm for evaluating the derivatives of the error function is known as back-propagation (BP). It was popularised in the mid-1980s by (Rumelhart et al., 1985). The term back-propagation could be used to stand for various of things, for example an MLP structure, or the training of MLP using gradient descent applied to a sum-of-squares error function (Rumelhart et al., 1985). However, in this thesis we use the term back-propagation to stand for the BP algorithm or the process of BP, regardless of what error function or optimisation algorithms are used.

The calculation of derivatives of error with respect to weights and biases can be done using the chains rule (Apostol, 1974). Now we continue with the MLP structure and denotations in the "forward propagation process" described earlier in this section. Taking the case of classification as an example, we use the softmax function at the output layer, and cross-entropy as the cost function. The cross-entropy error criterion to minimise is:

$$C = -\sum (t \ln(\mathrm{a}_{l-1}) + (\mathbf{1} - t)\ln(\mathbf{1} - \mathrm{a}_{l-1})), \tag{2.27}$$

summing over the network outputs. t is the one-hot target vector in which the target

phone has probability 1. **1** is a vector of ones of the same dimension.

Using equations 2.22 to 2.27 and the chains rule, derivatives of the error with respect to weights and biases $w_m = [W_m; b_m]$can be calculated as:

$$\frac{\partial C}{\partial w_{l-2}} = \frac{\partial C}{\partial o_{l-2}} \frac{\partial o_{l-2}}{\partial w_{l-2}} = (\mathrm{a}_{l-1} - t)\mathrm{a}_{l-2}, \tag{2.28}$$

and

$$\begin{aligned} \frac{\partial C}{\partial w_{l-2-i}} &= \frac{\partial C}{\partial w_{l-2}} \frac{\partial w_{l-2}}{\partial w_{l-3}} \cdots \frac{\partial w_{l-1-i}}{\partial w_{l-2-i}} \\ &= (\mathrm{a}_{l-1} - t) w_{l-2} [\mathrm{a}_{l-3}(\mathbf{1} - \mathrm{a}_{l-2})\mathrm{a}_{l-2}] \ldots [\mathrm{a}_{l-2-i}(\mathbf{1} - \mathrm{a}_{l-1-i})\mathrm{a}_{l-1-i}] \\ &(0 < i \leqslant l-2), \end{aligned} \tag{2.29}$$

Gradient descent optimisation algorithms

Regarding how derivatives are used to find the cost function local minimum, there are several back-propagation algorithms such as gradient descent, conjugate gradient, quasi-Newton algorithms etc. (Bishop, 1995). Mini-batch stochastic gradient descent (SGD) is by far the most popular and common way to optimise neural networks.

Figure 2.3 shows a simple 1-dimensional case of searching for local minimum of error $E(w)$ using the gradient descent algorithm.

We usually start with some initial guess for the weight vector (could be chosen at random), and iteratively update it with

$$w^{(t+1)} = w^{(t)} + \Delta w^{(t)}, \tag{2.30}$$

where w denotes the parameters to be adjusted (i.e. weights and biases), the (t) index indicates the values at training iteration t, and

$$\Delta w^{(t)} = -\eta \nabla E(w^{(t)}) + \nu \Delta w^{(t-1)}. \tag{2.31}$$

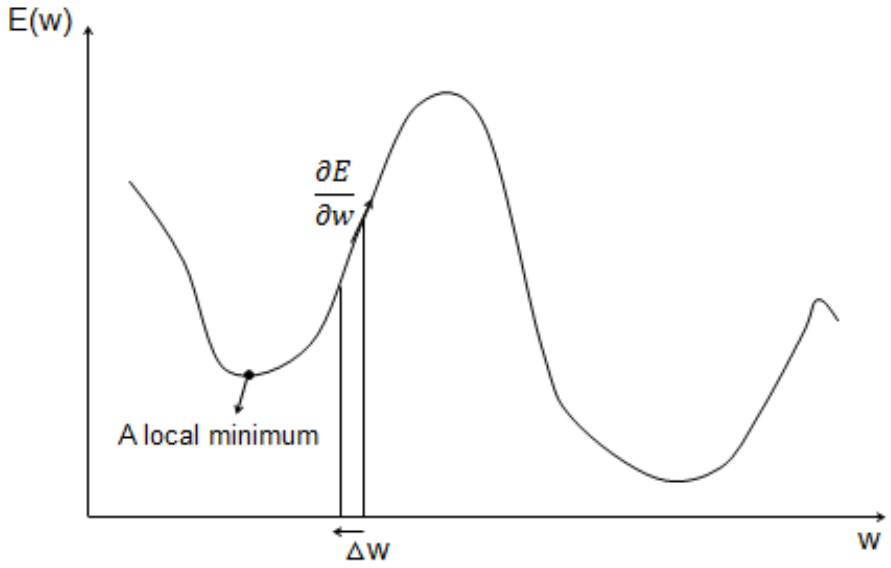


Figure 2.3: *A simple gradient descent example in 1-D.*

∇E denotes the gradient of E in weight space, η denotes the learning rate ($\eta > 0$) and ν denotes the momentum. The term $-\eta \nabla E(w^{(t)})$ indicates a small step in the direction of the negative gradient, in which the learning rate η determines how big the step is. Parameters are updated iteratively till it reaches a local optimum. Gradient descent can also be used without momentums or with adaptive learning rates.

Ordinary gradient descent performs each update based on the entire dataset, and stochastic gradient descent (SGD) performs one parameter update for each training example. The former one works slowly and is not suitable for big training set, whereas the latter one updates too frequently that it would result in high variance and therefore a heavy fluctuation in the objective function (Ruder, 2016).

Mini-batch SGD performs an update for every mini-batch of n training examples, and it takes the advantages of both the conventional gradient descent and the SGD mentioned above (Ruder, 2016). Therefore it is the most efficient among the three. Equation 2.31 is modified to:

$$\Delta w^{(t)} = -\eta \nabla E(w^{(t)}, x^{[i \times batch\_size:(i+1) \times batch\_size]}) + \nu \Delta w^{(t-1)}. \tag{2.32}$$

The experiments in this thesis use mini-batch SGD for training neural networks.

2.2.3 Deep learning with RBM pre-training

MLP, DNN and DBN

Neural networks with at least two hidden layers can be called deep neural networks (DNN), and it is a rather general concept. MLPs with multiple hidden layers is arguably just one type of DNN. The more layers a neural network has, the "deeper" it is. Sometimes DNN specifically indicates neural networks trained by layer-wise pre-training techniques, which is usually called deep learning and will be touched later in this section. As mentioned in Section 2.2.1, deep learning does a better job than conventional error back-propagation because the pre-training would end up at a better initialisation and therefore result in a better "local optimum". Moreover, the layer-wise training overcomes lots of problems back-propagations tend to have when networks are deep, and the generative unsupervised learning reduces the requirements of labels.

The deep belief network (DBN) is a classic deep learning structure and is another important structure used in this thesis. A DBN is a stack of RBMs (Hinton et al., 2006). It is usually used for DNN pre-training, which results in a better weight initialisation for further supervised training. After the pre-training, an output layer is added on top of the DBN, followed by the fine-tuning process with labels and back-propagation algorithm to finish the training of this DNN. A DNN trained with DBN is called a DBN-DNN, or simply a DBN. More descriptions on these deep learning techniques are covered later in this Section 2.2.3

Restricted Boltzmann machine (RBM)

The restricted Boltzmann machines (RBM) was proposed by (Ackley et al., 1985). It is a variant of the Boltzmann machine with the restriction that there should be no visible-visible or hidden-hidden connections. They are named after the Boltzmann distribution (also called Gibbs distribution) in statistical mechanics which is their theoretical basis.

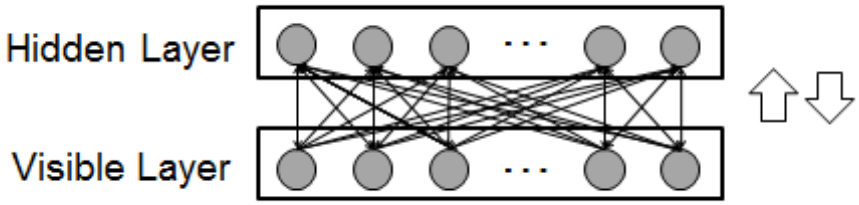


Figure 2.4: *The structure of an RBM.*

An RBM can be regarded as a two-layer neural network, with one visible layer (input layer) and one hidden layer. A simple RBM is shown in Figure 2.4. Each possible joint configuration of a visible unit v and a hidden unit h has a hop-field energy $E(v, h)$, which determines the probability that the network will choose the configuration:

$$P(v, h) = \frac{e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}, \tag{2.33}$$

then

$$P(v) = \sum_{h} P(v, h). \tag{2.34}$$

From equations 2.33 and 2.34, we have

$$P(h|v) = \frac{e^{-E(v,h)}}{\sum_{h} e^{-E(v,h)}}. \tag{2.35}$$

The hidden neurons usually take binary values and follow Bernoulli distributions, whereas the visible neurons take binary or real values, resulting in a *Bernoulli-Bernoulli RBM* or a *Gaussian-Bernoulli RBM*. In the case of a Bernoulli-Bernoulli RBM,

$$E(v, h) = -a^T v - b^T h - h^T W v, \tag{2.36}$$

where a and b are the visible and hidden layer bias vectors, and W is a weight matrix

that connects visible and hidden layer. In the case of a Gaussian-Bernoulli RBM,

$$E(v,h) = \frac{1}{2}(v-a)^T(v-a) - b^T h - h^T W v. \tag{2.37}$$

As hidden layer nodes take binary values, it can be easily derived from Equation 2.35 that

$$P(h=1|v) = \frac{1}{1+e^{-(Wv+b)}}. \tag{2.38}$$

Note that Equation 2.38 holds for both Bernoulli-Bernoulli and Gaussian-Bernoulli RBMs, and also has the same form as the logistic sigmoid function (Equation 2.23). Therefore, weights of RBMs can be used to initialise a feed-forward neural network with sigmoid hidden units.

For the binary visible neuron case, a completely symmetric derivation can be made and we have

$$P(v=1|h) = \frac{1}{1+e^{-(W^T h+a)}}. \tag{2.39}$$

For the Gaussian visible neurons, $P(v|h)$ can be estimated as

$$P(v|h) = n(v; W^T h + a, I), \tag{2.40}$$

where n stands for a Gaussian distribution and I is the appropriate identity covariance matrix, in other words, random variable v follows a Gaussian distribution with mean vector $W^T h + a$ and covariance matrix I.

RBM training and Contrastive Divergence (CD)

To train an RBM is to learn parameters W, a, b that most likely to generate what has been observed (reconstruction). In other words, run the Gibbs sampler to have a Gibbs distribution as close to the input visible distribution as possible. A picture of alternating Gibbs sampling is shown in Figure 2.5.

A straightforward approach to RBM training is to do a Maximum Likelihood learn-

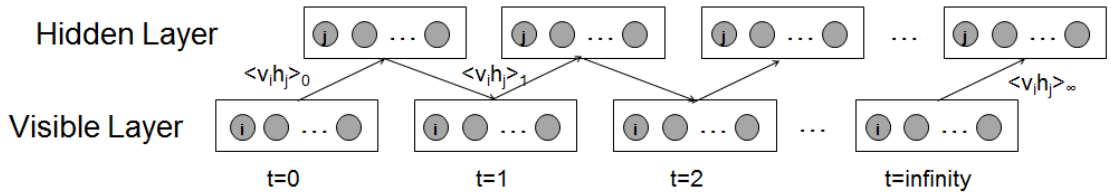


Figure 2.5: *Alternating Gibbs sampling which can be used to learn the weight of an RBM. (Hinton, 2012; Yu and Deng, 2014)*

ing: Measure the frequency of v_i and h_j being on together, which we annotate as $\langle v_i h_j \rangle$ and use the expectations from the data $\langle v_i h_j \rangle_0$ and that from the model $\langle v_i h_j \rangle_\infty$ to adjust weighs so that the distribution of the model gets close to that of the data.

SGD can be performed to minimise the negative log likelihood, i.e. $-logP(v)$.

$$\frac{\partial[-logP(v)]}{\partial w_{ij}} = -[\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_\infty]. \tag{2.41}$$

However, $\langle v_i h_j \rangle_\infty$ takes too long time to compute. A much more efficient approach is widely used, which is Contrastive Divergence (CD) (Hinton, 2006). The approximation made by the one-step CD algorithm is that

$$\begin{aligned} \frac{\partial[-logP(v)]}{\partial w_{ij}} &= -[\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_\infty] \\ &\approx -[\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_1]. \end{aligned} \tag{2.42}$$

Note that there could be more than one step but one step is usually enough, and that it is not a maximum likelihood learning but it works well (Hinton, 2006).

DBN-DNN and fine-tuning

A Deep Belief Network (DBN) consists of a stack of RBMs, where the hidden layer of each RBM performs as the visible layer in the next RBM (Hinton et al., 2006). That makes the number of DBN hidden layers equal to the number of RBMs.

We call this layer-wise RBM training of DBN *pre-training* and this is only for finding a better network initialisation. An output layer is added on top of the DBN

with the weights between them initialised to zeros. The final supervised training is called *fine-tuning*. Fine-tuning is done by standard error back-propagation.

The idea of fine-tuning may also be used to adapt a trained network to fit a particular training set. That is to say, after training on dataset A, by resetting the output layer and applying a few more epochs of error back-propagation using dataset B as the training data, the DNN may be fine tuned to work better on data similar to B.

2.3 Speech production and Speech representations

2.3.1 Speech production and English speech sound categories

The main organs of the human body responsible for producing speech are the lungs, larynx, pharynx, nose and various parts of the mouth (Holmes and Holmes, 2001), which are illustrated in Figure 2.6. Humans produce speech by coordinating these organs. However, these organs are not specialised for speech production - they play also significant roles in vital human functions such as the respiratory system (breathing) or the digestive system. From the perspective of human revolution, it is more likely that our ancestors found ways to make good use of these organs to achieve communications through speech, where articulations are limited by the organs.

These organs form an intricately shaped "tube" extending from the lungs to the lips. *Vocal tract* is the part of the tube that lies above the larynx and consists of the pharynx, mouth and nose (Denes and Pinson, 1993). The shape of the vocal tract can be varied extensively by moving the soft palate, tongue, lips and jaw, which are collectively called the *articulators*, and the process of adjusting the shape of vocal tract to produce different speech sound is called *articulation* (Denes and Pinson, 1993). Kenneth Stevens proposed using bundles of binary distinctive features to specify phonetic segments (Stevens, 2002), where each bundle of features indicates the positions of articulators of phonetic segments. An example of such features is shown Figure 2.7[6].

[6]This example copies the "TABLE V" from (Stevens, 2002)

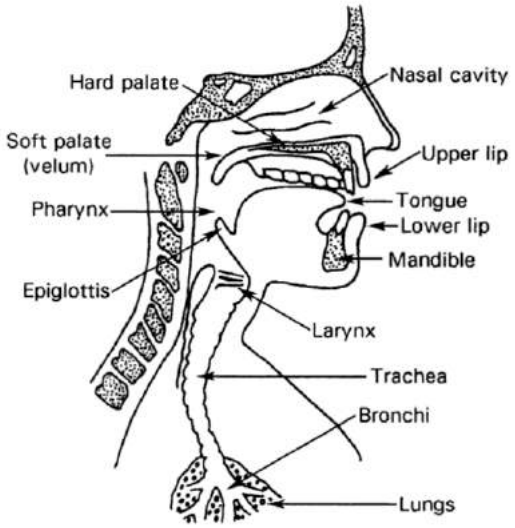
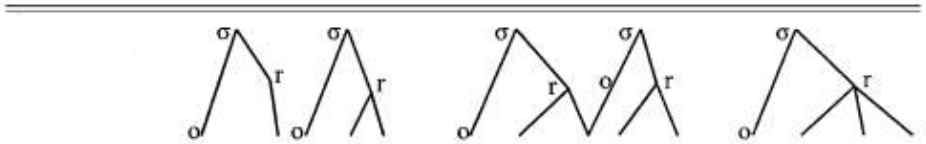


Figure 2.6: *Diagrammatic cross-section of the human head showing the vocal organs (Holmes and Holmes, 2001).*

A *phone* is defined to represent such phonetic segments that have a certain mode of production (articulation) despite of the difference between speakers, whereas a *phoneme* is defined as the smallest unit in language where substitution of one unit for another might make a distinction of words (Holmes and Holmes, 2001). A phoneme can correspond to many phones due to different individuals' speaking habits, for example accents.

English phones can be categorised by the type of excitation and their manners and places of articulation. Vowels—Consonants is a two-class phone categorisation. As defined in (Johns, 1975), "a vowel in normal speech is a voiced sound in forming which the air issues in a continuous stream through the pharynx and mouth, there being no obstruction and no narrowing such as would cause audible friction"; all other sounds are called consonants. Vowels and consonants are to be discussed later in this section. In this thesis we use the phone symbols according to the Arpabet symbol set developed for speech recognition uses (Shoup, 1980) and write phone symbols between oblique lines, for example, /aa/.

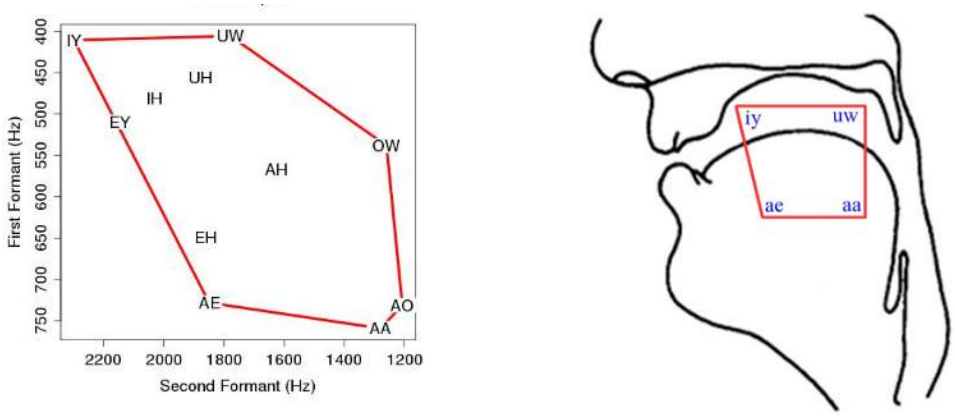


	d	ə	b	e	t		w	æ	g	ə	n		h	ɛ	l	p
Vowel		+		+				+		+				+		
Glide							+						+			
Consonant	+		+		+				+		+				+	+
Stressed		-		+				+		-				+		
Reducible		+		-				-		+				-		
Continuant	-		-		-				-		-				-	-
Sonorant	-		-		-				-		+				+	-
Strident																
Lips			+													+
Tongue blade	+				+						+				+	
Tongue body									+							
Round			-				+									-
Anterior	+				+						+				+	
Lateral															+	
High		+		-			+	-	+	-				-		
Low		-		-			-	+	-	-				-		
Back		-		-			+	-	-	+				-		
Adv. tongue root				+			+	-						-		
Spread glottis													+			
Nasal											+					
Stiff vocal folds	-		-		+				-							+

Figure 2.7: *Lexical representations for the words debate, wagon and help. The syllable structure of each word is schematized at the top (σ=syllable, o=onset, r=rime) (Stevens, 2002).*

Vowels

Vowels are made without any constriction in the vocal tract. The shape of vocal tract enhances some harmonics of fundamental, while suppressing others. The regions of enhanced harmonics are called formants and will be described in Section 2.3.2. Formants frequencies are related to position of tongue and lips. According to the tongue positions when producing vowels, vowels are called high or low, front or back, and central (also called neutral) vowels. For example, /iy/ is a high front vowel, and /aa/ is a low back vowel. A vowel space diagram is shown in Figure 2.8(a).



(a) (b)

Figure 2.8: *(a) A vowel space diagram, aggregated over many speakers (University of California Berkeley, 2011). (b) a text book image of some typical acoustic measurements for the vowels in "she", "who", "odd", and "rack" (symbolized here with the symbols "iy", "uw", "aa" and "ae" respectively), overlaid on an x-ray tracing of a vocal track (University of California Berkeley, 2011)*

It should be noticed that the absolute positions of phones could be different on different vowel space diagrams due to the variance in human vocal tracts. However, the general relative positions stay the same pattern.

Consonants

The consonants of English are best described by specifying their place of articulation and manner of articulation. The categories of manner-of-articulation are plosive (also called stop), fricative, affricate, nasal, and approximant (Denes and Pinson, 1993).

The *plosives* are made by blocking the air pressure and then suddenly releasing it (Denes and Pinson, 1993). Plosives can be voiced (for example, /b/, /d/, /g/) or unvoiced (for example, /p/, /t/, /k/). When labelling speech, there are also transcription such as TIMIT labelling (Garofolo et al., 1993, to be described in chapter 3.1) that separate the "blocking stage" and the "releasing stage", resulting in closures (in TIMIT labels: /bcl/, /dcl/, /gcl/, /p/, /t/, /k/) and releases (/b/, /d/, /g/, /p/, /t/, /k/).

The *fricatives* are articulated by constricting the air flow which causes turbulence and thereby produce a sound of hissy quality (Denes and Pinson, 1993). There are also

both voiced (/z/, /zh/) and unvoiced fricatives (/s/, /f/, /th/).

The *affricates* are /ch/ and /jh/. They are made by a brief stop followed by a fricative (Denes and Pinson, 1993).

The *nasals* are produced by lowering the soft palate thereby coupling the nasal cavities to the pharynx and allowing air flow through the nose (Denes and Pinson, 1993). Examples are /m/, /n/, /ng/, /en/.

The *approximants* are the consonants /w/, /y/, /r/, /l/. They are voiced consonants and are also regarded as "semi-vowels" as they have some vowel-like structure.

Phone categorisation used in this thesis

In this thesis, we use the categorisation by (Halberstadt and Glass, 1997). Table 2.1 shows how they define phone categories and what phones are included in each category.

Table 2.1: *Phone categorisation used in this thesis (Halberstadt and Glass, 1997).*

Phone category	Phone label
Plosive	/g/, /d/, /b/, /k/, /t/, /p/
Strong fricative	/s/, /z/, /sh/, /zh/, /ch/, /jh/
Weak fricative	/f/, /v/, /th/, /dh/, /hh/
Nasal/Flap	/m/, /n/, /en/, /ng/, /dx/
Semi-vowel	/l/, /el/, /r/, /w/, /y/
Short vowel	/ih/, /ix/, /ae/, /ah/, /ax/, /eh/, /uh/, /aa/
Long vowel	/iy/, /uw/, /ao/, /er/, /ey/, /ay/, /oy/, /aw/, /ow/
Silence	/sil/, /epi/, /q/, /vcl/, /cl/

2.3.2 Speech representations (speech features)

Speech representations for ASR are a sequence of feature vectors that are to be used to represent speech. The phonetic features and formant frequencies that have been mentioned in the previous section are both speech representations. However, there are various of speech features and their combinations, and there are even research using raw waveforms as neural network input (for example, Graves et al., 2012; Graves and Jaitly, 2014).

Filter bank energies (usually with a Mel scale) and Mel-scaled frequency cepstral coefficients (MFCCs) are two classic spectral features. Mel frequencies are used to take human perception sensitivity with respect to frequencies into consideration. In conventional GMM-HMM ASR systems, MFCCs of 39 dimensions are used as acoustic features. In recent DNN-HMM hybrid ASR systems, spectral features in all kinds of forms (for example with delta features, with contextual frames, with dimensionality reduction and with speaker normalisation) are used as the input to neural networks. In this thesis, we use filter bank energies as the input features to neural networks. A more detailed description of the features we use can be found in Section 4.2.

Formant frequencies are classic non-spectral features. As mentioned in Section 2.3.1, formants correspond to the resonances of the vocal tract. They are numbered from the low frequency end, and usually the first three or four formant frequencies are adequate for satisfactory perception (Holmes et al., 1997; Sjölander and Beskow, 2000). While formant frequencies provide a natural description of vowels, unvoiced consonants[7] are better described in terms of duration and mean energies in key frequency bands.

Neural networks are also used as feature extractors. Output of a hidden layer can be treated as a high-level feature and used in further task (for example, Deng and Chen, 2014). Particularly, a compression bottleneck layer can be used to obtain features of lower dimensions (Grézl et al., 2007; Jiang et al., 2014). This neural network architecture is known as bottleneck neural network and the output of the bottleneck layer is called bottleneck features (BNFs). In recent research, most BN-NNs have tens to hundreds of neurons at the bottleneck layer, such as (Grézl et al., 2007; Liu et al., 2014; Doddipatla, 2016; Petridis and Pantic, 2016).

[7]For example, stop consonants are studied in (Li et al., 2010; Stevens and Blumstein, 1978) and fricative consonants are studied in (Li et al., 2012; Heinz and Stevens, 1961; Wilde, 1995).

2.4 Feature visualisation and dimensionality reduction

2.4.1 Dimensionality reduction

Dimensionality reduction means to use a d-dimensional vector y_i to represent a D-dimensional vector x_i, where $d \ll D$. Generally, there are two ways to achieve dimensionality reduction: one is to select a subset of the original variables, and the other one is to apply a transformation from the higher dimensional space to a lower dimensional space. We only consider the latter in this thesis. The main purposes of using dimensionality reduction include: to reduce the amount of variables for less storage requirement and faster computing, to throw away redundant or distracting data, and to visualise high-dimensional data. Dimensionality reduction techniques are important in data analysis and data classification.

Popular dimensionality reduction techniques include principal component analysis (PCA), linear discriminant analysis (LDA), Isomap, locally linear embedding (LLE), t-distributed stochastic neighbour embedding (t-SNE), and neural networks. Table 2.2 lists a set of popular dimensionality reduction techniques and compares their characteristics.

Principal component analysis (PCA) is perhaps the most popular linear dimensionality reduction technique. PCA tries to find the linear projection that maximises the variance of the projected data, so as to keep as much information in the original data as possible. However, it is mainly concerned with large pairwise distances and is sometimes not appropriate to use, for example Figure 2.9 shows two cases where PCA is not an appropriate choice.

The structure in Figure 2.9(a) can be learned with linear discriminant analysis (LDA), which learns with label information and tries to maximise the separation between classes. LDA is further described in Section 2.4.2. Non-linear techniques such as Isomap, LLE and t-SNE are needed to learn the structure in Figure 2.9(b). Among

(a) (b)

Figure 2.9: *Two examples where PCA is not an appropriate choice.(a) when the separation rather than the variance of the data is more of interest to learn (Czech Technical University in Prague, 2008); (b) when the data is not Gaussian-distributed (Yang, 2007).*

them t-SNE is the best to use for visualisations (van der Maaten and Hinton, 2008).

Table 2.2: *A review of several popular dimensionality reduction techniques.*

Technique	**Linearity**	**Learning**	**Aim**	**Main assumption**
PCA	Linear	Unsupervised learning	To keep as much information in the original data as possible by seeking the projection that maximises the variance of the projected data along each component.	The data has a Gaussian distribution.
LDA	Linear	Supervised learning	To make data easy to classify in the lower dimensional space by seeking the projection that maximises the separation between classes.	Data in each class has a Gaussian distribution with the same covariance but different means; The data can be categorised to $d+1$ classes.
Isomap	Non-linear	Unsupervised learning	To map data to lower dimensional spaces while maintaining geodesic distances between all points.	Geodesic distances between two local points on the manifolds can be approximated by Euclidean distance.
LLE	Non-linear	Unsupervised learning	To map data to lower dimensional spaces while keeping the local structure in the original space.	In the original space, each data point can be represented by its neighbour points with linear weights at a reasonable accuracy.
t-SNE	Non-linear	Unsupervised learning	To map data to 2-dimensional or 3-dimensional spaces with emphasising local structures in the original space.	In the original space, one point obeys a Gaussian distribution centred at any other point of the data; In the low-dimensional space, the pairwise distances obey a Student's t-distribution with one degree of freedom.

next page

Table 2.2(continued)

Technique	**Linearity**	**Learning**	**Aim**	**Main assumption**
NN (auto-encoder)	Non-linear	Unsupervised learning	To reconstruct the NN input.	Neural networks are capable of learning structures in the high-dimensional data.
Bottleneck NN	Non-linear	Supervised learning	To learn from labelled data, such as to differentiate between classes.	Neural networks are capable of learning structures of the high-dimensional data.

Neural networks with compressed hidden layers, i.e. what we call bottleneck layers can be used for dimensionality reduction. The part from the high-dimensional input to the compressed layer defines a mapping from the high-dimensional space to a lower dimensional space. Auto-encoders learn in an unsupervised manner with the data only, trying to preserve as much information as possible in the original data for reconstruction. This is like what PCA does but in a non-linear way. Classification neural networks, on the other hand, learn with class labels aiming to distinguish between classes. This is like what LDA does but in a non-linear way.

Neural networks have been discussed in Section 2.2. In this thesis, LDA and t-SNE are also used and therefore will be described in detail in Section 2.4.2 and 2.4.3.

2.4.2 Linear discriminant analysis (LDA)

Linear discriminant analysis (LDA) is a widely used dimensionality reduction technique. As mentioned in Table 2.2, it aims to find a projection that maximises the separability of known categories. LDA was first proposed by (Fisher, 1936) for two-class problems in which data are projected from a D-dimensional space to a 1-dimensional space, and was later generalised for multi-class problems (for example, Rao, 1948).

To obtain a projection vector W that projects D dimensional data to d dimensional, the main idea is to find d eigenvectors that most dominate the ratio of the between-class and within-class covariance. By introducing a between-class covariance S_b and a within-class covariance S_w, the ratio of the between-class covariance and within-class covariance is

$$\mathbb{S} = S_b S_w^{-1}. \tag{2.43}$$

S_w and S_b are also called within-class and between-class scatters (scatter matrices). A decomposition of the form $\mathbb{S} = VAV^T$ can be applied, where the eigenvalues in A indicate the ratio of the between-class over within-class variance along the direction of the corresponding eigenvectors. The d eigenvectors corresponding to the largest d

eigenvalues are selected to form the projection vector W.

For a c-class problem, the within-class covariance is calculated as

$$S_w = \sum_{i=1}^{c} S_i, \tag{2.44}$$

where c is the number of classes, and S_i is the scatter matrix of class i,

$$S_i = \sum_{x \in i} (x - m_i)(x - m_i)^T, \tag{2.45}$$

where m_i is the mean of the vectors in class i;

The between-class covariance is

$$S_b = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T, \tag{2.46}$$

where N_i is the number of samples in class i, and m_i and m are the mean of each class and the overall mean, respectively.

2.4.3 t-distributed stochastic neighbour embedding (t-SNE) visualisation

t-Distributed Stochastic Neighbour Embedding (t-SNE) is a non-linear dimensionality reduction algorithm that was developed by (van der Maaten and Hinton, 2008) and is particularly well suited for the visualisation of high-dimensional datasets.

The t-SNE algorithm tries to map the original data space R^D to a 2-dimensional or 3-dimensional space R^d (we call it the map space or the map), where D and d are the dimensionality of the spaces. We denote a data point in the original data space as $x_i \in R^D$ and a map point in the map space $y_i \in R^d$. There is a bijection between the data points and the map points. If two data points x_i and x_j are close in R^D, we want the two corresponding map points y_i and y_j also to be close.

To measure how close (i.e. how similar) the two data points are, first a conditional

similarity $p_{j|i}$ is defined:

$$p_{j|i} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)}, \tag{2.47}$$

where $\|x_i - x_j\|$ is the Euclidean distance between x_i and x_j, and σ_i is the variance of the Gaussian distribution around x_i. This variance is set differently for every point to keep the perplexity fixed and thus adapt to different densities in different parts of the space. A larger perplexity indicates a bigger amount of data concerned in the neighbourhoods.

Then the similarity is defined as a symmetrised version of the conditional similarity:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}, \tag{2.48}$$

where N is the number of samples in the whole. In this way, we get a similarity matrix for the original space, which is obviously a symmetric matrix.

The similarity matrix for the map space is defined by

$$q_{ij} = \frac{f(\|y_i - y_j\|)}{\sum_{k \neq i} f(\|y_i - y_k\|)}, \tag{2.49}$$

where

$$f(z) = \frac{1}{1 + z^2}. \tag{2.50}$$

This is the same idea as for the original space, but with a Student's t-distribution (with one degree of freedom) instead of a Gaussian distribution. The t-distribution differs from the Gaussian in a way that forces close points (in the dense region) to be closer (denser) and in the sparse region (tails) further apart (sparser).

Intuitively, we want the values in the two similarity matrices q_{ij} and p_{ij} to be as close as possible, so that the structure of the map to be as similar as possible to the structure of the data. The difference between the two similarity matrices q_{ij} and p_{ij} is

measured by Kullback Leibler divergence:

$$KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}. \tag{2.51}$$

Points in the map space are moved around to minimise the $KL(P||Q)$, and this is done by gradient descent which has been described in Section 2.2.2. The asymmetric measurement of the difference leads to a learning that focuses on local structures. For example, when a large p_{ij} is modelled by a small q_{ij}, it leads to a big penalty, but when a small p_{ij} is modelled by a large q_{ij}, the penalty would be small.

One limitation of the t-SNE algorithm described above is that it is not efficient when dealing with large data set (for example, more than 5000 objects) as it needs to consider all pairwise interactions between points in every gradient update which would be very computationally intensive. For large datasets, t-SNE can be implemented via Barnes-Hut approximations. Basically, what the Barnes-Hut algorithm does is to find a set of "centre" points, with each one $\hat{y_j}$ representing a cluster of points that are very close to each other but are relatively far away from the point y_i that is of interest, so that the interaction between this point y_i and the cluster of points can be approximated as the interaction between $\hat{y_j}$ and y_i times the number of points involved in the cluster. The Barnes-Hut t-SNE is able to learn embeddings of data sets with millions of objects.

It should also be noticed that t-SNE applies a non-parametric learning, which means there is no such training process that learns mapping functions or models. Therefore, if one wants to include new data, the whole process has to be restarted from the very beginning.

2.5 Topological Manifolds

In mathematics an n-dimensional manifold is a topological space that is locally equivalent to n dimensional real Euclidean space $\mathbb{R}^n$ (see example, Lee, 2010). A simple example of a 1-dimensional manifold is a circle C embedded in the plane (Figure 2.10),

because any point on C has a neighbourhood that can be "straightened out" to be an open interval in $\mathbb{R} = \mathbb{R}^1$. However note that C cannot be embedded as a whole as a subset of $\mathbb{R}^1$.

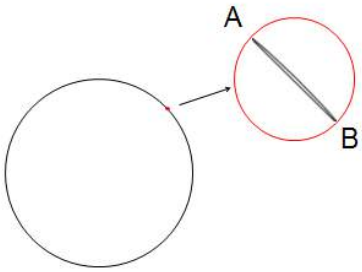


Figure 2.10: *The distance between two neighbour points A and B is calculated as Euclidean distance.*

Formally, a manifold consists of a topological space M such that for each $x \in M$ there is a neighbourhood U_x and bijection $\phi_x : U_x \to \mathbb{R}^n$ that establishes the equivalence between U_x and $\mathbb{R}^n$. Normally additional restrictions are placed on ϕ_x to ensure that it preserves relevant mathematical structure. Thus, in topology ϕ_x would be a homeomorphism (ϕ_x and ϕ_x^{-1} are both continuous) but for the purposes of calculus it would need to be a diffeomorphism (ϕ_x and ϕ_x^{-1} are also both differentiable). There is also a "consistency" property. If $x, y \in M$ and $U_x \cap U_y \neq \emptyset$ then $\phi_x \phi_y^{-1} : \phi_y(U_x \cap U_y) \to \phi_x(U_x \cap U_y)$ is a bijection with the same additional properties as ϕ_x and ϕ_y that ensures that the overlap $U_x \cap U_y$ is treated equivalently by ϕ_x and ϕ_y.

In speech processing there are a number of computational models where an acoustic space M is embedded into $\mathbb{R}^n$ for some n by a single global mapping ϕ. For example, in speaker or language identification M is the set of sequences of acoustic vectors corresponding to a spoken utterance, $\phi : M \to \mathbb{R}^n$ maps $x \in M$ to its i-vector $\phi(x)$, or M is the set of acoustic feature vectors in context, $\phi : M \to \mathbb{R}^n$ is implemented by a DNN and $\phi(x)$ is a bottleneck feature representation of $x \in M$. In contrast, the linear model described in (Huang et al., 2016) is one of few examples which attempt to exploit the varying local structure offered by a manifold. In their work they defined a set of overlapping neighbourhoods according to phone classes and applied linear learning to

each of them to achieve a phone classification task. Our work presented in Chapter 7 is also inspired by the manifold structure and can be regarded as a non-linear extension of the work in (Huang et al., 2016).

Chapter 3

Speech Corpus

This chapter describes the speech corpus used throughout the thesis - TIMIT.

3.1 TIMIT corpus

The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT - Texas Instruments/Massachusetts Institute of Technology) is a corpus of read speech (Garofolo et al., 1993). It contains recordings of phonetically-balanced prompted English speech, and was recorded at 16 kHz rate with 16 bit sample resolution using a Sennheiser close-talking microphone.

TIMIT contains a total of 6300 sentences (5.4 hours), from 630 speakers representing 8 major dialect divisions of American English, each speaking 10 phonetically-rich sentences. 70% of the speakers are male and 30% are female.

The prompts for the 6300 utterances consist of 2 dialect sentences (SA, spoken by all speakers), 450 phonetically compact sentences (SX) and 1890 phonetically-diverse sentences (SI). Each sentence has an associated orthographic transcription, time-aligned word boundary transcription and time-aligned phonetic transcription where a 61-phone Arpabet-based phone list is used (The phone list is included in Table 3.2). The phone level segmentation was done manually.

Train and test sets are suggested in the TIMIT corpus. A "core test set" is given

as an abridged test set which includes 192 utterances from 24 speakers (2 males and 1 female from each dialect). We followed the train/test division but excluded all SA sentences in both training and test process in order to avoid any bias due to their identical content. Table 3.1 shows details of the train, full test and core test set.

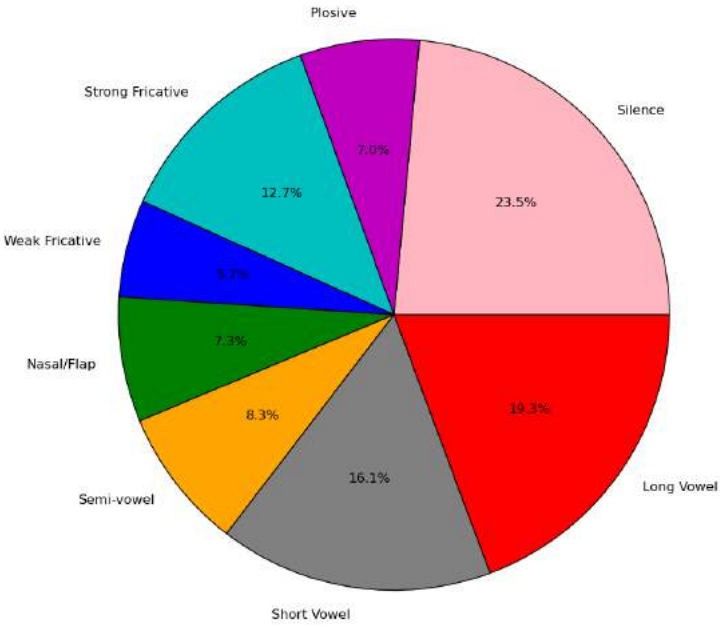
Table 3.1: *Number of speakers, utterances, hours and tokens of speech in TIMIT training, full test, development, and core test sets, excluding SA sentences.*

Set	# speakers	# utterences (non-SA)	# hours	# tokens
Train	462	3696	3.14	142910
Test (full)	168	1344	0.81	51680
Development test	50	400	0.34	15334
Core test	24	192	0.16	7333

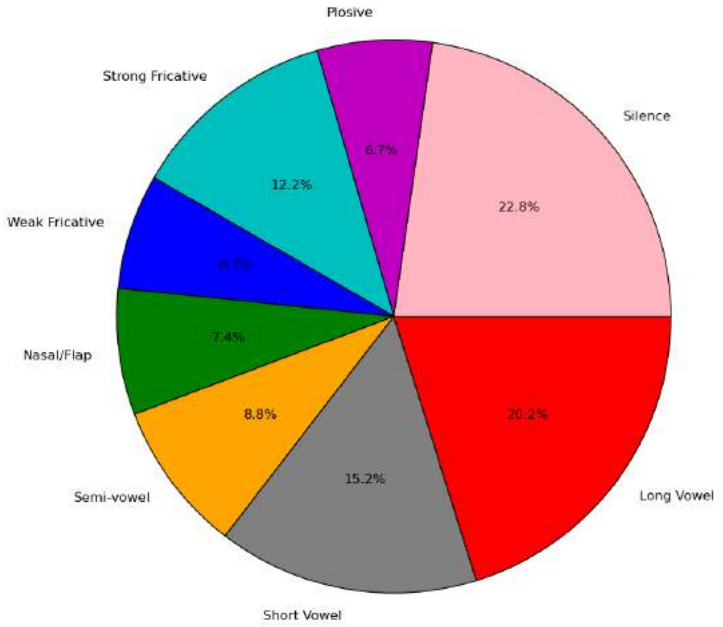
Using the categorisation described in Table 2.1 in Section 2.3.1, we calculated the proportion of each category in the training set[1], presented in Figure 3.1.

It is true that the TIMIT corpus is a small dataset and may seem "old fashioned" from the perspective of volume and recording method. However, it has always been a significant database in tasks with phonetic targets, as it is the most widely used dataset with hand labelling. Especially in recent years when the DNN approaches get popular, the TIMIT dataset is of great use having faithful labellings and there are many baseline DNN-based development using TIMIT. Particularly, the TIMIT corpus is preferred in research with emphasis on science discovery, whereas for works focusing more on practical use, it may be beneficial to use bigger corpus to train systems. The research in this thesis belongs to the former, therefore the TIMIT corpus was chosen as the speech corpus.

[1]This is calculated by counting the number of frames resulting from the front-end analysis sampling at 16 kHz using a 25ms-width window with a 10ms frame rate



(a) TIMIT training set (non-SA)



(b) TIMIT core test set (non-SA)

Figure 3.1: *The proportion of each broad phone category (calculated by the number of sampling frames) in TIMIT training (a) and core test (b) set.*

3.2 TIMIT phone labels and phone mappings

In the TIMIT database, there are phone-level labels with segmentation information. TIMIT uses a 61-symbol set which is shown in the first column in Table 3.2. When using the TIMIT corpus, the usual convention is to use the 49-symbol set (Holmes, 1997) to build models and score on the 40-symbol set (Lee and Hon, 1989), as TIMIT makes distinctions between very similar sounds, whose confusion is not considered important in the context of ASR.

Table 3.2 lists TIMIT phone labels and phone mappings that we use to map the 61-symbol set in TIMIT labels to the 49-symbol set and to the 40-symbol set in this thesis.

In many experiments in this thesis, we use the phone segmentations provided in the TIMIT database. Particularly, in some of the phone classification experiments, we use only feature vectors corresponding to the centre frames of the TIMIT phone segments (this will be described in detail in Section 7.2.2).

Table 3.2: *TIMIT phones and mappings to the 49 and 40 symbol sets.*

61-symbol set	**49-symbol set**	**40-symbol set**
ih	ih	ih
ix	ix	
iy	iy	iy
ae	ae	ae
ah	ah	ah
ax-h		
ax	ax	
eh	eh	eh
uh	uh	uh
uw	uw	uw
ux		
aa	aa	aa
ao	ao	
er	er	er
axr		
ey	ey	ey
ay	ay	ay
oy	oy	oy
aw	aw	aw
ow	ow	ow
p	p	p
t	t	t

next page

Table 3.2(continued)

61-symbol set	**49-symbol set**	**40-symbol set**
k	k	k
b	b	b
d	d	d
dx	dx	dx
g	g	g
pcl	cl	sil
tcl		
kcl		
bcl	vcl	
dcl		
gcl		
q	q	
pau	sil	
h#		
epi	epi	
f	f	f
v	v	v
s	s	s
z	z	z
sh	sh	sh
zh	zh	zh
th	th	th
dh	dh	dh
ch	ch	ch

next page

Table 3.2(continued)

61-symbol set	**49-symbol set**	**40-symbol set**
jh	jh	jh
m	m	m
em		
n	n	n
nx		
en	en	
ng	ng	ng
eng		
l	l	l
el	el	
r	r	r
w	w	w
y	y	y
hh	hh	hh
hv		

Chapter 4

Very Low-dimensional Bottleneck Neural Network Representation of Speech

4.1 Introduction

As mentioned in the background in Section 1.1, there is a need for a compact representation of speech, suitable for segmental models, that can be estimated for all speech sounds. In this chapter, low-dimensional features extracted from the bottleneck of neural networks are analysed, with the focus on their capability of representing speech and their suitability for modelling speech dynamics. Various ways of designing and training the network are assessed. This includes varying: 1. the input to the neural network; 2. the neural network output target, i.e. neural network function; 3. the size of bottleneck and intermediate hidden layers. Bottleneck features (BNFs) are evaluated by a GMM-HMM phone recogniser (described in Section 2.1.2) and are compared with MFCC features and formant features in respect to ASR accuracies. The suitability of the BNFs to be used in a Continuous-State HMM (CS-HMM, described in Section 2.1.3) system is assessed. Experimental evaluations are performed on the

TIMIT speech corpus (described in Section 3.1).

We demonstrate that in a conventional HMM-based ASR system, 9-dimensional BNFs can give comparable phone recognition performance to 39-dimensional conventional MFCCs, and BNFs give on average 33.7% improvement in phone accuracies compared with formant-based features of the same dimensionality. It is observed that the BNFs preserve well the trajectory continuity and fit the CS-HMM modelling better than formants. The BNFs provide a compact representation in terms of the number of model parameters and they seem overall to be well suited to be employed for segmental models of speech dynamics.

Section 4.2 describes the experimental setup, explaining briefly how the BNFs are obtained and evaluated. Section 4.3.1 to Section 4.3.5 analyses the bottleneck representation of speech using conventional GMM-HMM recognisers. Section 4.3.6 explores if the BNFs fit the assumption of CS-HMM for modelling speech dynamics.

4.2 Experimental setup for extracting very low dimensional BNFs

4.2.1 Bottleneck neural network structure

The architecture of the bottleneck neural network that was used is depicted in Figure 4.1. BNFs are the output of the compression layer of a bottleneck neural network. There could be either no layers or several layers between the bottleneck layer and the input/output layer.

In experiments in this thesis, the input to the network is a vector containing the 26 Mel-scaled logarithm filter-bank energies (logFBEs), implemented based on the Fourier transform. The TIMIT corpus sampled at 16 kHz was analysed using a 25-ms Hamming window with a 10-ms frame rate.

Two approaches to training the network were explored, one aimed at the recon-

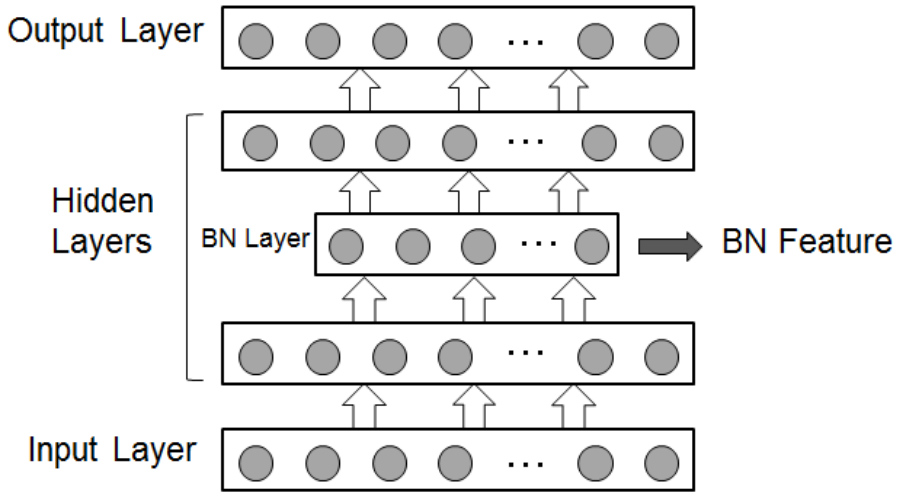


Figure 4.1: *Architecture of the multi-layer bottleneck neural network employed for speech representation.*

struction of the input spectrum and the other at discrimination between phones. In the case of reconstruction, when the input logFBEs cover more than one signal frame, i.e. contain context frames, the target is either the logFBEs of the current signal frame or the full context (same as the input). In the case of phone discrimination, the targets are the posterior probabilities of the 49 phones. These were obtained based on the labels and time-stamp information supplied with TIMIT. The mapping of 61 to 49 phones and the mapping of 49 to 40 phones were performed as described in (Holmes, 1997; Lee and Hon, 1989). The mappings are described in Section 3.2. The Softmax function was used at the last layer in the phone discrimination neural networks.

4.2.2 Neural network training

Filter bank energies of different forms were used as the input to the neural networks. They were normalised to have zero mean and unit variance based on the entire training set. 90% of the TIMIT training set (random 90% of the training sample frames for each gender in each dialect) was used as the neural network training set, and the remaining 10% as validation set.

The neural networks were trained using the Theano Toolkit (Bastien et al., 2012) with GPU computing (Bergstra et al., 2010). The use of GPU enables much faster neural network training without affecting the training results. Stochastic gradient descent was used as the back-propagation training algorithm.

The learning rate was set to 0.01 and the maximum number of epochs was set to 3000. These were chosen empirically after exploring suitable values. The training stopped when the error on the validation set started to rise or when the epoch reached the maximum. In most cases, the error was still decreasing very slowly when the epoch reached the maximum of 3000, but the error curve was almost flat and the classification error would reduce very little if we continue training.

4.2.3 Evaluation of bottleneck outputs with GMM-HMM recognisers

We evaluated BNFs quantitatively on their capability of representing speech, using an ASR phone recognition system. Speech recognition experiments were performed using a standard GMM-HMM system built using HTK (Young et al., 1997). BNFs were fed into the recognition system as acoustic features. HMMs were built for the 49 phone set (Holmes, 1997), each model consisting of 3 states. The number of GMM components per state was set to increase from 1 in powers of 2 up to 512. The number of components for silence was twice that for phones. A bi-gram language model was used. For evaluating recognition performance, recognition rates were scored on the 40 phone set which was mapped from the 49 set according to (Lee and Hon, 1989) as described in Section 3.2. Percent correct and percent accuracy, shown as "%Corr" and "%Acc" in the tables, were calculated following equations 2.3 and 2.4. The reported results, if not specified, are on the core test set using the number of GMM components corresponding to the best accuracy achieved on the development test set.

4.2.4 Continuous-State HMM trajectory modelling

In this work we use a Continuous-State Hidden Markov Model (CS-HMM, as described in Section 2.1.3) to recover the dwell-transition trajectory that best fits the features, ignoring phone targets. To do this we supply the model with an inventory with a flat prior, consisting of a single phone target with very high variance. The top hypothesis with the same number of dwells as there are TIMIT labelled phones in the utterance was returned. Note that a distinguishing feature of the dwell-transition CS-HMM used in this work is that continuity is preserved across the segment boundaries.

4.3 Experiments and results

4.3.1 Comparisons between networks with different network inputs and network functions

Varying context window widths were assessed on reconstruction neural networks and phone discrimination neural networks. For both the neural network structures, the 49 phone set was used when training neural networks and HMM models then the 40 phone set was used when scoring. Table 4.1 and Table 4.2 show how BNFs perform in GMM-HMM recognisers when using various widths of context windows in the neural network input. All networks used three hidden layers, and are denoted by the number of neurons at each layer, for example, "26-32-4-32-26" represents a neural network in which the numbers of neurons at the input layer, three hidden layers and the output layer are 26, 32, 4, 32 and 26 respectively.

Table 4.1 indicates that including context input in the reconstruction neural networks has little effect on the ASR recognition performance using BNFs. However, having context input in the phone discrimination networks does improve the ASR performance as the width of context window increases from 0 to 15 frames, as shown in Table 4.2. We use the context window width of ± 5 frames in the following experiments

Table 4.1: *ASR performance using BNFs extracted from reconstruction neural networks in GMM-HMM recognisers when varying the width of context window at the input layer.*

Exp	Context frames	NN target	NN size	monophone	
				%Corr	%Acc
A1	0	centre frame logFBEs	26-32-4-32-26	43.44	40.93
A2	±1	centre frame logFBEs	78-32-4-32-26	43.00	40.85
A3	±2	centre frame logFBEs	130-32-4-32-26	42.24	40.27
A4	±5	centre frame logFBEs	286-32-4-32-26	43.00	40.68
A5	±1	same as input	78-32-4-32-78	44.05	41.37

in this thesis. This was chosen with consideration of a balance between reasonable error rate and efficiency of experiments (time and memory). The context window width of ±5 frames gives reasonable error, and the results in this thesis are not dependent on having the absolute minimum error here.

Table 4.2: *ASR performance using BNFs extracted from phone discrimination neural networks in GMM-HMM recognisers when varying the width of context window at the input layer.*

Exp	Context frame	NN size	monophone	
			%Corr	%Acc
B1	0	26-32-4-32-49	59.84	54.63
B2	±1	78-32-4-32-49	61.33	56.85
B3	±2	130-32-4-32-49	61.97	57.18
B4	±5	286-32-4-32-49	63.03	58.52
B5	±6	338-32-4-32-49	63.56	59.44
B6	±7	390-32-4-32-49	63.63	59.80

4.3.2 Effect of hidden layer sizes

Next, the effect of varying the number of neurons in hidden layers of the network was explored. Experiments were carried out using the phone discrimination network. In this part, all networks used three hidden layers, denoted as H-B-H, where 'B' stands for the bottleneck layer. The number of neurons in both "H" hidden layers was fixed to 32, 128, 512, 1024, and 2048 in turn. The number of neurons in the bottleneck layer was set to 4, 9, 16, and 32. Experiments used the context of 5 frames before and 5 frames after the current frame in the input layer, i.e. the input layer was of size 286. The output layer of the phone discrimination NNs was of size 49. The results achieved

by monophone HMMs are listed in Table 4.3, and a corresponding graph is depicted in Figure 4.2.

Table 4.3: *Phone recognition performance using BNFs when varying the number of neurons in the bottleneck and other hidden layers using phone-posterior network.*

Exp	NN: 286-H1-B2-H3-49		monophone	
	H1, H3	B2	%Corr	%Acc
1	32	4	62.18	58.17
2	32	9	69.71	65.61
3	32	16	69.96	65.88
4	32	32	70.27	65.96
5	128	4	65.85	61.88
6	128	9	70.88	67.61
7	128	16	72.88	69.69
8	128	32	73.60	69.94
9	512	4	68.51	64.55
10	512	9	73.70	70.08
11	512	16	74.16	71.01
12	512	32	73.30	70.70
13	1024	4	68.21	64.14
14	1024	9	74.04	70.21
15	1024	16	73.84	70.82
16	1024	32	74.79	71.01
13	2048	4	68.31	64.60
14	2048	9	73.47	70.57
15	2048	16	73.71	70.62
16	2048	32	73.87	70.58

For phone posterior neural networks, it can be seen that a considerable improvement in ASR performance is obtained when the bottleneck layer increases from 4 to 9 neurons. Increasing the bottleneck further beyond 9 neurons gives only minor improvements. Increasing the size of the other two hidden 'H' layers from 32 to 512 also gives a great improvement in ASR performance, but only minor improvements are seen when the size is above 512.

Note also that in experiments described in Section 7.3.1, we found that the position of the bottleneck layer also affects the ASR performance using BNFs, and that training DNNs with deep belief networks (DBNs) provides a small improvement of the ASR performance. In the next chapters however, we use the NN structure of 286-512-B-512-

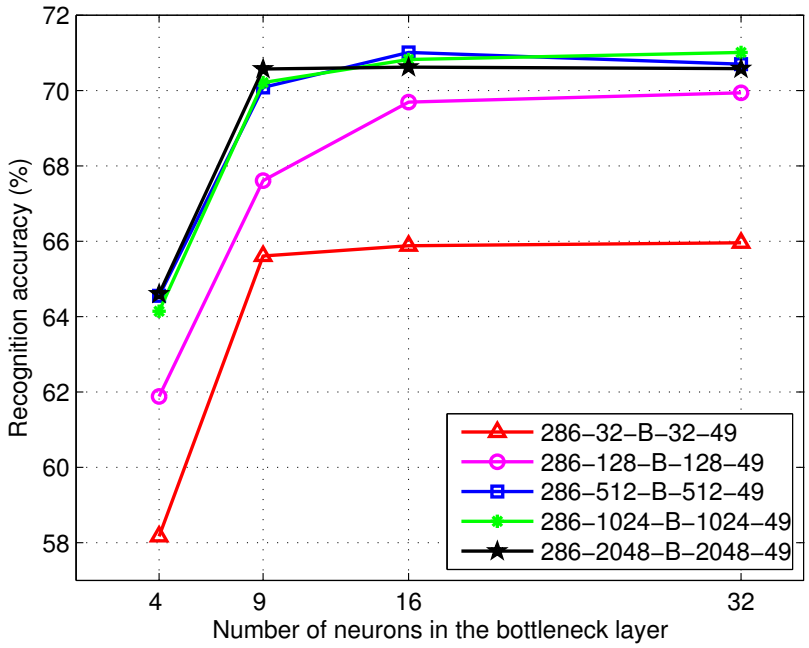


Figure 4.2: *Phone recognition accuracy using BNFs as a function of the number of neurons in the bottleneck and other hidden layers when using phone-posterior network.*

49 trained by standard back-propagation, where B, the size of the bottleneck layer, varies in different experiments. Although the resulting ASR performance is slightly lower, it does not affect how we analyse the BNFs in the relevant chapters.

4.3.3 Comparison between monophone and triphone models

Table 4.4 and Figure 4.3 show the ASR performance with BNFs obtained from the phone-posterior network when using monophone and triphone modelling in the ASR system. The network configuration 286-512-B-512-49 was used, varying the bottleneck layer size.

Table 4.4: *Phone recognition accuracy using BNFs extracted from phone-posterior network 286-512-B-512-49 when varying the size of the bottleneck layer.*

Exp	286-512-X-512-49	monophone		GMM-HMM triphone	
	X	%Corr	%Acc	%Corr	%Acc
1	4	68.51	64.55	64.69	59.63
2	9	73.70	70.08	72.27	67.72
3	16	74.16	71.01	75.01	70.14
4	32	73.30	70.70	75.13	70.40

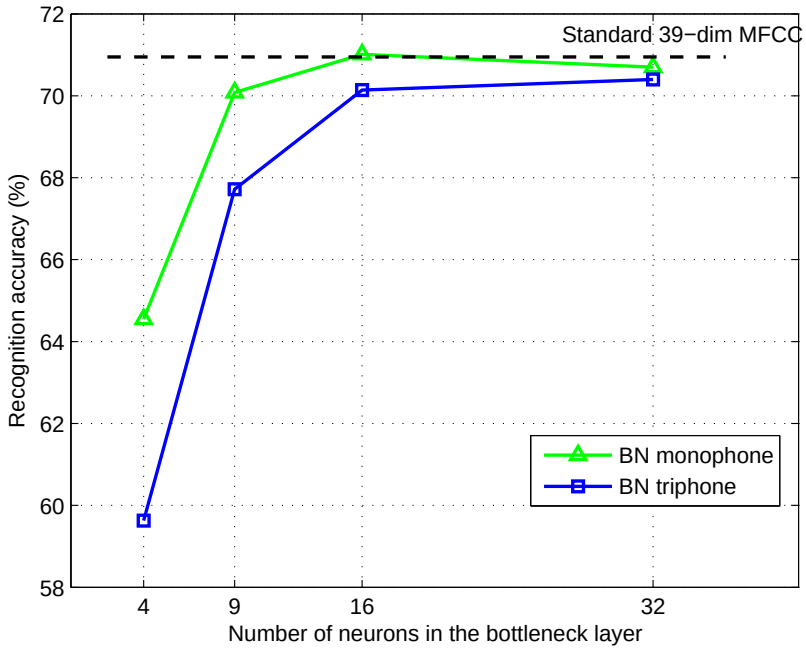


Figure 4.3: *Phone recognition accuracy using BNFs extracted from phone-posterior network 286-512-B-512-49 when varying the size of the bottleneck layer.*

Interestingly, the use of triphone models gives lower recognition accuracies than monophone models, especially for very low-dimensional BNFs. This suggests that the contextual information which is required by triphone modelling in the ASR process may have been discarded to some extent during the neural network training due to the very low-feature dimensionality. Also the network was trained to distinguish between monophones, and this could also have made the neural network more suitable for monophone recognisers.

4.3.4 BNFs with delta and delta-deltas

Table 4.5 and Figure 4.4 compare the ASR performance using BNFs obtained from the phone discrimination network when using static and static plus delta information in the ASR modelling. For the latter, we calculated deltas on the BNFs, as part of the ASR system training. The network configuration 286-512-B-512-49 was used, varying the bottleneck layer size.

It can be seen that appending Δ and $\Delta\Delta$ to BNFs gives improvement between 2%

Table 4.5: *Phone recognition accuracy using BNFs extracted from phone-posterior network 286-512-B-512-49 when varying the size of the bottleneck layer.*

Exp	286-512-X-512-49	with BNF		with BNF+Δ+ΔΔ	
	X	%Corr	%Acc	%Corr	%Acc
1	4	68.51	64.55	73.51	68.31
2	9	73.70	70.08	76.89	72.84
3	16	74.16	71.01	78.02	73.10
4	32	73.30	70.70	78.20	72.93

to 4% (absolute), depending on the size of bottleneck. Note that the results obtained using 9-dimensional BNFs are better than using 4-dimensional BNFs appended by Δ and ΔΔ, i.e., 12-dimensional features. This suggests that the BNFs are containing information about both the spectral and temporal properties of speech.

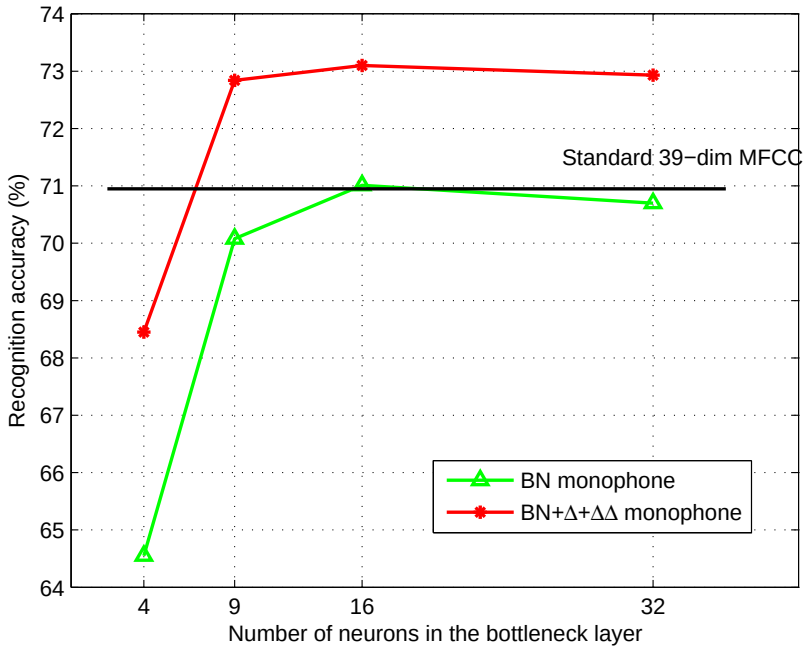


Figure 4.4: *Phone recognition accuracy using BNFs extracted from phone-posterior network 286-512-B-512-49 when varying the size of the bottleneck layer.*

4.3.5 Comparison between BNFs and formant data

The BNFs have been compared with MFCCs in the above sections. This section compares BNFs with formant features (the dimensions of the two types of features are

kept the same). Formant features are accepted to capture the true speech dynamics (in voiced sounds).

First, we compare the phone recognition performance using the bottleneck and formant features in a conventional HMM-based ASR system. Results are presented in Table 4.6. Bottleneck neural networks with 3 nodes at the bottleneck layer were trained and 512 nodes were used at the non-bottleneck layers. Experiments were performed on both phone discrimination neural networks and logFBEs reconstruction neural networks. The former achieved better results and as such only these are reported here. Experiments were performed with formants estimated using the Wavesurfer (Sjölander and Beskow, 2000) and Praat (Boersma and Weenik, 2013) tools. The former achieved better results and as such only these are reported here.

Comparing line 1 under the "Formants" section in Table 4.6 to line 1 under the "BNFs" section, we can see that the use of 3 BNFs obtained from the phone discrimination neural network considerably outperforms the use of 3 formant frequencies (around 20% absolute or 50% comparative advantage in recognition accuracy). It indicates that the BNFs are able to encode more information than formant frequencies, We hypothesise that this extra information may include amplitudes and bandwidths.

Thus we also performed experiments with formant-based features containing the formant frequencies, amplitudes and bandwidths. Comparing line 3 in the "Formants" section to line 1 and 3 in the "BNFs" section in Table 4.6, we see that the use of BNFs of both 3 and 9 dimensions considerably outperform the use of 3 formant frequencies plus amplitudes and bandwidths (9 dimensions in total). The use of 3 BNFs outperforms 9 formant features by 8.9% (absolute) in recognition accuracies, and BNFs of the same dimension (9-dimension) provide a further accuracy increase of 9.6% (absolute). Experiments were also performed with formant-based features containing the formant frequencies, amplitudes and bandwidths with delta features, resulting in a 27-dimensional feature vector (with delta and delta-delta) and compared with the same dimensionality BNFs (with delta and delta-delta). The results are in the bottom lines

in the sections "Formants" and "BNFs" in Table 4.6. Again, the use of BNF-based features provides much better ASR accuracy than formant-base features

It can be seen that BNFs considerably outperformed formant-based features. The use of the bottleneck-based feature representation results, on average, in a 33.7% comparative improvement in phone accuracies compared with formant-based features with the same dimension.

In addition, the confusion matrices from the recognition results showed that the BNFs achieved nearly uniform improvement over formant-based features across all phones. A figure illustrating ASR recognition accuracies of individual phones using 3 BNFs, 9 BNFs, 3 Formant frequencies and 3 Formant frequencies plus 3 amplitudes and 3 bandwidths is shown in Appendix A .

Table 4.6: *Recognition performance of an HMM-based ASR system when using formant or bottleneck feature representation.*

Feature representation	Dim.	Recognition	
		Corr (%)	Acc (%)
MFCC + Δ + $\Delta\Delta$	39	76.23	70.95
Formants			
3 freq	3	49.30	40.71
3 freq + Δ + $\Delta\Delta$	9	56.32	51.12
3 freq & amp & bw	9	55.96	52.04
3 freq & amp & bw+ Δ + $\Delta\Delta$	27	65.06	60.43
BNFs			
3 BNFs	3	65.02	60.94
3 BNFs + Δ + $\Delta\Delta$	9	70.87	65.73
9 BNFs	9	74.37	70.57
9 BNFs + Δ + $\Delta\Delta$	27	76.77	73.07

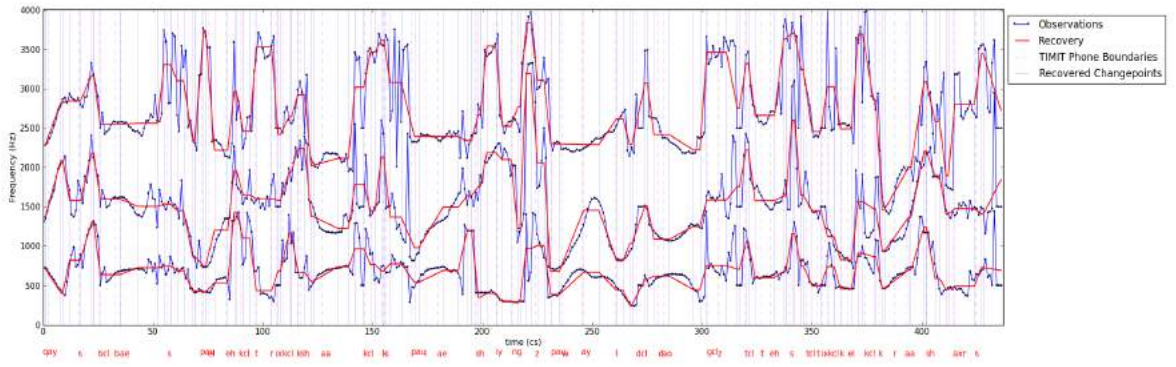
4.3.6 Analysis of the BNFs for modelling speech dynamics

Having demonstrated that bottleneck neural networks can be used to extract very low-dimensional speech features (i.e. BNFs) that perform relatively well for phone recognition in conventional ASR systems, we now try to analyse if the BNFs are suitable for modelling speech dynamics. Exploring speech dynamics is not the main scope of

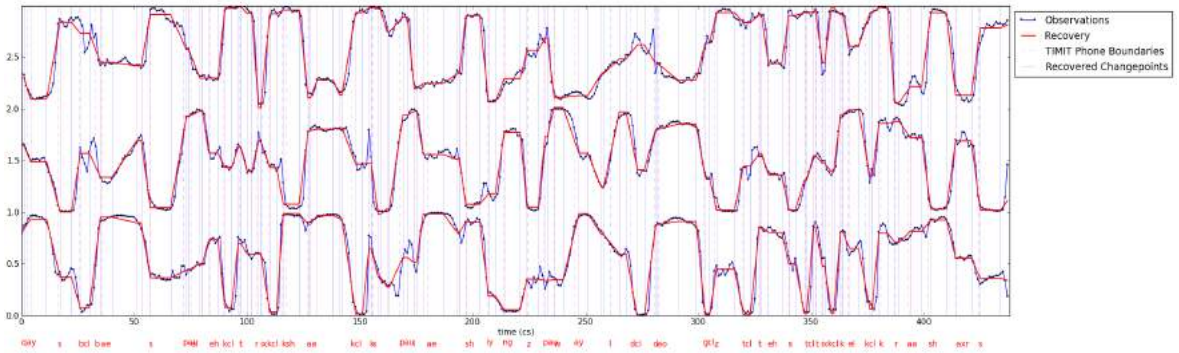
this thesis, so we will only be using the CS-HMM models, and we explore whether the BNFs fit the assumption of the CS-HMM models.

The CS-HMM has been described in Section 2.1.3. How we performed the experiment has been described in Section 4.2.4. Figure 4.5 shows an example of trajectories recovered by the CS-HMM when using formant frequencies (F1, F2, F3), 3-dimensional BNFs from the phone discrimination neural network and the logFBEs reconstruction neural network, top to bottom. The size of hidden layers in both NN structures is 512-3-512. The example utterance is TRAIN/DR1/MKLW0/SI1571. The beginning and ending silence "h#" has been removed for clearer display.

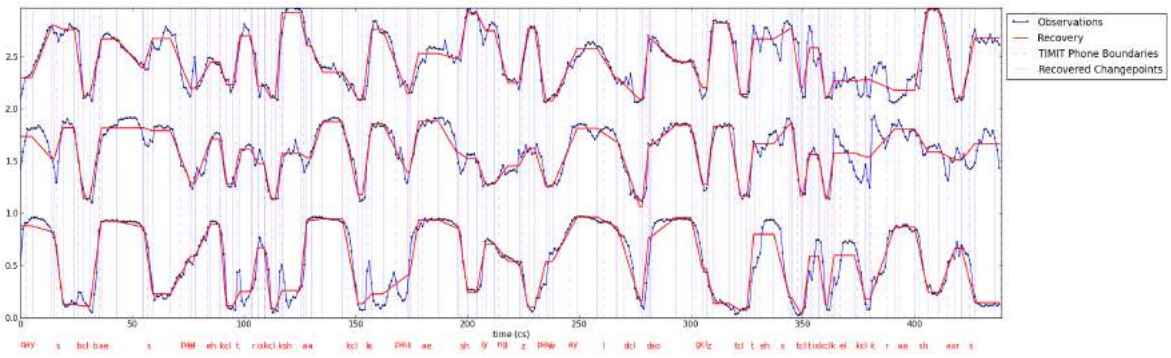
The formant frequencies are estimated using Wavesurfer (Sjölander and Beskow, 2000). The formants range from 0 to 4000 Hz, while the BNFs are in the range [0, 1] but plotted shifted on the vertical axis for clarity. These plots suggest that BNFs, especially those obtained from phone discrimination networks, fit the dwell-transition model (described in Section 2.1.3) considerably better than the formant features, for all of the speech sounds shown. While the formant trajectories are smooth in the voiced regions (where they can be more reliably estimated), they vary widely in the unvoiced regions. This seems to affect their fit to the model, since the CS-HMM algorithm tries to fit additional segments in the regions of high variability, necessitating a coarser fit in the low-variability regions, and an overall poorer fit to the trajectory.



(a)



(b)



(c)

Figure 4.5: *An example of the dwell-transition trajectories recovered by the CS-HMM when using estimated formant frequencies (a) and BNFs obtained from phone the discrimination neural network (b) and the reconstruction network (c). Blue lines with dots show the observations (feature values) and solid red lines the estimated dwell-transition trajectories. TIMIT phone boundaries are indicated by thin vertical lines, recovered dwell starts (magenta) and ends (blue) by vertical dashed lines.*

4.4 Summary and discussion

Segmental models of speech hold promise for speech recognition due to their ability to parsimoniously model speech dynamics. However they have been hampered by lack of a good representation. Formants model voiced sounds well, but are inappropriate for unvoiced speech, while articulatory parameters are difficult to obtain. This chapter presents results of ASR experiments using very low-dimensional BNFs extracted from neural networks, and an initial analysis of their temporal dynamics. The results show that when the networks are trained to predict phone posteriors, BNFs significantly outperform formant features of similar dimensionality, and 9-dimensional BNFs can give comparable ASR performance to 39-dimensional MFCCs in a conventional ASR system.

Chapter 5

Interpretation of Bottleneck Features and The Neural Network Learning Behaviour

5.1 Introduction

In this chapter, several approaches are applied to visualise bottleneck features (BNFs) and interpret BNFs and the neural network learning behaviour.

In Section 5.2, linear discriminant analysis (LDA) and t-distribution stochastic neighbour embedding (t-SNE) are applied to visualise 9-dimensional BNFs, and a neural network with a narrower bottleneck layer of 2 neurons is used to generate 2-dimensional BNFs that can be directly visualised. Through the BNF visualisations, it is shown that the BNFs obtained from phone discrimination neural networks convey phonetic-related information.

Section 5.3 explores some details of how the neural networks extract BNFs. We first propose a method to obtain representative BNFs for individual phones. Then we look at the neural network neuron responses at the bottleneck layer. We also applied LDA to non-bottleneck layer activations.

5.2 Visualisations of BNFs

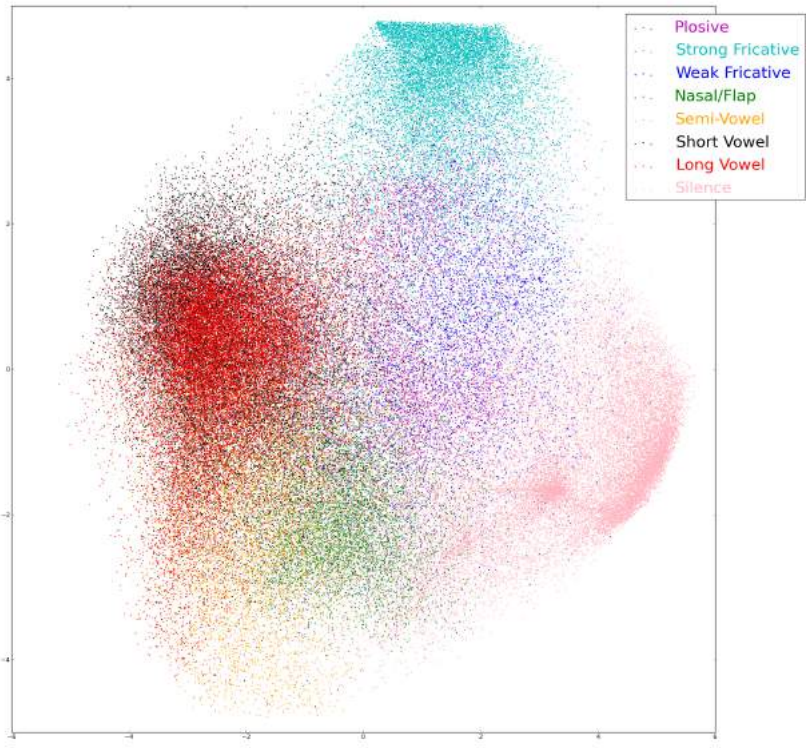
5.2.1 Visualisation of BNFs with LDA

In this subsection we show 2-dimensional visualisations of LDA projections of 9-dimensional BNFs obtained from a phone-discrimination neural network with layers of structure 286-512-9-512-49[1], which was trained in the same way as described in Chapter 4. The LDA mapping was trained on the same training set as used in the neural network training, and was later applied to the core test set.

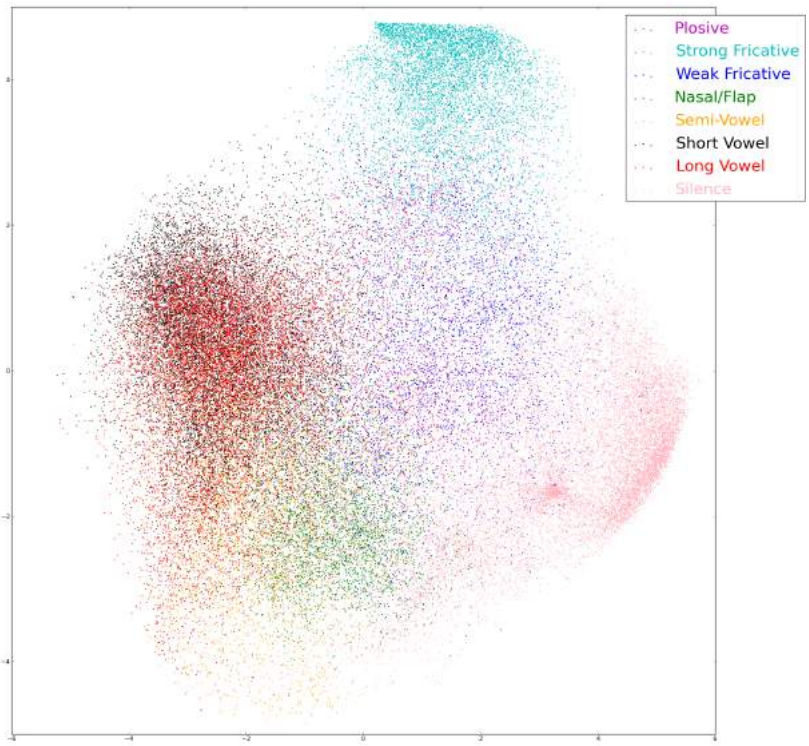
Both the BNFs for the training set and for the core test set were visualised. Figure 5.1 shows the 1^{st} and the 2^{nd} dimension of the LDA-based projections of 9-dimensional BNFs from a phone classification DNN. Figures (a) and (b) are plots of training set and core test set, respectively. For clarity, only a random 10% of the frames in the training set are plotted in Figure 5.1(a). The LDA mapping was learned based on the TIMIT labels of 49 phones (Holmes, 1997), before the projection was applied to the 9-dimensional BNFs of the corresponding dataset. The plotted points are coloured by their broad phone categories according to (Halberstadt and Glass, 1997).

We can see that the overall structures of the training and the test data are alike. For each category, there is a "cloudy" distribution of BNFs. Vowels, consonants and silences are fairly well separated. However there are many overlaps among the sub-categories of vowels, especially long vowels and short vowels, and among plosive and fricatives, especially plosive and weak fricatives. These overlaps indicate that the overlapping data are alike in some way and may lead to confusions in speech recognition using these features between the broad phone categories. In addition, the 1^{st} LDA dimension (horizontal axis) seems to indicate voicing, with voiced phones on the left and unvoiced on the right. Moving from left to right, we observe vowels, nasals, then fricatives and plosives, and finally silences.

[1] As defined in Section 4.3.1, "286-512-9-512-49" represents a neural network in which the numbers of neurons at the input layer, three hidden layers and the output layer are 286, 512, 9, 512 and 49 respectively.



(a) BNFs of the TIMIT training set (random 10%)

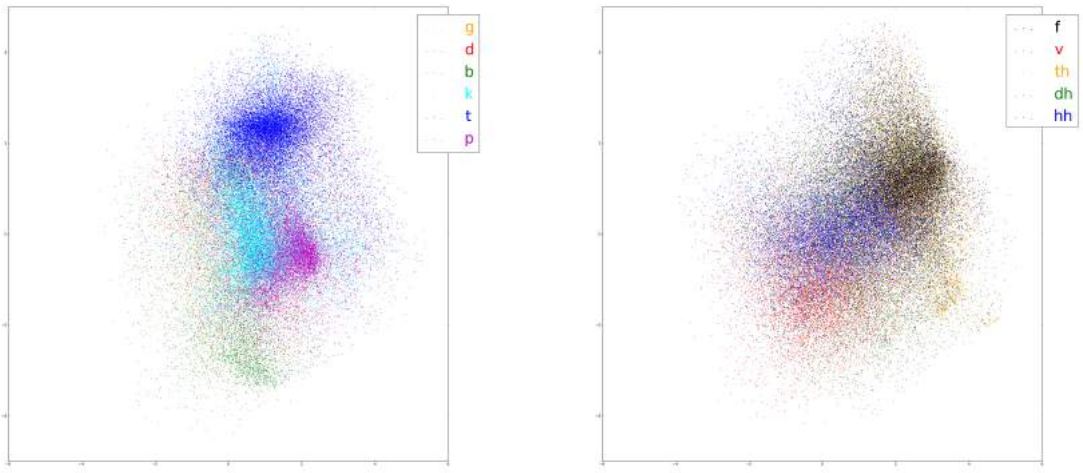


(b) BNFs of the TIMIT core test set

Figure 5.1: *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of 9-dimensional BNFs from a phone classification DNN of structure 286-512-9-512-49. Horizontal axis: the 1^{st} dimension of LDA projections; vertical axis: the 2^{nd} dimension of LDA projections.*

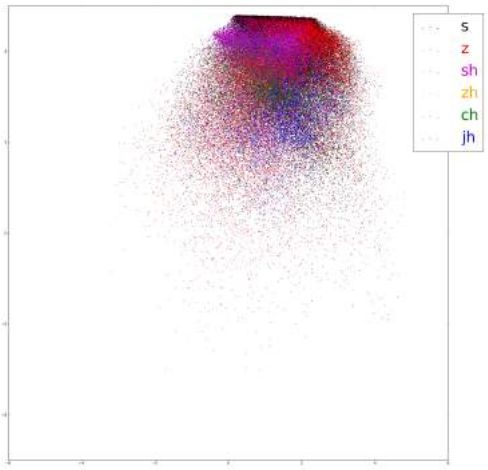
Figure 5.2 shows the 1^{st} and the 2^{nd} dimension of LDA visualisations for each category. Each figure represents a phone category, with individual phones coloured differently. We only show plots of the training set, as the test set has approximately the same structure. For clarity, the plot for silence is on a random 10% of the training set for a reduced number of points, while others are on the full training set.

We can see some clusters within phone categories, though not as clear as the structure between them as shown in Figure 5.1. It seems that the BNF features separate out to some extent the individual phones, but there are considerable overlaps, especially in figures for strong fricatives, nasal/flaps, short vowels.

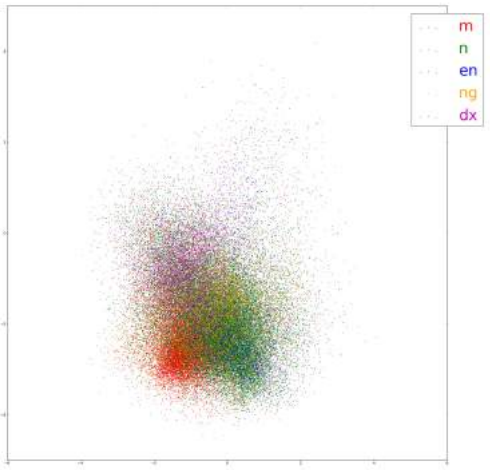


(a) Plosive (b) Weak Fricative

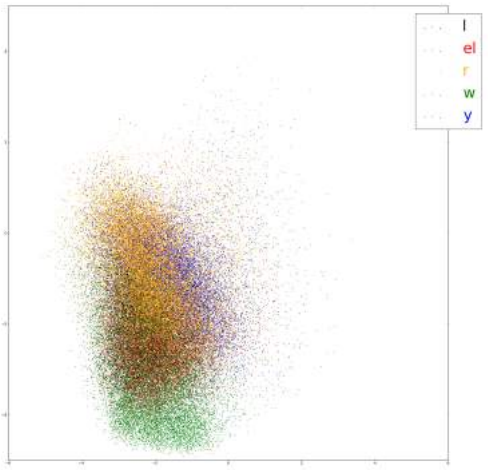
Figure 5.2: *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of 9-dimensional BNFs from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure. Horizontal axis: the 1^{st} dimension of LDA projections; vertical axis: the 2^{nd} dimension of LDA projections.*



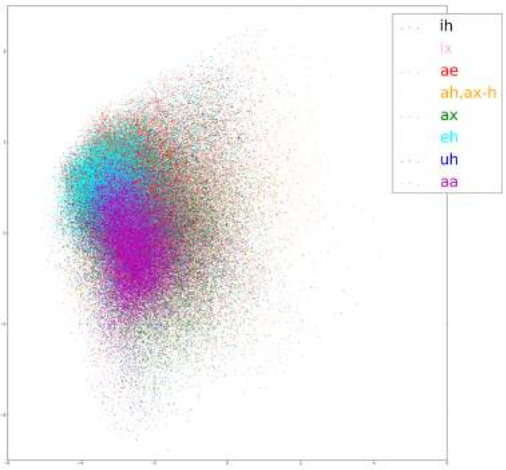
(a) Strong Fricative



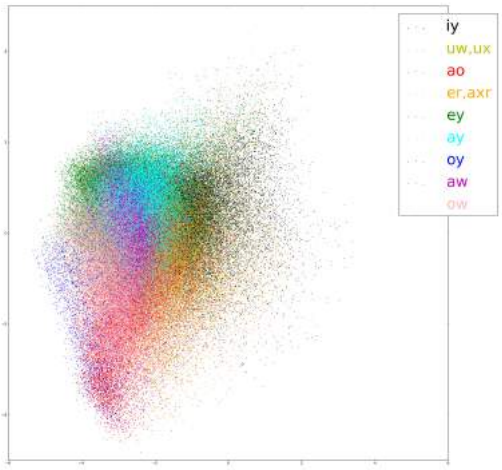
(b) Nasal, Flap



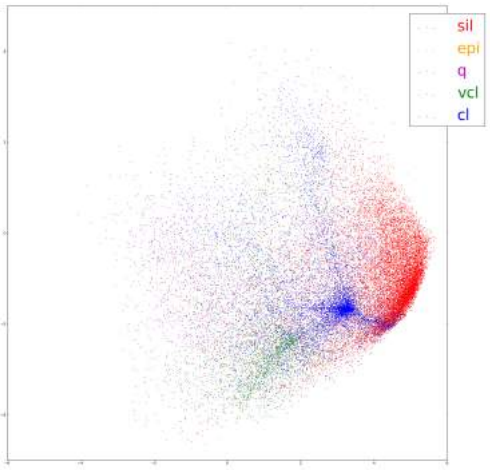
(c) Semi-vowel



(d) Short Vowel



(e) Long Vowel



(f) Silence (*)

Figure 5.2 (cont.): *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of 9-dimensional BNFs from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure. (*)For silence only 10% of the data are plotted or clarity. Horizontal axis: the 1^{st} dimension of LDA projections; vertical axis: the 2^{nd} dimension of LDA projections.*

The visualisations of the 3^{rd} and the 4^{th} dimension of the LDA-based projections are included in Appendix A.

5.2.2 Visualisation of BNFs with t-SNE

In the previous section, we visualised the 9-dimensional BNFs with LDA. In this section, we visualise the 9-dimensional BNFs with another technique - t-SNE. The t-SNE algorithm is described in Section 2.4.3. Note that LDA applies a linear mapping and is trained with class labels (i.e. supervised training), whereas t-SNE applies unsupervised training and non-linearly maps high-dimensional data into a 2-dimensional or 3-dimensional space.

Figure 5.3 shows 2-dimensional t-SNE visualisations of BNFs obtained from the same network as in Section 5.2.1. The t-SNE experiment was performed on the 10% of TIMIT training data. The main parameters for t-SNE training, perplexity and training iterations were set to 50 and 2000, respectively. These were chosen empirically after exploring suitable values. Note that the t-SNE is an unsupervised learning technique, and the labels were only used when colouring the 2-dimensional vectors after the t-SNE dimensionality reduction process. The definition of phone categories and colours are the same as used in Section 5.2.1.

We can see from Figure 5.3 that with an unsupervised dimensionality reduction by t-SNE, the 2-dimensional visualisations of BNFs show fairly clear separations between phone categories. Similar to what we observed in Figure 5.1, there tends to be confusions between the vowel subcategories (semi-vowels, short vowels and long vowels), and plosive are sometimes mixed with weak fricatives.

The detailed plots for individual phones in each category are shown in Figure 5.4. They are subsets of the points shown in Figure 5.3. We can see that most data for phones within a category is quite strongly clustered. The visualisations of BNF features using t-SNE provide clearer separations between phones in each phone categories than using LDA.

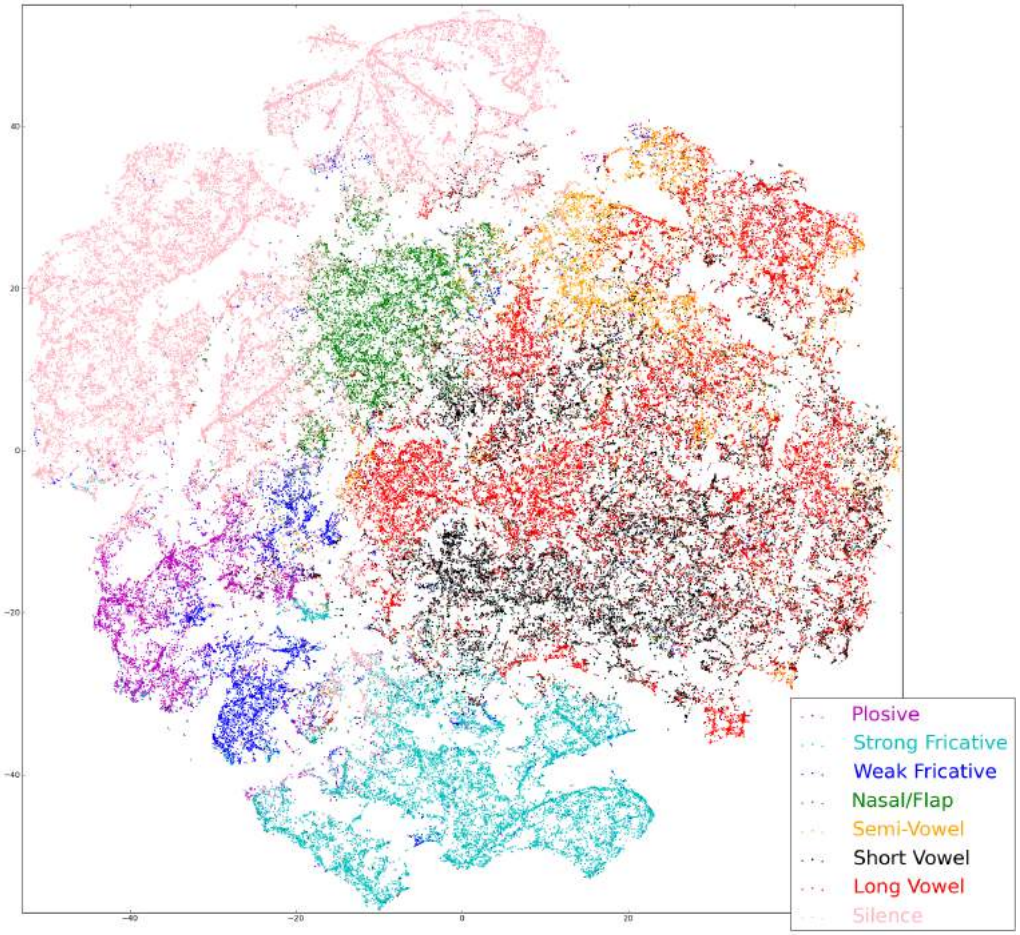
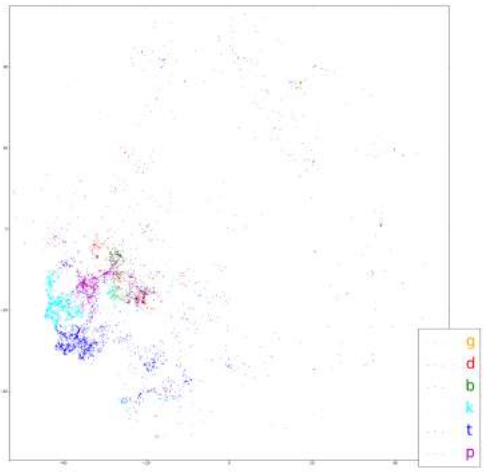
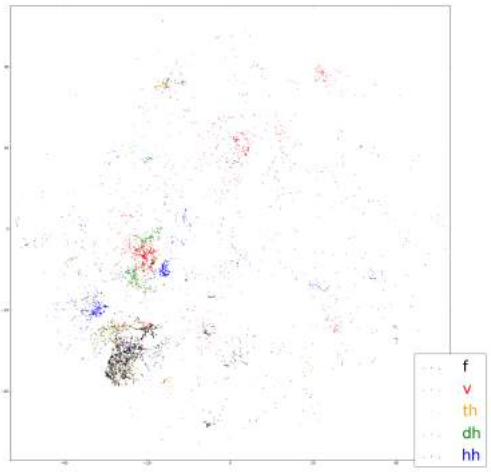


Figure 5.3: *2-dimensional t-SNE visualisations of 9-dimensional BNFs from a phone classification DNN of structure 286-512-9-512-49.*

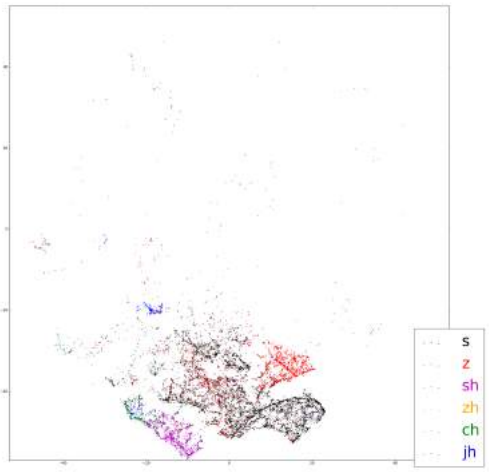
Note that the sizes of clusters and distances between them, in a t-SNE plot, cannot be directly related to the size or importance of clusters in the original high-dimensional data (Wattenberg et al., 2016). The t-SNE algorithm tries to learn local structures and map similar data close together in the low-dimensional space, while separating non-similar data.



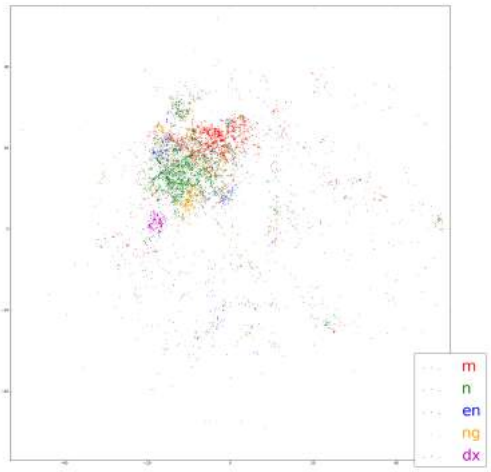
(a) Plosive



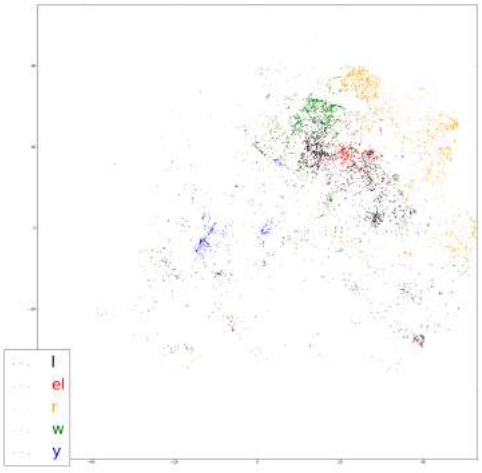
(b) Weak Fricative



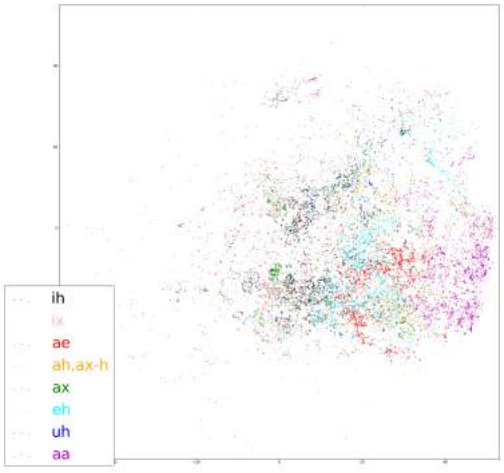
(c) Strong Fricative



(d) Nasal, Flap

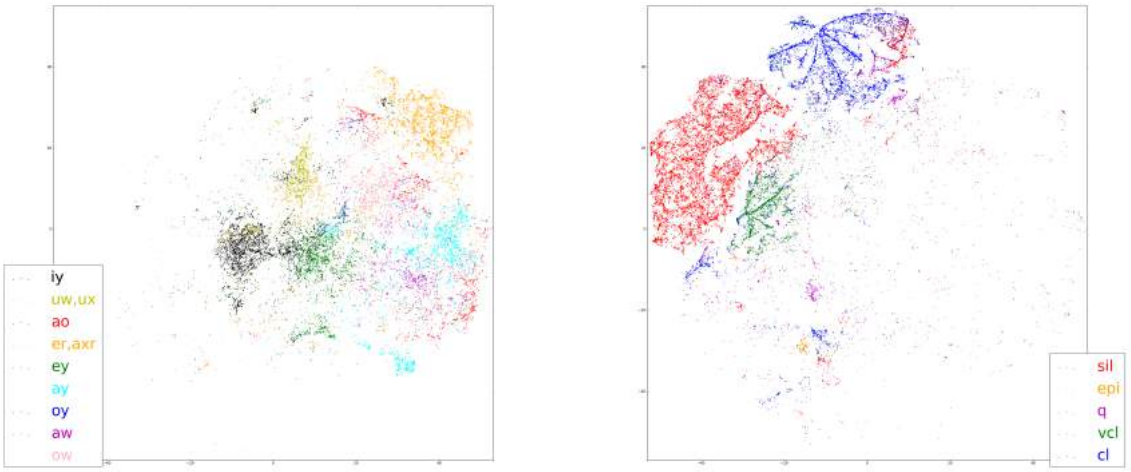


(e) Semi-vowel



(f) Short Vowel

Figure 5.4: *2-dimensional t-SNE visualisations of 9-dimensional BNFs (10% of the TIMIT training set) from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure.*



(a) Long Vowel (b) Silence

Figure 5.4 (cont.): *2-dimensional t-SNE visualisations of 9-dimensional BNFs (10% of the TIMIT training set) from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure.*

5.2.3 2-dimensional BNFs

In the previous subsections we mapped the 9-dimensional BNFs into a 2-dimensional space to try to visualise the BNFs. Since a bottleneck neural network itself is an approach to dimensionality reduction and producing low-dimensional features, it would be interesting to visualise 2-dimensional BNFs. Thus a phone discrimination network of structure 286-512-2-512-49 was trained and 2-dimensional BNFs were extracted. To keep consistency with the plots in the previous sections, when plotting all phones we use the same 10% of the TIMIT training set, and the plot of these 2-dimensional BNFs is shown in Figure 5.5. The definition of phone categories and colours are the same as used in Section 5.2.1 and 5.2.2.

From Figure 5.5 we can see fairly clear organisations of phone categories: vowels (red, black, and orange) are distributed at the left top half, nasals (green) at the right top corner, strong fricatives (cyan) at the lower left, plosive (purple) somewhere at lower middle, some weak fricatives (blue) mixing up with plosive and some at the mid lower edge, and silence takes the right lower part of the figure.

The BNFs are constrained within the range of [0,1], due to the "squashing effect" of

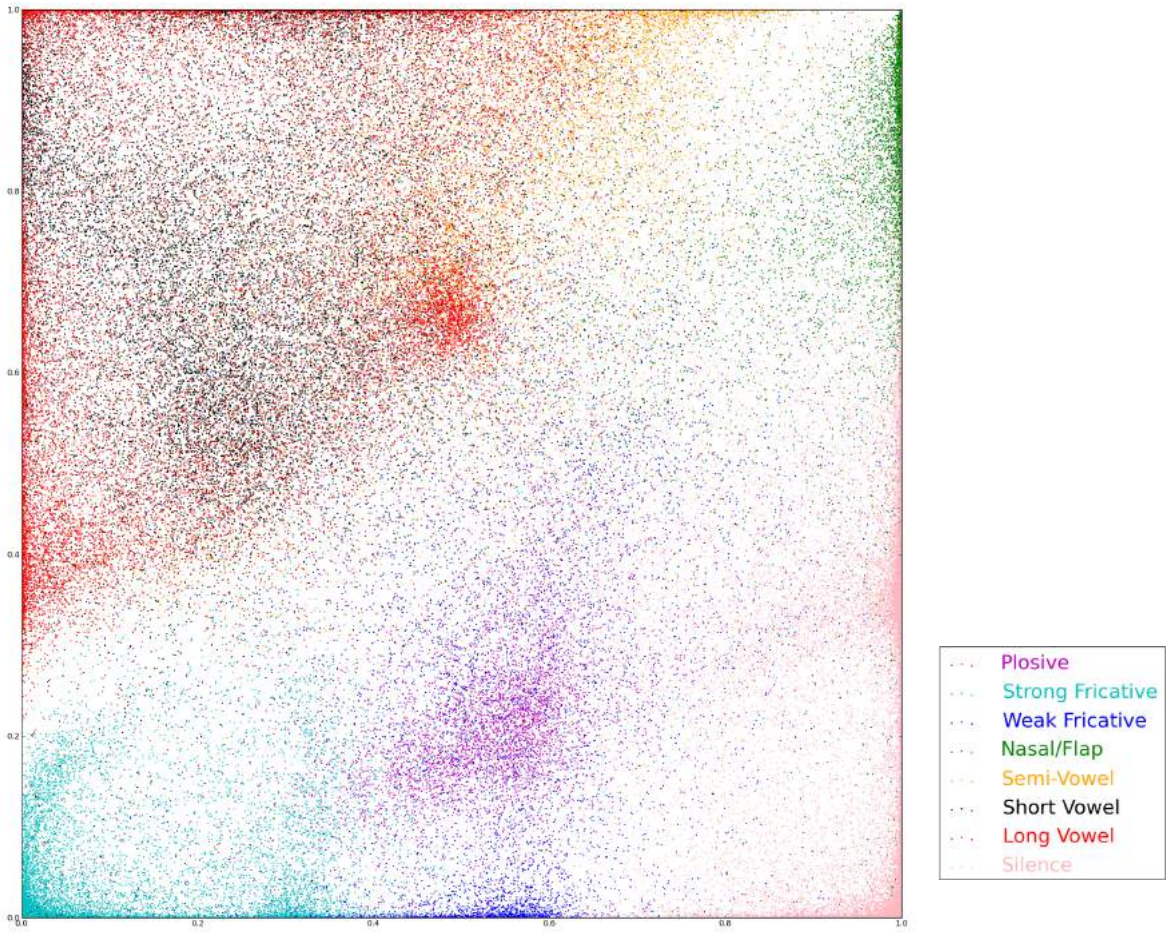
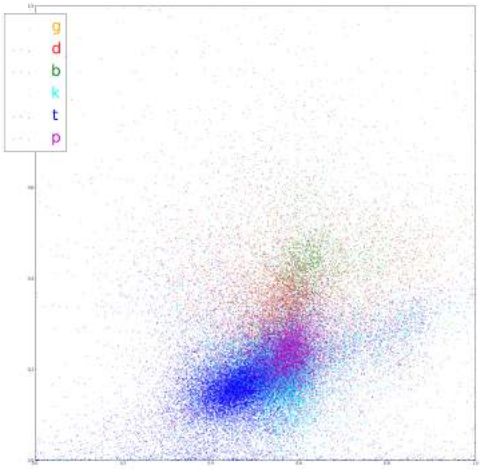


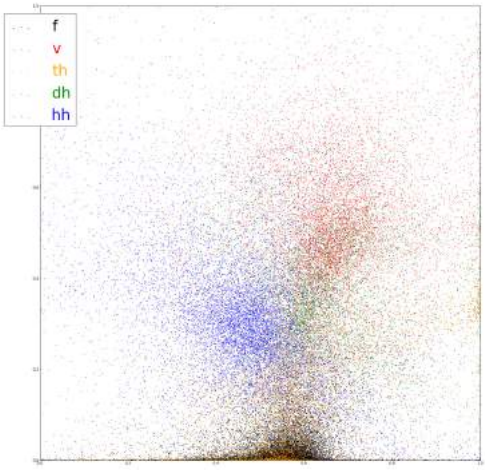
Figure 5.5: *2-dimensional BNFs from a phone classification DNN of structure 286-512-2-512-49.*

the sigmoid function. "Concentrated" edges along the four sides of the square appear to indicate "hard" or "certain" decisions made by the sigmoid for those BNFs. Moreover, the direction from top-left to bottom-right seems to indicate voicing.

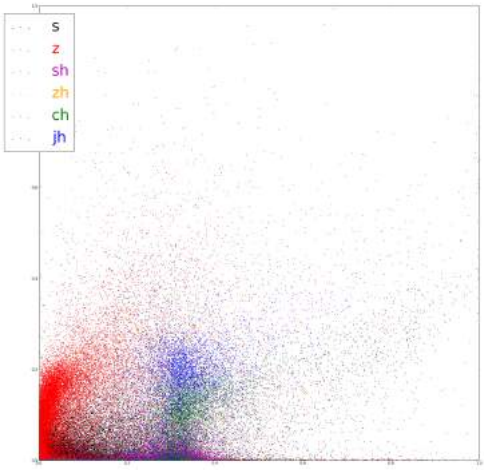
We plot individual phones in each phone category in Figure 5.6. Again for silence we use the same subset of TIMIT data as was used in Section 5.2.1, i.e. 10% of the TIMIT training set for silence and full training data for the others.



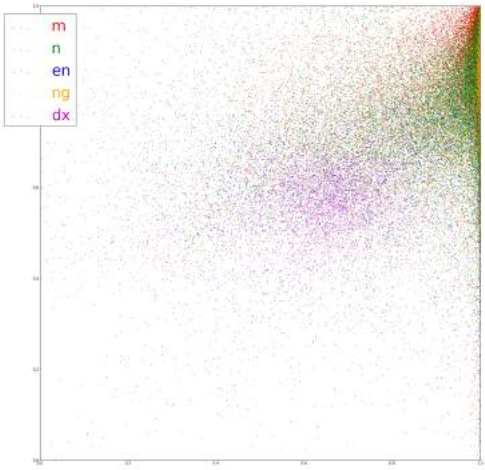
(a) Plosive



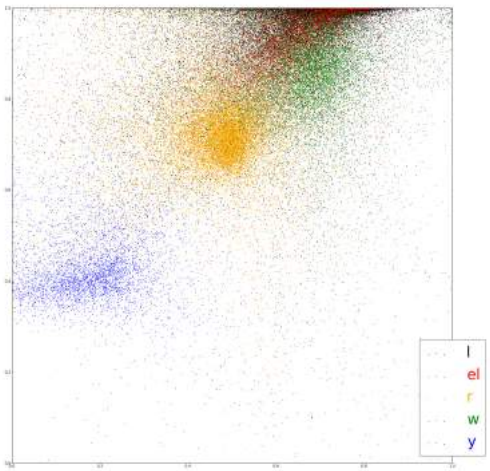
(b) Weak Fricative



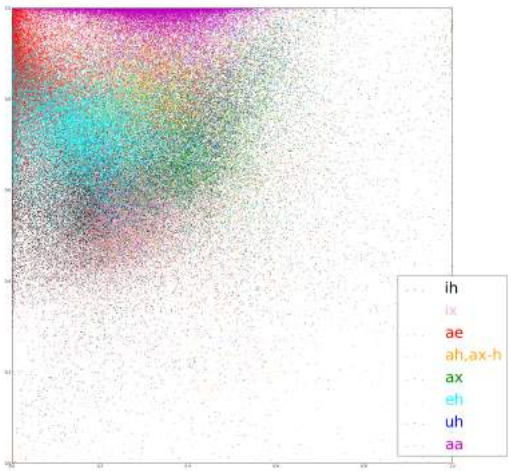
(c) Strong Fricative



(d) Nasal, Flap

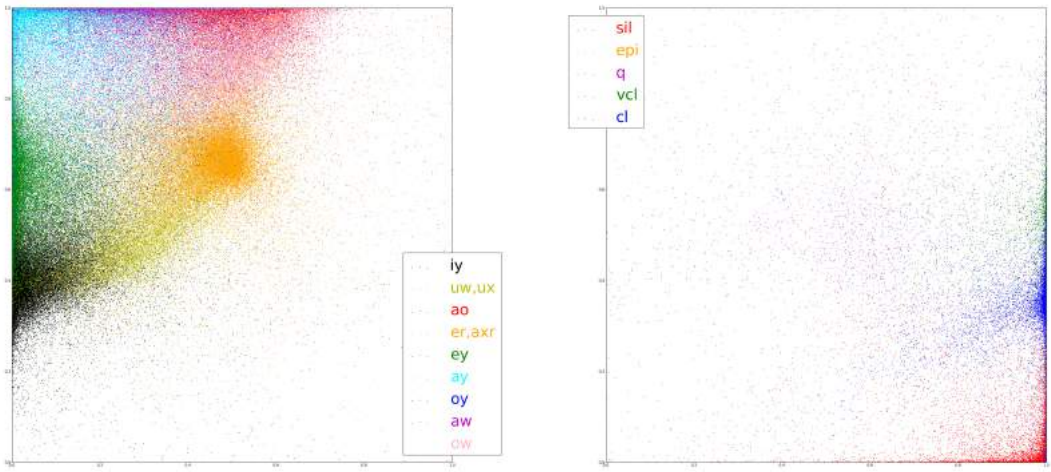


(e) Semi-vowel



(f) Short Vowel

Figure 5.6: *2-dimensional BNFs from a phone classification DNN of structure 286-512-2-512-49. Plot on one phone category in each figure.*



(a) Long Vowel (b) Silence (*)

Figure 5.6 (cont.): *2-dimensional BNFs from a phone classification DNN of structure 286-512-2-512-49. Plot on one phone category in each figure. (*)For silence only 10% of the data are plotted.*

The 2-dimensional BNFs are visually more structured than the 2-dimensional LDA projection of the 9-dimensional BNFs, especially for individual phones in named categories. We propose a way to obtain "representative" features in the following Section 5.3.1, where more detailed interpretation of the 2-dimensional BNFs will be made.

5.3 Exploring the neural network learning behaviour

5.3.1 Optimised neural activations

Neural networks for phone classification are usually trained to optimise the weights to maximise phone posterior probabilities given input of spectra-in-context, and learn an intermediate representation at each layer. We now ask, given a neural network, what pattern of activation in the hidden layers would be optimal to maximise the probability of the network predicting each phone? These activations would represent the "best" or "cardinal" phone representations under this network.

An approach using gradient descent back-propagation is proposed to obtain such optimised activations at a given layer. By keeping the network weights fixed and

calculating the derivatives of errors with respect to the layer activations, the layer activations can be adapted.

The process works as follows: Assume a trained l-layer neural network with $l-2$ hidden layers between the input and output layers. Let L_m denote the layer m after the input ($m \geq 0$), such that L_0 is the input layer, L_{l-1} is the output layer. Let W_m and b_m be the weight matrix and bias vector respectively, between layer L_m and L_{m+1}. Take the case optimising the activations a_m at layer L_m to maximise the output probability of phone ϕ as an example. The process can be divided into a forward propagation process and a back-propagation process that run alternately.

Forward Propagation process: We first initialise a_m, for example, with random samples from a uniform distribution $U(0,1)$. Then at the next layer, the linear output is given by

$$\mathrm{o}_{m+1} = \mathrm{a}_m W_m + b_m, \tag{5.1}$$

to which a squashing function is applied:

- If $m = l-2$:

  This means the next layer is the network output layer L_{l-1}

  $$\mathrm{o}_{l-1} = \mathrm{a}_{l-2} W_{l-2} + b_{l-2}. \tag{5.2}$$

  The Softmax function is then applied for an element-wise normalisation to the phone posterior probabilities

  $$\mathrm{a}_{l-1} = Softmax(\mathrm{o}_{l-1}). \tag{5.3}$$

  The cross-entropy error criterion to minimise is:

  $$C = -\sum (t \ln(\mathrm{a}_{l-1}) + (\mathbf{1} - t)\ln(\mathbf{1} - \mathrm{a}_{l-1})), \tag{5.4}$$

summing over the network outputs. t is the one-hot target vector in which the target phone has probability 1. $\mathbf{1}$ is a vector of ones of the same dimension.

- If $m < l - 2$:

  L_{m+1} is a hidden layer. The sigmoid function is applied element-wise

$$\mathrm{a}_{m+1} = \frac{1}{1 + e^{-\mathrm{o}_{m+1}}}, \tag{5.5}$$

  and forward the activations to the next layer.

Back-propagation process: From Equations 5.3 and 5.4, we derive the change in error with respect to th linear (pre-squashing) network outputs. Differentiation and division act element-wise.

$$\begin{aligned} \frac{\partial C}{\partial \mathrm{o}_{l-1}} &= \frac{\partial C}{\partial \mathrm{a}_{l-1}} \frac{\partial \mathrm{a}_{l-1}}{\partial \mathrm{o}_{l-1}} \\ &= -\Big(\frac{t - \mathrm{a}_{l-1}}{\mathrm{a}_{l-1}(\mathbf{1} - \mathrm{a}_{l-1})}\Big)\mathrm{a}_{l-1}(\mathbf{1} - \mathrm{a}_{l-1}) \\ &= \mathrm{a}_{l-1} - t. \end{aligned} \tag{5.6}$$

From Equations 5.2 and 5.6, the change in the error with respect to the activations at the final hidden layer,

$$\begin{aligned} \frac{\partial C}{\partial \mathrm{a}_{l-2}} &= \frac{\partial C}{\partial \mathrm{o}_{l-1}} \frac{\partial \mathrm{o}_{l-1}}{\partial \mathrm{a}_{l-2}} \\ &= (\mathrm{a}_{l-1} - t)W_{l-2}. \end{aligned} \tag{5.7}$$

Then for $m < l - 2$, relate the activations to the layer backwards,

$$\begin{aligned} \frac{\partial \mathrm{a}_{m+1}}{\partial \mathrm{a}_m} &= \frac{\partial \mathrm{a}_{m+1}}{\partial \mathrm{o}_{m+1}} \frac{\partial \mathrm{o}_{m+1}}{\partial \mathrm{a}_m} \\ &= \mathrm{a}_{m+1}(\mathbf{1} - \mathrm{a}_{m+1})W_m. \end{aligned} \tag{5.8}$$

Finally, the change in the error with respect to the activations at arbitrary layer L_m, can then be calculated:

$$\frac{\partial C}{\partial \mathrm{a}_m} = \frac{\partial C}{\partial \mathrm{a}_{l-2}} \frac{\partial \mathrm{a}_{l-2}}{\partial \mathrm{a}_{l-3}} \cdots \frac{\partial \mathrm{a}_{m+1}}{\partial \mathrm{a}_m}. \tag{5.9}$$

A gradient descent algorithm such as

$$\Delta \mathrm{a}_m^{(t)} = -\eta \big(\frac{\partial C}{\partial \mathrm{a}_m} \big)^{(t)} + \nu \Delta \mathrm{a}_m^{(t-1)}, \tag{5.10}$$

with learning rate η and momentum parameter ν, is used to update the layer activations following

$$\mathrm{a}_m^{(t)} = \mathrm{a}_m^{(t-1)} + \Delta \mathrm{a}_m^{(t)}, \tag{5.11}$$

where the (t) index indicates the values at training iteration t. The forward and backward processes run alternately until the cross entropy error no longer reduces or the epochs reach the maximum. Optimised hidden layer activations for each phone can be obtained using this process.

In our experiments, when calculating the "cardinal" bottleneck layer activations, i.e. BNFs, we applied the back-propagation process twice. First we back-propagate to the input layer as a pre-training process (with the maximum epoch being 1000), and then use the bottleneck layer activations resulting from this pre-training as the start point, to apply back-propagation to the bottleneck layer (with the maximum epoch being 100), which can be regarded as fine tuning.

Using this method, we obtain 49 2-dimensional BNF vectors representing the 49 phones for the DNN used in Section 5.2.3. We plot them in Figure 5.7 using dots. We also plot the centroids of the 2-dimensional BNFs (i.e. feature means) of each phone in the training set in Figure 5.7 using circles. In the figure we link every pair of dot and circle points that correspond to the same phone for a clearer view.

The circle points in Figure 5.7 actually present the means of the BNFs plotted in Figure 5.5, and therefore show similar organisations of phone positions to Figure 5.5. Most dot points are close to the corresponding circle points, and the organisations of circle points and that of dot points are similar. The direction from top left to bottom right seems to indicate voicing, with vowels distributing at the top left half of the graph, and silences being at the right bottom corner. Compared to the feature means

(circle points), for the "cardinal" features (dot points) the various categories seem to be pushed to the edges of a local space. The reason may be that "cardinal" features is trained to provide more certain phone decisions than random BNFs, which forces the hidden layer to make harder decisions.

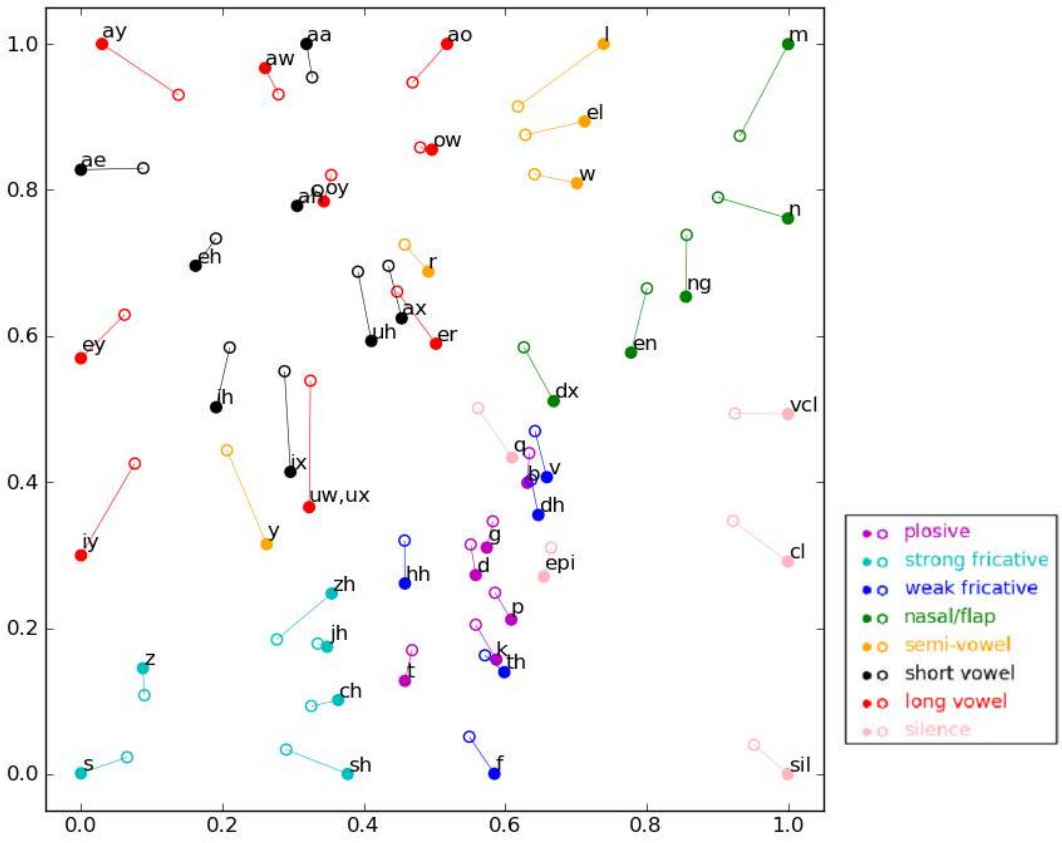


Figure 5.7: *Optimised 2-dimensional BNFs (dots) and feature means of 2-dimensional BNFs (circles) for each phone for a phone classification DNN of structure 286-512-2-512-49.*

In Figure 5.8, we take only the set of "cardinal" features from Figure 5.7 (dot points) and analyse them.

We now look at long vowels and short vowels (shaded area) in Figure 5.8. It looks similar to a "traditional" F1:F2 vowel space diagram used by phoneticians (described in Section 2.3.1, Figure 2.8(a)), rotated. Vertically from top to bottom, we observe /ay/, /ey/, /iy/ (left side) and /ao/, /uh/, /uw/ (right side) - that roughly corresponds

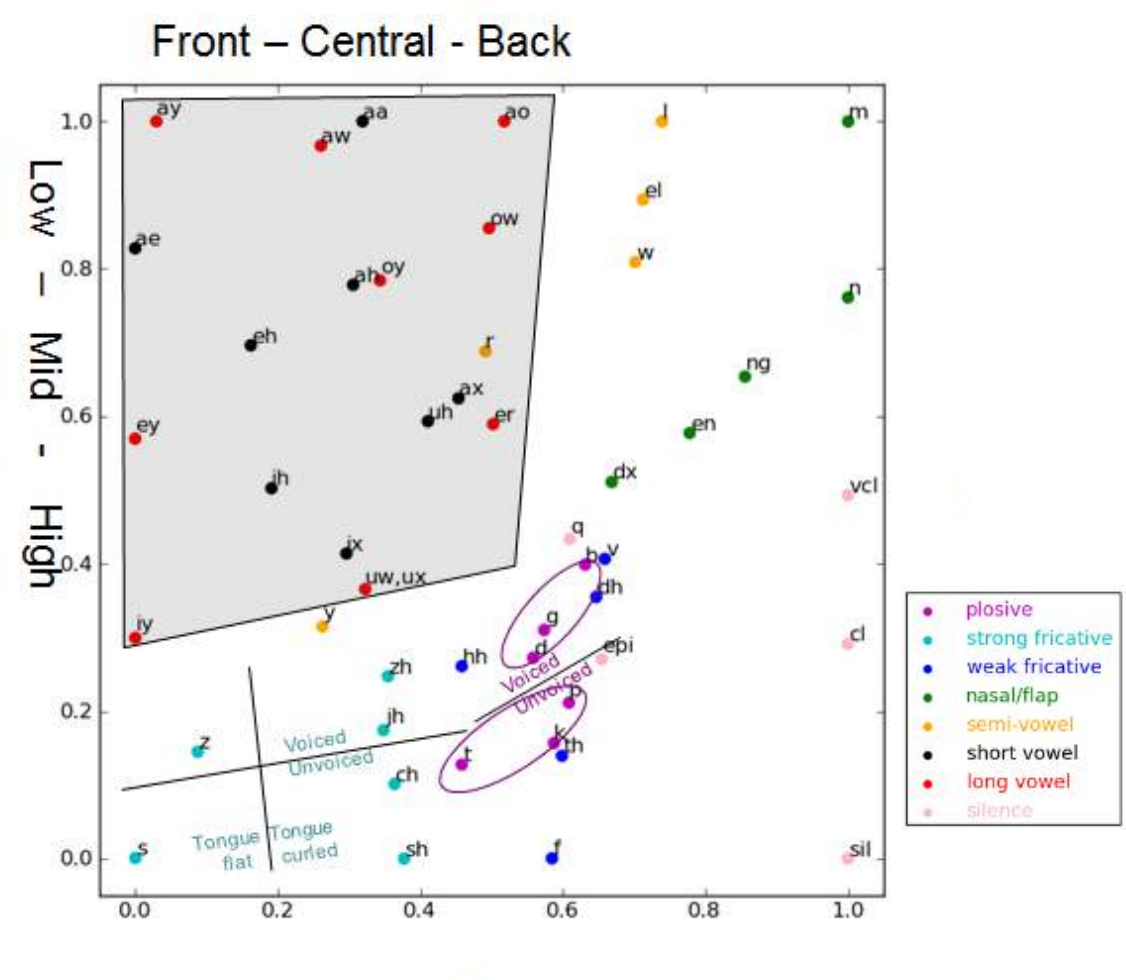


Figure 5.8: *Optimised 2-dimensional BNFs for each phone for a phone classification DNN of structure 286-512-2-512-49.*

to the places of articulations (tongue positions) from low to high; horizontally from left to right, we observe /ey/, /ah/, /ow/ - roughly front to back regarding the places of articulations. We plot the shaded area (long and short vowels) of Figure 5.8 on top of an x-ray tracing of a vocal tract to vividly present these interpretations of vowel BNFs (Figure 5.9)

Strong fricatives are in cyan. /s/ and /z/ are distributed at the left bottom corner, and they are produced with a flat tongue. /zh/, /jh/, /ch/ and /sh/ are distributed in mid-lower region, and they are produced with the tip of the tongue curled up. Strong fricatives at the top (/z/, /zh/, /jh/) are voiced, and those at the bottom (/s/, /sh/,

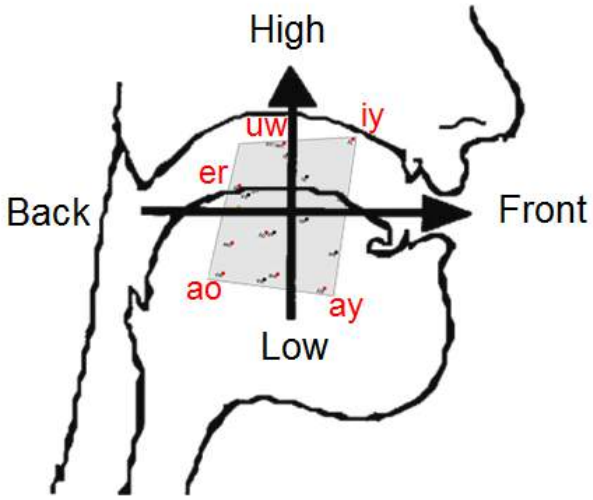


Figure 5.9: *Overlaying the shaded area of Figure 5.8 on an x-ray tracing of a vocal track.*

/ch/) are unvoiced. A similar pattern is seen for the plosives (in purple)- voiced at the top (/d/, /g/, /b/) and unvoiced at the bottom (/t/, /k/, /p/). For both voiced and unvoiced plosives, phones are placed horizontally in an order that reflects their place of articulation (from left to right: teeth, soft palate and lips).

The 2-dimensional BNF space shows distinct regions used for each phonetic category. Within each category, the organisation of phones appears to correspond to phone production mechanisms. However, the interpretations of axes of one phone category do not simply apply to other categories, and the BNF space seems to be a union of phonetic category related subspaces that preserve local structures within each subspace.

5.3.2 Neural network neuron responses at the bottleneck layer

In this subsection we analyse the activations in trained networks to show that different parts of the network are involved in predicting different phonetic categories. We explore the neural network neuron responses[2] at the bottleneck layer.

We measure magnitude of average node responses (z-score) of the neurons in the bottleneck layer to all phones. The z-score, also known as standard scores, of a raw

[2]Similar method was used in the work presented in (Nagamine et al., 2015)

score x is defined as (Kreyszig, 2000)

$$z = \frac{x - \mu}{\delta}, \tag{5.12}$$

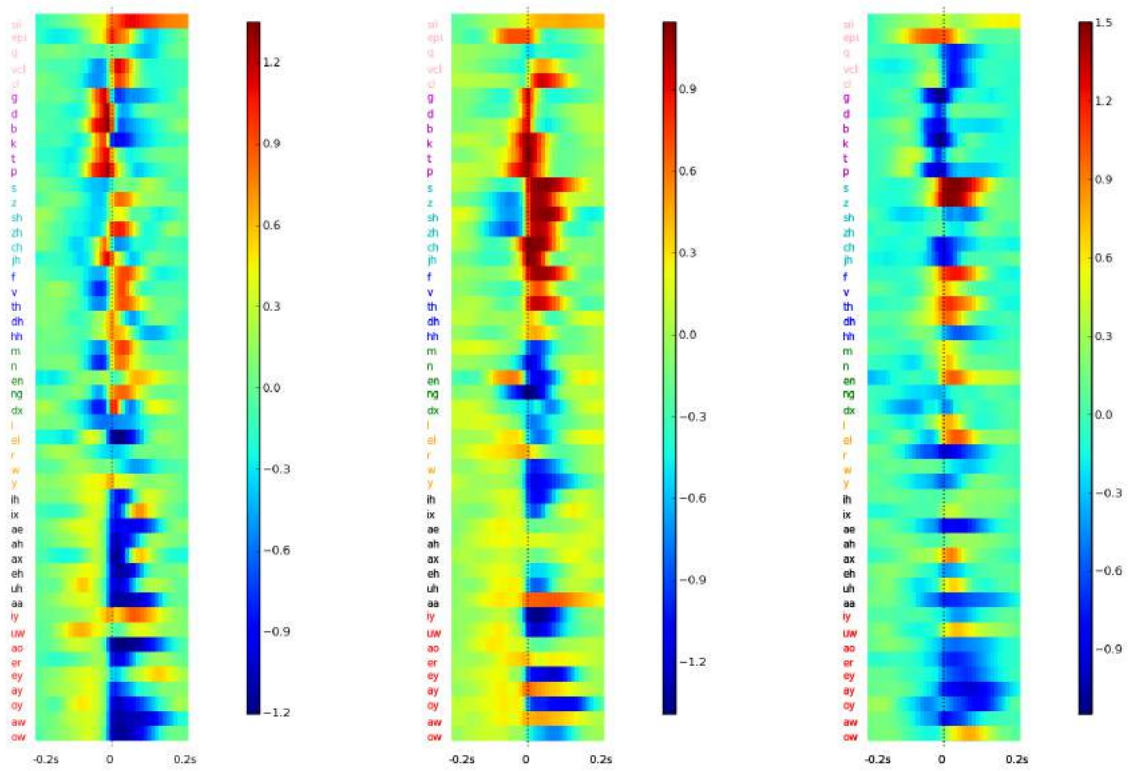
where μ is the mean of the population and δ is the standard deviation. Let the z-score for phone ϕ and neuron b be denoted $Z(\phi, b)$, calculated as follows: The frame-level activation at neuron b is recorded for each qualified frame (for all phones) in the TIMIT training set. Temporally we can assume the qualified "frames" to be the "start frames" of phone segment according to the TIMIT labelling. These approximately 143,000 activations are normalised to zero mean and unit variance across the whole training set. $Z(\phi, b)$ is the mean of the normalised activations at neuron b which are associated by the TIMIT labelling with instances of ϕ. For each phone we calculate an averaged and normalised response ("z-score") from each neuron in the 9-neuron bottleneck layer.

This is first done for the 286 dimension spectra-in-context network input associated with the start frames indicated by the TIMIT phone transcriptions, and then repeated for 20 preceding and succeeding frames, allowing the response to be visualised over about 0.4 seconds around the phone start boundary. Figure 5.10 shows the z-score plots for each bottleneck node (9 in total) from a DNN of structure 286-512-9-512-49. For plots of each node, we arrange the phones by the phone classes they belong to, annotating them on the left of each z-score plot with different colours. The colours are the same as used in Section 5.2. Form top to bottom, the broad phone classes are silence (light pink), plosive (purple), strong fricative (cyan), weak fricative (blue), nasal (green), semi-vowel (orange), short vowel (black) and long vowel (red).

Figure 5.10 again suggests that the neurons are responsive to phones in a broadly phonetically-meaningful manner. The structures of excitations are complicated, usually covering several phone classes, but we can still identify particular patterns exclusively to related phones. For example, node 1 negatively responds to plosives and most

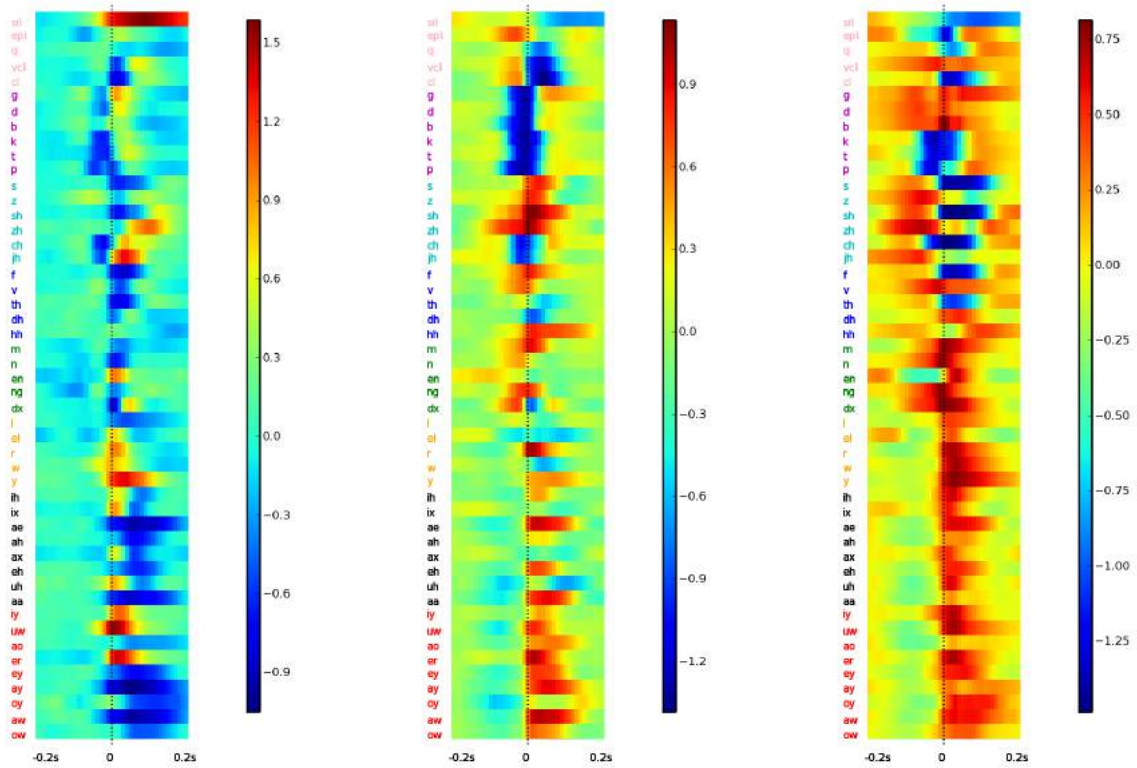
vowels; node 2 positively responds to fricatives, closures and unvoiced plosives (/k/, /t/, /p/), and negatively responds to nasal; node 5 negatively responds to all plosives; node 6 distinguishes between voiced and unvoiced phones; node 7 negatively responds to all strong fricatives; node 8 positively responds to closures and strong fricatives and negatively responds to plosive and most vowels; node 9 positively responds to unvoiced plosives and strong fricatives.

The z-score patterns are also observed to be similar for networks trained from different random initialisations, in a sense that each node is sensitive for sets of related phones. The results suggest that mappings from acoustic space to BNF space for different classes of phone are implemented by separate parts of the network.

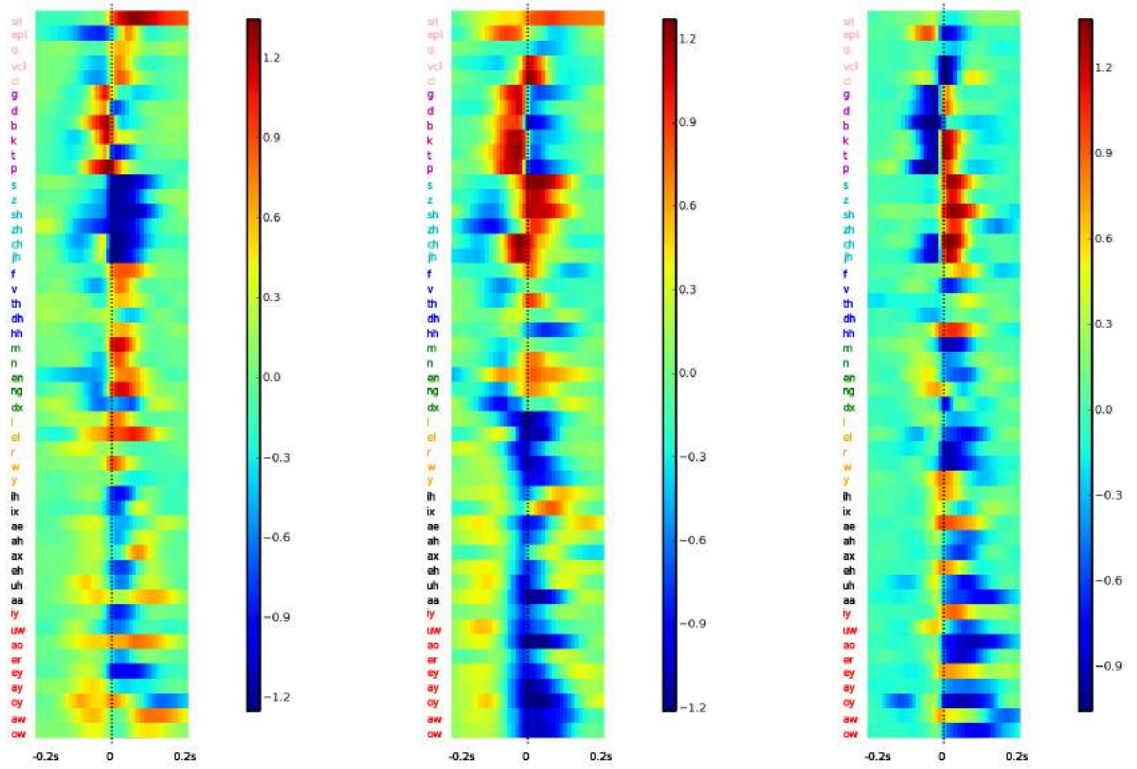


(a) Node 1 of 9 (b) Node 2 of 9 (c) Node 3 of 9

Figure 5.10: *Magnitude of average node activations (z-score), for each phone, over a 0.4s window centred on the phone onset (dotted vertical lines).*



(d) Node 4 of 9 (e) Node 5 of 9 (f) Node 6 of 9



(g) Node 7 of 9 (h) Node 8 of 9 (i) Node 9 of 9

Figure 5.10 (cont.): *Magnitude of average node activations (z-score), for each phone, over a 0.4s window centred on the phone onset (dotted vertical lines).*

5.3.3 Visualising non-bottleneck hidden layers with LDA

In this subsection we apply linear discriminant analysis (LDA) to non-bottleneck layers to investigate what main information are carried through these layers. We use the same DNN as used in Sections 5.2.1, 5.2.2 and 5.3.2. The DNN hidden layer size was 512-9-512, thus we now visualise the 1^{st} and the 2^{nd} dimensions of the LDA projection for the two 512-node layers. The LDA projection was learned on a random 10% of the frames in the TIMIT training set. The plots in this subsection use the same subset.

Figure 5.11 shows the 1^{st} and the 2^{nd} dimension of the LDA-based projections of the activations of the first hidden layer, plotted on the same 10% of the training set as in Section 5.2.1. The same process is applied to the 3rd hidden layer, giving Figure 5.12.

Both figures show a clear "triangular" shape with similar structures, where vowels, strong fricatives and silences each occupy a corner of the triangle. Along the horizontal axis, from left to right, we see silence and fricatives first and then vowels, which could be interpreted as from unvoiced to voiced, or energy in low frequency bands increasing; Along the vertical axis, from upper to lower, we see silence first, and then vowels, finally fricatives - this could be interpreted as energy in high frequency bands increasing. As the horizontal and vertical axis correspond to the first two dimensions of LDA, such interpretations may indicate that energies in low and high frequency bands are two main pieces of information learned by the DNN. Another interpretation of this triangular shape could be that there is some inherent 3 dimensionality structure in the high dimensional data, corresponding to 3 properties of phones: silence, frication and voicing.

Comparing Figure 5.11 and 5.12, we can see the triangular plot of the third hidden layer is much less fuzzy than that of the first hidden layer. One explanation is that as hidden layer moves towards the output layer, it gets forced to make harder decisions towards the phone classification goal. Specifically, in the 512-dimensional hypercube representing the activations of the third hidden layer, the 512-dimensional features are pushed towards the edging limit of "0"s and "1"s which is constrained by sigmoid func-

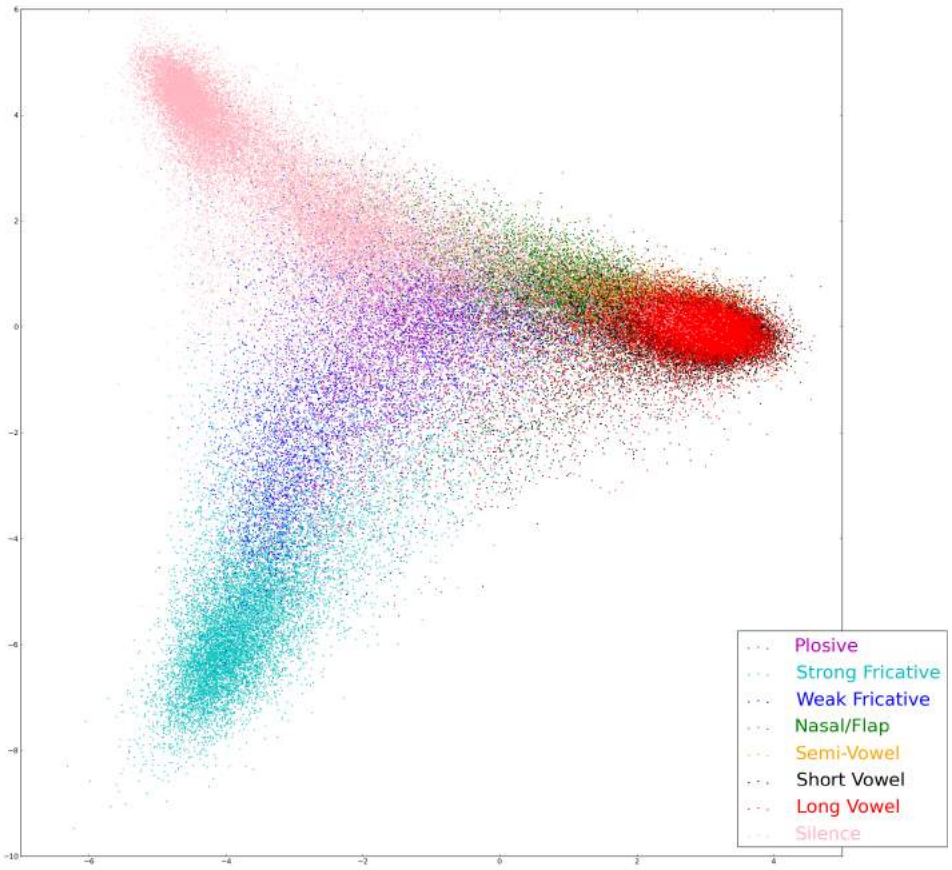


Figure 5.11: *Visualisation of LDA-based projections (1^{st} vs. 2^{nd} dimension) of the 1^{st} hidden layer activations from a phone classification DNN of structure 286-512-9-512-49. Horizontal axis: the 1^{st} dimension of LDA projections; vertical axis: the 2^{nd} dimension of LDA projections.*

tion. As a result, the 2-dimensional LDA projection of the "more confident activations" would be less fuzzy.

Plots of individual phones for each category are included in Appendix A, along with visualisations of the 3^{rd} and the 4^{th} LDA-based projections for the first and the third hidden layers.

We also find that this triangular visualisation of the 1^{st} and the 2^{nd} dimension of the LDA-based projections is always observed when analysing a "bigger" hidden layer (more than about 30 nodes) from a DNN of a similar structure.

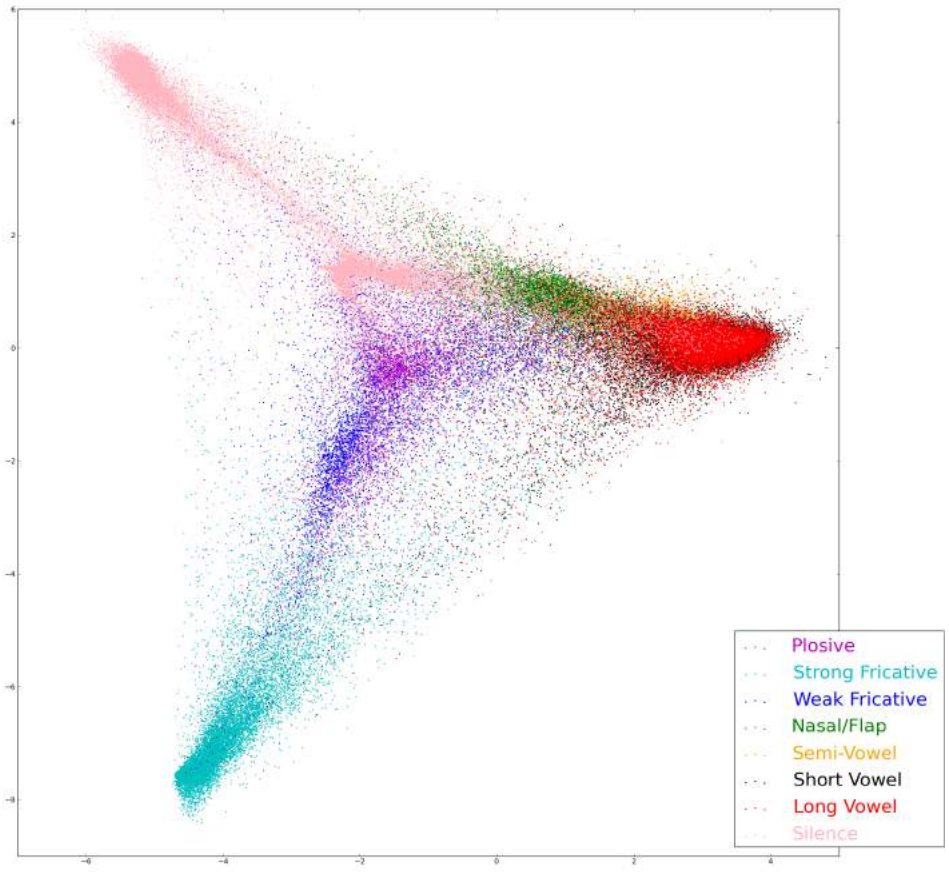


Figure 5.12: *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of the 3^{rd} hidden layer activations from a phone classification DNN of structure 286-512-9-512-49. Horizontal axis: the 1^{st} dimension of LDA projections; vertical axis: the 2^{nd} dimension of LDA projections.*

5.4 Summary and discussion

This chapter explores visualisations and interpretations of BNFs and the neural network learning behaviour. We started from visualising the 9-dimensional BNFs using LDA and t-SNE, where we observed phonetically meaningful clusters in the projected 2-dimensional spaces. We narrowed the bottleneck layer to 2-unit and extracted 2-dimensional BNFs. A back-propagation method was developed to obtain BNFs that are optimal under a particular neural network, by computing the values of the bottleneck outputs that are optimised for a particular output target. Using such method,

we obtained one "cardinal" 2-dimensional BNF for each phone and interpreted these BNFs. The 2-dimensional BNF space shows distinct regions used for each phonetic category, where within each category the organisations of phones appear to correspond to phone production mechanisms. We also investigated the neural network neuron responses at the bottleneck layer and visualisations of non-bottleneck layer activations. We demonstrate that different parts of the network are involved in capturing different phonetic information, or information for different phonetic categories.

Chapter 6

Relationships Between Bottleneck Features From Networks with Different Initialisations

6.1 Introduction

Both network parameters and BNFs would be different when training with a different random initialisation. It is important to ensure the same conclusions can be drawn from networks trained from multiple random initialisations. Therefore it is interesting to ask if there are any relationships between differently initialised neural network pairs. If such relationships exist, then interpretations from one network can be applied onto other ones trained from any random initialisations. Most neural networks are of huge size and it is difficult to interpret the differences or similarities between them, whereas our BNFs are of very low dimensions and it is more convenient to explore relationships between BNFs from differently initialised neural networks.

In this chapter, the relationships between BNFs obtained from different initialisations of the same network are analysed. Experiments are performed on the 9-dimensional BNFs obtained as described in Chapter 4. It is shown that the resulting

sets of BNFs are different, but that they give similar phone recognition performance (Section 6.2), and that the relationship between them is not simply linear but approximately piecewise linear (Section 6.3 and 6.4). Through a hierarchical clustering of phone-dependent linear transformations, it is shown that the piecewise linear components approximately correspond to intuitively meaningful phonetic categories. In addition, the biggest decreases in phone recognition accuracy occur when transforms corresponding to categories that differ significantly in their phonetic properties are combined. This result suggests that the network is able to learn and combine multiple phone category dependent feature extraction mappings to optimise a low-dimensional representation for its phone classification task (Section 6.5). This anticipates the consideration of models motivated by topological manifolds in Chapter 7.

6.2 Effect of NN initialisation on BNFs

In this section, the effect of different initialisations on the resulting BNFs is explored. Networks of the structure NN286-512-9-512-49 are used, with three hidden layers where the 9-neuron bottleneck layer is the second hidden layer (as described in Chapter 4). The networks were trained with a standard MLP stochastic gradient descent back-propagation algorithm (descried in Section 2.2.2).

6.2.1 Are BNFs corresponding to different weight initialisations the same?

Figure 6.1 depicts an example of two sets of 9-dimensional BNFs of the same speech utterance.

The BNFs are in the range [0, 1] but plotted shifted on the vertical axis for clarity. These feature sets were produced from networks with the same structure, but different random parameter initialisations. We show the correlation coefficients between the two BNFs in Table 6.1. Correlation coefficients whose absolute values are greater than 0.6

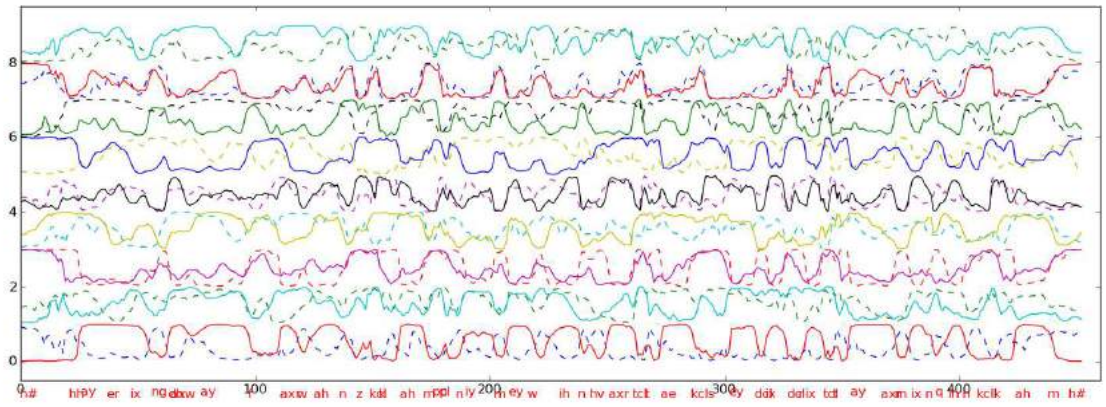


Figure 6.1: *Two sets of 9d BNFs for utterance TRAIN/DR2/MEFG0/SI491, from network 286-512-9-512-49 with two different initialisations.*

Table 6.1: Correlation coefficients between two sets of 9d BNFs for the TIMIT utterance TRAIN/DR2/MEFG0/SI491, from network 286-512-9-512-49 with two different initialisations (the two sets of BNFs are displayed in Figure 6.1). Rows and columns correspond to the solid and the dashed lines in Figure 6.1.

	1	2	3	4	5	6	7	8	9
1	-0.44	-0.05	**-0.93**	0.31	0.08	0.12	0.54	-0.23	0.40
2	-0.43	0.07	-0.07	0.11	**0.69**	0.42	-0.11	-0.09	-0.10
3	0.11	0.22	0.56	-0.12	-0.34	-0.31	**-0.68**	0.29	-0.37
4	-0.14	0.56	-0.11	0.13	-0.01	**-0.65**	0.41	-0.48	-0.18
5	**-0.62**	0.10	-0.33	0.28	0.43	-0.14	0.02	-0.45	0.53
6	-0.02	-0.22	**0.72**	-0.21	0.07	-0.56	-0.19	0.12	-0.09
7	0.16	0.16	0.35	0.05	**-0.62**	0.22	-0.14	0.39	0.12
8	0.27	-0.33	0.09	-0.39	-0.42	0.06	-0.35	**0.70**	-0.55
9	0.19	-0.31	-0.25	**0.69**	0.09	-0.03	0.44	-0.06	-0.24

are displayed in bold. The two sets are clearly different, and not simply permutations of each other. There is only one pair that is highly correlated (-0.93 in row 1 column 3), which correspond to the bottom red solid line and the 3rd dashed line (red) from the bottom.

There is no obvious intuitive interpretation of the features, beyond speculation that one (the bottom red solid line and the 3rd from the bottom red dashed line in the figure) may be related to voicing or overall energy of the signal. It is interesting that such observation is obtained in all BNF sets used in this chapter.

6.2.2 Do different BNF sets give similar recognition accuracy?

Different BNF sets correspond to different non-linear mappings between the input spectra-in-context and phone posterior probability targets. Do they give similar recognition accuracies? Four sets of BNFs are tested, where the structure of the network is the same in each case, but the random initialisation is different. Speech recognition experiments are performed using a standard GMM-HMM system created using HTK (Young et al., 1997), as described in Section 4.2.3. For evaluating recognition performance, the 49 phone set is reduced to 40 in the standard way (Lee and Hon, 1989). Table 6.2 shows ASR performance using four sets of BNFs obtained from four networks trained with different initialisations.

Table 6.2: *Phone recognition performance on the TIMIT core test set with four sets of BNFs obtained using different random network initialisations.*

NN structure	BNF set	Recognition Result		
		%Corr	%Acc	#GMM
286-512-9-512-49	*A*	73.12	69.40	512
	B	73.76	69.49	256
	C	72.24	68.89	128
	D	73.14	69.92	128

Table 6.2 suggests that the BNFs from multiple network initialisations give similar recognition accuracies in a standard GMM-HMM system, in other words, they are approximately equivalent in terms of ASR performance. It is natural to ask if any simple relationships exist between them. In the following sections, it is explored whether any simple relationships exist between them, beginning in Section 6.3 with the simplest, a linear relationship.

6.3 Linear mappings between BNFs extracted from networks with different initialisations

In this section, it is explored whether there is a linear relationship between the sets of BNFs obtained using different random initialisations.

Consider two sets of m-dimensional BNFs, A and B, of size N frames. A is divided into training and test sets A_{tr} and A_{te}, of N_{tr} and N_{te} frames respectively. B is divided likewise. We slightly abuse notation by using the same notation for the matrices containing the BNFs, i.e. A denotes the $m \times N$ matrix containing BNF set A. We say that these two sets are approximately linearly equivalent if there exists a $m \times m$ linear transform matrix mapping B to A,

$$T_{B \to A} : B \to A, \tag{6.1}$$

such that the phone recognition performance when testing using the transformed feature set $T_{B \to A}(B_{te})$ on models trained on the set A_{tr} is similar to using those trained on A_{tr} and tested on A_{te}. Figure 6.2 illustrates this process.

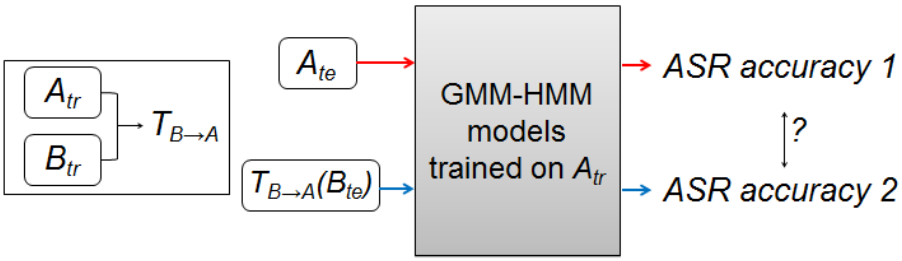


Figure 6.2: *An illustration of how we decide whether two BNF sets are linearly equivalent. If "ASR accuracy 1" and "ASR accuracy 2" are similar, we say there is an approximately linear relationship between the two sets.*

Set A_{tr} was chosen to be the fixed set used for training the GMM-HMM models. From each training set B_{tr}, C_{tr} and D_{tr}, a corresponding optimal linear transformation can be computed: $T_{B \to A}$, $T_{C \to A}$ and $T_{D \to A}$ by minimising the least square error between BNF set A_{tr} and the BNFs resulting from the transform, using the pseudo inverse method (Golub and Kahan, 1965). Phone recognition on the transformed features $T_{B \to A}(B_{te})$, $T_{C \to A}(C_{te})$ and $T_{D \to A}(D_{te})$ is performed using models trained on A_{tr}.

The recognition results obtained are shown in Table 6.3. A significant reduction of around 7% (absolute) is apparent between the phone recognition performance obtained with test features A_{te} from BNF set A (row 2) and with transformed features from BNF

sets B_{te}, C_{te} and D_{te} (rows 3-5), which indicates that information is lost by the mapping. We conclude that there is not a straightforward single linear mapping between the two BNF sets.

Table 6.3: *Recognition on TIMIT core test using transformed test features, models trained on BNF set A_{tr}. Baseline1 shows mean and standard deviation over the four BNF sets, matched train and test.*

Test Features (NN286-512-9-512-49)	%Corr	%Acc
Baseline1 (matched train and test, average)	73.1 (0.6)	69.5 (0.4)
Baseline2 (matched train and test, BNF set A)	73.12	69.40
B_{te} mapped towards A_{te}	64.49	60.66
C_{te} mapped towards A_{te}	65.54	61.56
D_{te} mapped towards A_{te}	64.01	60.06

6.4 Piecewise linear mappings between BNFs extracted from networks with different initialisations

Since there is no simple linear relationship between BNF sets, it is natural to ask if there is any phone-dependent linear relationships, i.e. a piecewise linear relationship between BNF sets. We begin by estimating three sets of 49 phone-dependent linear transformations between BNF set A and each of B, C and D. Linear transformations are computed in the same way as in Section 6.3, but estimate a transform for each phone. For example, for BNF sets A and B, the linear transform matrix mapping the features corresponding to phone ϕ in set B to that in set A is

$$T^{\phi}_{A\rightarrow B} : A_{\phi} \rightarrow B_{\phi}, \tag{6.2}$$

and the transform from set A to B is

$$T_{A\rightarrow B} = \{T^{\phi}_{A\rightarrow B} | \phi \in \Phi\}. \tag{6.3}$$

We use the same set Φ of 49 phones as was used for training the network.

However, in this case, because during testing we do not know the correct phone category of a feature vector, we transform the models rather than the data. Let $\mathcal{A} = \{\mathcal{A}_\phi | \phi \in \Phi\}$ denote the set of per-phone GMM-HMM models trained on BNF set A_{tr}, and similarly $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$. Let

$$S^{\phi}_{\mathcal{A}\to\mathcal{B}} : \mathcal{A}_\phi \to \mathcal{B}_\phi \tag{6.4}$$

define a linear transform mapping the parameters (GMM means and covariance matrices) of $\mathcal{A}_\phi$ to those of $\mathcal{B}_\phi$, using $T^{\phi}_{A\to B}$. We consider two sets of BNFs A and B to be approximately piecewise linearly equivalent if there exists a set of phone-dependent linear transforms such that the recognition accuracy is similar using models $\mathcal{A}$ trained on BNF set A_{tr} and tested on A_{te}, or transformed models obtained with the set of transforms $S_{\mathcal{A}\to\mathcal{B}}(\mathcal{A}) = \{S^{\phi}_{\mathcal{A}\to\mathcal{B}}(\mathcal{A}) | \phi \in \Phi\}$ to test on BNF set B_{te}. Figure 6.3 illustrates this process.

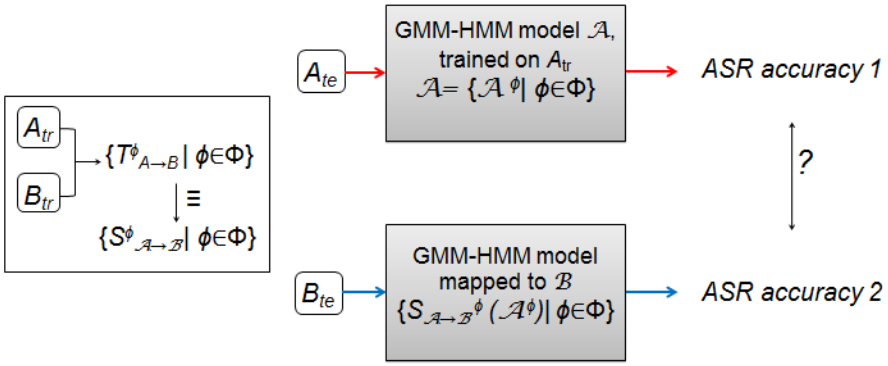


Figure 6.3: *An illustration of how we decide whether two BNF sets are piecewise linearly equivalent. If "ASR accuracy 1" and "ASR accuracy 2" are similar, we say there is an approximately piecewise, or phone-dependent linear relationship between the two sets.*

We may also apply the set of transforms to the BNFs themselves (rather than the models), if we know the phone segmentation, using Equation 6.3

$$B_{recovered} = T_{A\to B}(A) = \{T^{\phi}_{A\to B}(A_\phi) | \phi \in \Phi\}. \tag{6.5}$$

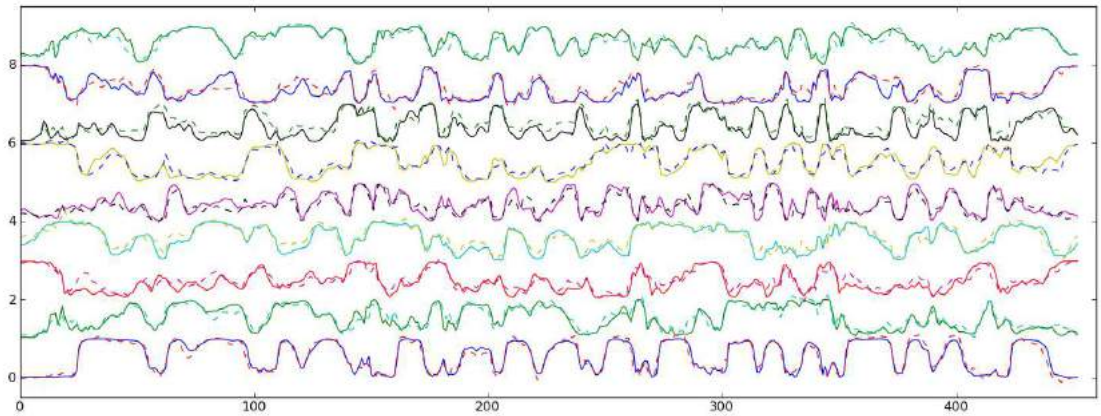


Figure 6.4: *Plot of 9-dimensional BNFs for the train utterance DR2/MEFG0/SI491. Solid lines are true BNFs of sett B, and dashed lines are transformed feature using transform* $T_{A \to B}$.

Figure 6.4 shows features from the transformed BNF set (dashed lines) for one utterance, against those from the original BNF set B (solid). The transformed features appear visually quite similar to the original features that are extracted directly from the neural network.

Table 6.4 shows the GMM-HMM recognition results using the transformed phone models. The reported results are on the core test set. The baseline results (line 1-3) are for models trained on A_{tr}, tested on A_{te}, for 1, 64, and 512 GMM mixture components. As the number of GMM components increases, the GMMs fit BNF A better and the recognition accuracy increases. In lines 4-6, phone-dependent linear transforms are used to map the parameters of the GMM-HMMs trained on BNF set A_{tr} to be appropriate for testing on BNF sets B_{te}, C_{te} and D_{te}.

Table 6.4: *GMM-HMM recognition on the TIMIT core test set with transformed models. HMMs are originally trained on BNF set A.*

Model Transformation	Test Set	%Corr	%Acc	#GMM
N/A (baseline)	BNF A_{te}	73.12	69.40	512
	BNF A_{te}	72.24	68.68	64
	BNF A_{te}	70.30	66.56	1
$\mathcal{A}$ mapped towards $\mathcal{B}$	BNF B_{te}	72.56	66.97	32
$\mathcal{A}$ mapped towards $\mathcal{C}$	BNF C_{te}	72.04	67.39	64
$\mathcal{A}$ mapped towards $\mathcal{D}$	BNF D_{te}	72.26	66.93	64

The results show the best accuracies and corresponding number of GMM compo-

nents, which is maximised with around 64 GMM components. This could be interpreted as meaning that when there are too many Gaussian mixtures, the models overfit BNF A and thus fit the other BNF sets less well after transformation. The difference in accuracies obtained using models $\mathcal{A}$ trained and evaluated on BNFs from set A, and models $S_{\mathcal{A}\to\mathcal{B}}(\mathcal{A})$, $S_{\mathcal{A}\to\mathcal{C}}(\mathcal{A})$, $S_{\mathcal{A}\to\mathcal{D}}(\mathcal{A})$ trained on A_{tr} and then mapped to and evaluated on, BNF sets B_{te}, C_{te} and D_{te} respectively, is between 1% and 2%. We conclude that the relationship between the BNF sets is approximately piecewise linear.

6.5 Hierarchical clustering for phone-dependent linear transformations

In the previous section, using phone-dependent linear transforms, we established that there is an approximately piecewise-linear relationship between the different BNF sets. In this section we investigate whether a mapping is needed for each phone, or whether the same linear transform can be used for models corresponding to phones belonging to broader phonetic classes.

We apply agglomerative hierarchical clustering to the 49 per-phone linear transforms $T_{A\to B} = \{T^{\phi}_{A\to B} | \phi \in \Phi\}$ estimated for later mapping GMM-HMM set $\mathcal{A}$ to $\mathcal{B}$. Let $\mathcal{K}_{A\to B} = \{\mathcal{K}^{(i)}_{A\to B} | 0 < i \leqslant N_K\}$ be the set of clustered piecewise-linear transforms from A to B, represented by the centroids of the clusters, which is calculated as the mean of the cluster elements (transform matrices). $\mathcal{K}_{A\to B}$ initially contains $N_K = 49$ elements (one transform for each phone in Φ). At each step we merge the two closest transforms, measured by the Euclidean distance d_E between features generated by the transforms. Let the distance between the two cluster $\mathcal{K}^{(i)}_{A\to B}$ and $\mathcal{K}^{(j)}_{A\to B}$ be

$$d_E(i,j) = \sqrt{\sum_{a=1}^{m}\sum_{b=1}^{N_{tr}} (\bar{\mathcal{K}}^{(i)}_{A\to B}(A) - \bar{\mathcal{K}}^{(j)}_{A\to B}(A))^2_{ab}}, \tag{6.6}$$

where $\bar{\mathcal{K}}^{(i)}_{A\to B}$ and $\bar{\mathcal{K}}^{(j)}_{A\to B}$ denote the transform matrices corresponding to clusters $\mathcal{K}^{(j)}_{A\to B}$

and $\mathcal{K}^{(j)}_{A\to B}$, m and N_{tr} are the dimensionality of the BNFs and the number of training frames respectively, and subscript ab indicates the element in row a, column b of a matrix. At each step, the number of clusters N_K in $\mathcal{K}_{A\to B}$ decreases by 1. The centroid transforms are updated at each iteration by re-calculating the means of the transforms in the merged clusters.

The result of clustering the piecewise linear transforms from BNF set A to B is depicted in Figure 6.5. The top part describes the clusters formed at each step as the number of transform matrices decreases from 49 to 1, from left to right. At each step, two clusters are merged and marked with the same colour. The clustering follows an approximately phonetic sequence. First to merge is a pair of close vowels (/ih/ and /ix/), the new cluster marked red. Next, nasals /m/ and /n/ are merged and marked orange, then close vowels /ah/, /eh/ (yellow). The fifth merge (subsuming the yellow cluster into the red) merges the two-phone clusters /ih/, /ix/ and /ah/,/eh/. As clustering progresses, we see vowel-like sounds clustered earlier, merging with nasals and some fricatives relatively early. Consonants are comparatively more "scattered", usually merging in pairs and only later forming a big group.

The lower part of Figure 6.5 shows recognition performance using models obtained by the piecewise linear transformations at each clustering stage (in blue) and mean square error (MSE) between original and transformed features of the test set (in red). The recognition accuracy sees a big drop when the number of clusters decreases from 7 to 6, when two relatively large groups are merged - one containing vowels, most nasals, and some voiced fricatives; the other group, closures, plosive and some fricatives. The other five clusters contains sibilants, the affricate pair /ch/, /jh/, velar plosive pair /g/, /k/, and singleton clusters /p/ and /sil/. The plots of MSE evaluates the feature transformation from another perspective, and the conclusions from it is the same as the recognition accuracy graph, but to show a direct piecewise linear relationship in the bottleneck feature space without the help of the model space. Note that the MSE experiment is only an additional test to the phone recognition experiment, and it cannot

prove things on its own because it is taking phone label information of the test data.

The hierarchical clustering task has also been applied to other pairs of BNF sets with similar results. Although the clustering results were not identical, the combined linear transforms which emerge correspond to intuitively meaningful phonetic categories. In each case, the biggest decreases in phone recognition accuracy occurs when combining transforms corresponding to categories that differ significantly in their phonetic properties.

The mappings between sets of BNFs are therefore more similar for phonetically similar phones than for phone in different phonetic categories. It may be that this can be interpreted as evidence that the networks learn different representations for different phonetic categories, and that these are generated by phone category-specific mappings from acoustic to BNF space.

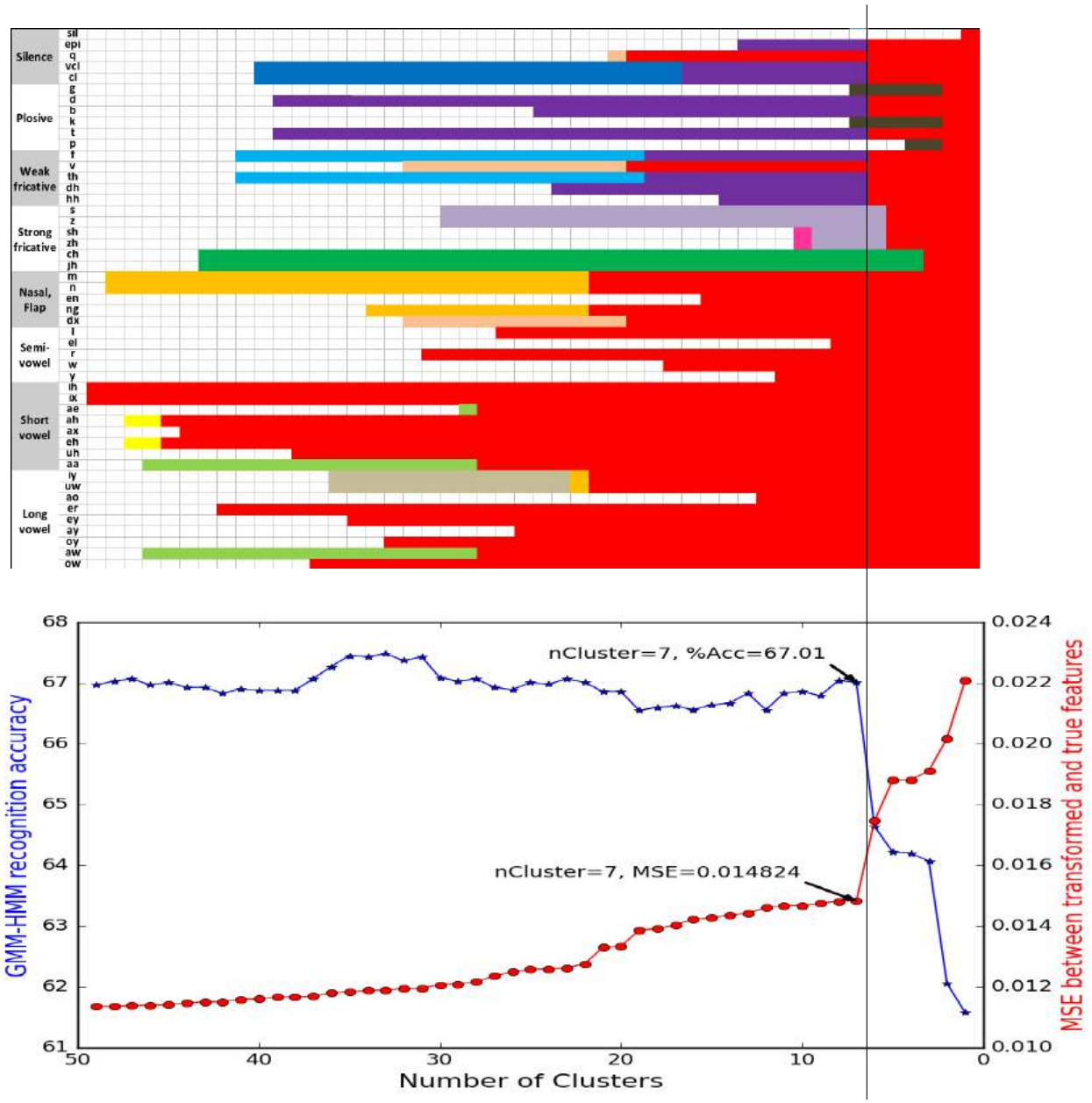


Figure 6.5: *Hierarchical clustering of 49 transform matrices (upper part) and corresponding recognition accuracy (lower part blue) and mean squared error between transformed and target BNFs (lower part red).*

6.6 Summary and discussion

This chapter explores the relationships between BNFs resulting from different neural network weight initialisations. We show that the relationship between them is approximately piecewise linear. We present the results of experiments in which hierarchical clustering is applied to phone dependent linear transformations between BNF sets, and show that the combined linear transforms that emerge correspond, in general, to phonetically meaningful phone classes. In addition, we show that the biggest decreases in phone recognition accuracy occur when transforms corresponding to categories that differ significantly in their phonetic properties are combined.

This result suggests that the type of feature extraction applied by the network to extract BNFs is different for different phone categories. The network is able to learn and combine multiple phone category dependent feature extraction mappings to optimise a low-dimensional representation for its phone classification task. This raises the question of whether it is better to treat from spectra space to bottleneck space as a single mapping or a set of phone class dependent mappings. If the feature extraction by DNN changes when it moves across phone categories, is it advantageous to use different DNNs for different phone classes? In this case the appropriate mathematical structure is a manifold. We continue with this in the next chapter.

Chapter 7

Phone Classification using a Non-Linear Manifold with Broad Phone Class Dependent DNNs

7.1 Introduction

Most state-of-the-art ASR systems use a single DNN to map the acoustic space to the decision space via a sequence of layers. Likewise in the previous chapters, single global DNNs were used to map spectra onto BNFs. However, different phonetic classes employ different production mechanisms, and so may warrant different types of feature extractions. Therefore, it may be advantageous to replace this single DNN with several phone class dependent DNNs. The appropriate mathematical formalism for this is a manifold, which has been described in Section 2.5.

Previous work has considered systems comprising multiple linear phone class dependent mappings (Huang et al., 2016), and our approach in this chapter can be regarded as a non-linear extension of this work. This chapter extends the previous study of very low-dimensional BNFs, including phone classification and visualisation and interpretation. The objective is to determine whether it is advantageous for phone-classification

of feature vectors to treat the acoustic space A as a non-linear manifold, in which several broad phone class (BPC)-dependent DNNs rather than a single DNN are used for phone classification.

Various ways of designing the system and training the DNNs are assessed, and the use of different BPC definitions are explored. Also we propose a new way for testing the significance of improvements for neural network based systems, with the effect of NN initialisations taken into consideration. The results show a small but significant improvement of 3.6% compared with a single DNN, when a non-linear manifold structure incorporating multiple BPC-dependent DNNs is used for phone classification. Experiment details are in Sections 7.3.1 to 7.3.5. LDA visualisations are applied to the BNFs from BPC-dependent DNNs and are compared to those from baseline global DNNs. It is shown that BPC-dependent DNNs learn clearer local structures than a global DNN.

7.2 Experimental setup for learning non-linear manifolds with BPC-dependent DNNs

7.2.1 Proposed structure of phone classification systems

Two neural network structures are proposed for the phone classification task, inspired by a non-linear manifold model of speech. Both structures take speech spectra with context as input, as was the case for the experiments in previous chapters. A 9-node bottleneck layer is used in each BPC-dependent network to enable comparison with the baseline single global bottleneck neural network.

Structure I

The first structure of phone classification system comprises three levels. The structure is depicted in Figure 7.1. The first level, shown by the left-hand part of Figure 7.1, is a BPC classification network that estimates the probability of a phone being in

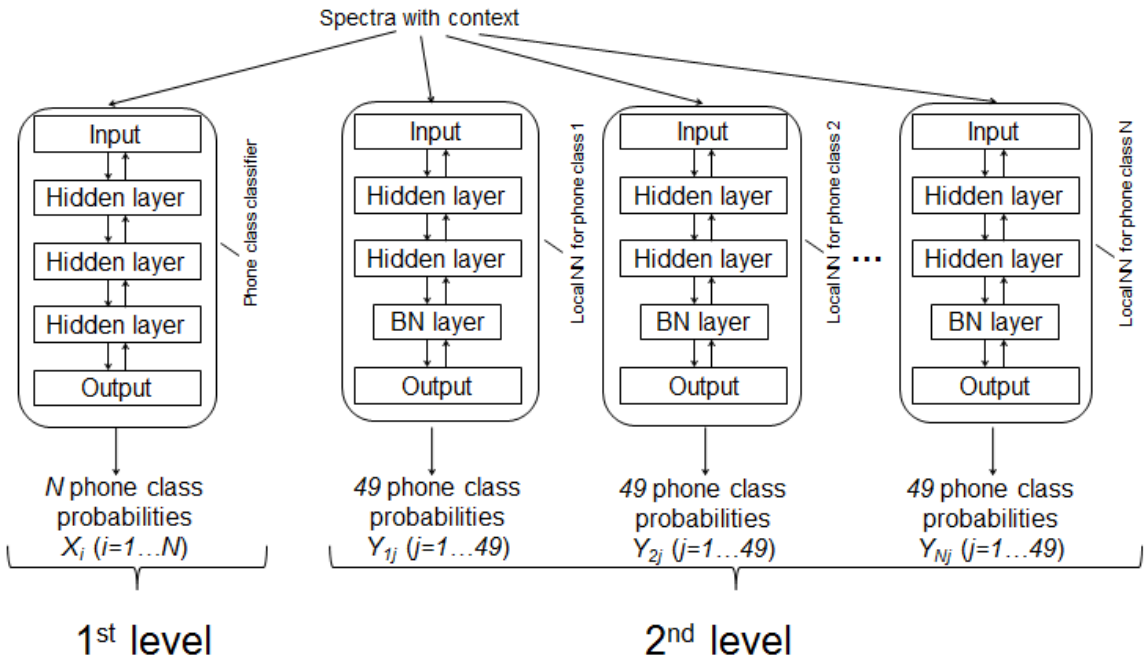


Figure 7.1: *Architecture I of phone classification system exploiting DNN-based manifold learning of speech (first two levels).*

each broad phone class. In our experiment we used 8 phone classes (as described in Section 2.3.1), thus we have 8 nodes at the output layer of the first level DNN. We denote the output of the i^{th} node as $X_i(i = 1...8)$. The second level (right-hand part of Figure 7.1) is a set of N parallel DNNs, each focusing on a particular part of the speech acoustic space, i.e. a set of BPC-dependent phone classification networks. The BPC-dependent DNNs are trained to distinguish between individual phones within each named phone class. We denote the output of the j^{th} node of the i^{th} BPC-dependent DNN as $Y_{i,j}$. Note that in all the BPC-dependent DNNs, there are 49 output nodes defined on the full phone set. Although in the second level not all output nodes are involved in the training process, they are needed for the test process.

The third level (not included in Figure 7.1) is only involved in the test process. In this level we calculate a sum over the output posterior probabilities from the neural networks in the second level, weighted by the posterior probabilities of phone classes

according to the first level. For each frame, we calculate

$$Z_j = \sum_{i=1}^{8} X_i Y_{i,j}, \tag{7.1}$$

resulting in a 49-dimensional vector Z_j, where the value of each dimension indicates a phone probability. The phone that corresponds to $\hat{j}$

$$\hat{j} = \arg\max Z_j \tag{7.2}$$

is the phone classification decision to this frame.

Note that both the first and the second level take spectra-in-context as input, and that during the training process, one BPC-dependent DNN is only trained on one phone class (within-class data only).

Structure II

The second structure is shown in Figure 7.2. It is in some sense up-ended from Structure I. It comprises two levels. The first level is a set of N parallel DNNs, which implement the non-linear local mapping functions $\phi_i (i = 1, ..., N)$, each focusing on a particular part of the speech acoustic space A. The second level fuses the outputs from the individual local mappings in the first level to arrive at a final classification decision.

At the first level, all the training data is passed to each network, regardless of which broad class that network is focussing on. This ensures that a given local network has information about data which do not belong to its BPC. Suppose that the i^{th} local DNN implements a mapping ϕ_i for the subset Q_i of phones which comprise the i^{th} BPC. The output layer of this network has $K_i + 1$ nodes, where the first K_i nodes correspond to each phone in the category Q_i and the additional node is used to indicate the "out-of-class" label used for input features which are not contained in the i^{th} BPC. The targets in the output layer are the phone or 'out-of-class' posterior probabilities. The DNN structure of a "plosive focused" local DNN is shown in Figure 7.3 as an example.

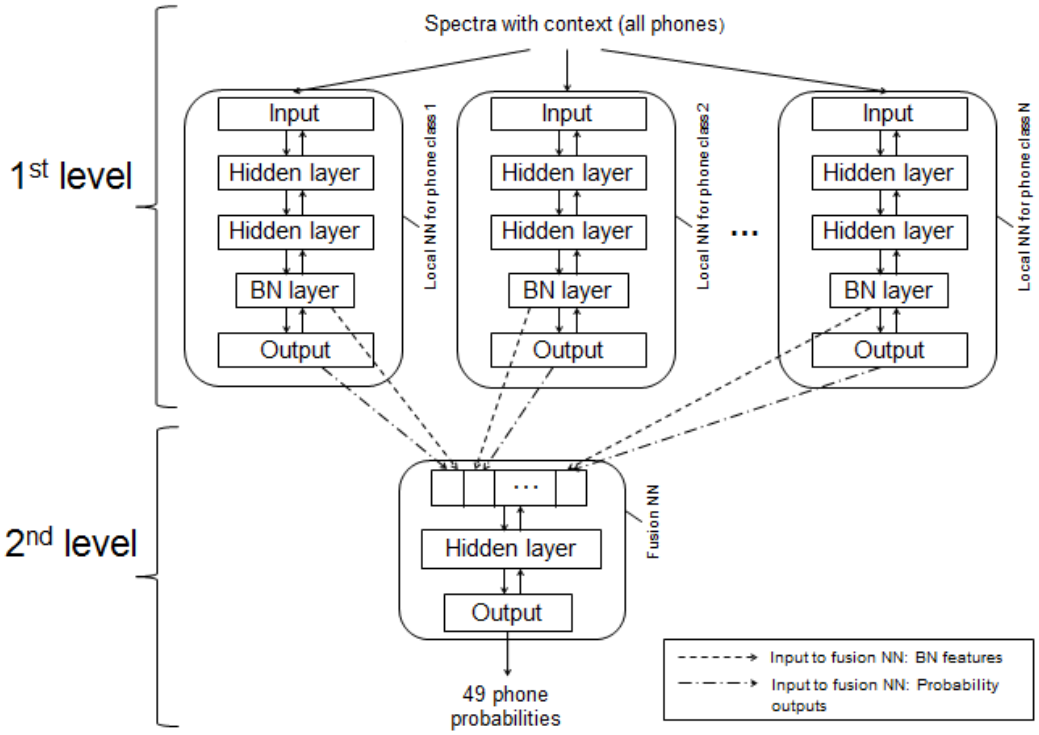


Figure 7.2: *Architecture II of phone classification system exploiting DNN-based manifold learning of speech.*

The second level of the classification system is a "fusion network" that serves to provide the final phone classification decision. The input vector passed to this second level contains information concatenated from all the first level BPC-dependent DNNs. In out experiments, we used either the outputs of the bottleneck layer, or the softmax probability outputs from the output layer of each first level networks. This is indicated by dashed and dot-dashed lines in Figure 7.2. The output of the fusion network is a vector of posterior phone probabilities of 49 phones.

7.2.2 Speech corpus and phone class

Experiments are performed on the TIMIT corpus (described in Section 3.1) including the phone level segmentations and labelling, as used in previous chapters. In addition to the standard TIMIT data, we also use a subset of TIMIT that only contains information of phone centres, where the phone centre timings are decided by the TIMIT labelling. This is obtained by selecting only the spectra-in-context vectors correspond-

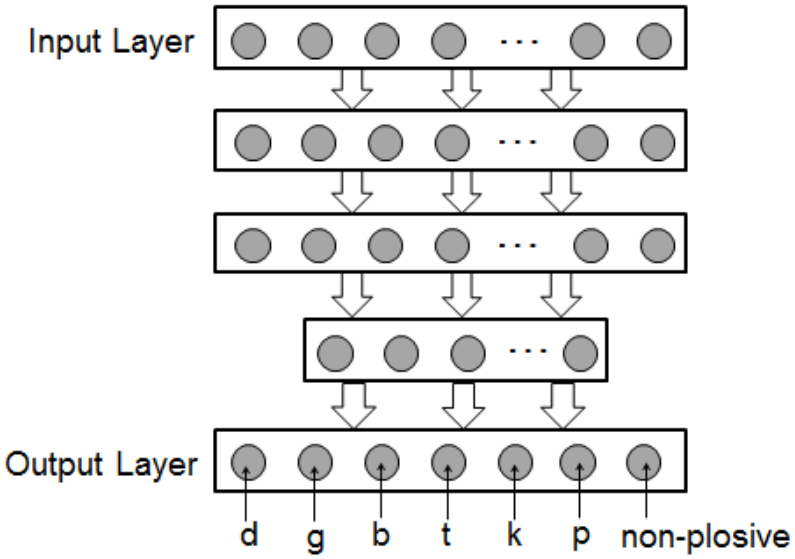


Figure 7.3: *Architecture of a "plosive focused" local DNN used in the first level of the Structure II.*

ing to the centre frame of each phone segment from the overall spectra-in-context data. Intuitively, the phone labelling in this "centre frames only" subset should be more reliable.

The 61-phone set used in TIMIT labels is mapped to the 49-phone set (the mapping can be found in Section 3.2), before the phones are grouped to BPCs. As the baseline, the phones are grouped into the 8 non-overlapping BPCs as described in Section 2.3.1. This is the same as the BPCs used by (Huang et al., 2016). These correspond to BPCs Q_1–Q_8 in the upper part of Table 7.1. For structure II, we also use several "super broad" classes which are unions of two or more BPCs from Q_1–Q_8. These are defined in the lower half of Table 7.1. The super broad classes are defined based on the definition used in (Huang et al., 2016). Plosive and weak fricatives are combined; short vowels, long vowels and semi-vowels are combined in all possible ways. This way of defining super broad phone classes echoes the observation we obtained in Chapter 5, where the visualisations of the BNFs showed confusions (points overlapping) between plosive and weak fricatives, and between short, long and semi vowels. In addition, one super broad class is defined to cover all phones (Q14).

Table 7.1: *Phonetic broad classes used to define the set of local DNN-based projections.*

Group	Phonetic class	Phone label
Q_1	Plosive	/g/, /d/, /b/, /k/, /t/, /p/
Q_2	Strong fricative	/s/, /z/, /sh/, /zh/, /ch/, /jh/
Q_3	Weak fricative	/f/, /v/, /th/, /dh/, /hh/
Q_4	Nasal/Flap	/m/, /n/, /en/, /ng/, /dx/
Q_5	Semi-vowel	/l/, /el/, /r/, /w/, /y/
Q_6	Short vowel	/ih/, /ix/, /ae/, /ah/, /ax/, /eh/, /uh/, /aa/
Q_7	Long vowel	/iy/, /uw/, /ao/, /er/, /ey/, /ay/, /oy/, /aw/, /ow/
Q_8	Silence	/sil/, /epi/, /q/, /vcl/, /cl/
Q_9	$Q_5 \cup Q_6 \cup Q_7$:	Semi-vowel, Short vowel, Long vowel
Q_{10}	$Q_1 \cup Q_3$:	Plosive, Weak fricative
Q_{11}	$Q_5 \cup Q_6$:	Semi vowel, Short vowel
Q_{12}	$Q_5 \cup Q_7$:	Semi vowel, Long vowel
Q_{13}	$Q_6 \cup Q_7$:	Short vowel, Long vowel
Q_{14}	$Q_1 \cup Q_2 \cup \ldots \cup Q_8$:	All phones

7.2.3 Neural network training

We used the same corpus and spectra-in-context data as used in previous chapters to train DNNs. They are 286-dimensional vectors, containing logFBEs of the current frame and five preceding and five following frames.

The networks were trained as deep belief networks (DBNs) with Gaussian-binary restricted Boltzmann machines (GRBMs, for the bottom RBM learning spectra-in-context) and restricted Boltzmann machines (RBMs, for all the others) pre-training. Stochastic gradient descent back-propagation was conducted using the Theano toolkit (Bastien et al., 2012; Bergstra et al., 2010). The learning rates of GRBM, RBM and fine-tuning were 0.002, 0.02, 0.1 respectively. In the fine-tuning process the training stopped when the error on the validation set started to rise or when the epoch reached the maximum of 100[1]. For the experiments on centre feature vectors, the neural networks were further fine-tuned using only feature vectors corresponding to the centre

[1]This is notably different from the 3000 maximum epochs in previous chapters. Because the pre-training step has provided a better starting point, it usually takes less than 100 epochs before the training error on the validation set stop decreasing.

frames of the TIMIT phone segments.

Most neural networks were trained as DBNs with pre-training. For experiment using the "centre frames only" data, an additional fine-tune process is included. In this process we take the weights of the neural network trained on the "all frames" data as a starting point and train on the "centre frames only" set for a few more times (maximum epochs was set to 100).

7.2.4 System evaluation and test of significance

The systems were evaluated with respect to their ability of classifying phones at the feature vector level. This is different from the previous chapters where phone recognition performance was evaluated. Two sets of experimental evaluations were performed: i) using all the feature vectors, and ii) using only the centre feature vector from each TIMIT phone segment. When evaluating phone classification accuracy, the 49 phone set was reduced to 40 according to (Lee and Hon, 1989).

For the test of significance, DNNs are trained multiple times with various initialisations to reduce the influence caused by DNN training initialisations. Pairwise comparisons between global neural networks and local neural network systems are applied using the McNemar's significance test (Gillick and Cox, 1989). We say one system is significantly better than the other if in more than 95% of the pairwise comparisons the improvement in performance is significant at the 0.05 level according to the test.

7.3 Experiments and results

7.3.1 Baseline: a global bottleneck neural network

We use the DNN structure 286-1024-1024-9-49 trained with deep belief networks (DBN) as the baseline in the experiments in this chapter, as this was found to achieve best performance — it was found that using DNNs trained as DBNs (described in Section 2.2.3) and with the bottleneck layer placed directly before the output layer, gave

better results. The settings for the neural network retraining are described in Section 7.2.3. Table 7.2 compares the GMM-HMM monophone recognition performance using 9-dimensional BNFs from this "new baseline" neural network structure and the structure used in the previous sections (NN286-512-9-512-49 trained by standard SGD back-propagation), which can be regarded as an extended experimental result for the experiment in Chapter 4.

Table 7.2: *Evaluation of BNFs in a GMM-HMM HTK recogniser on the TIMIT core test set.*

NN structure	%Corr	%Acc
286-512-**9**-512-49	73.1	69.4
286-1024-1024-**9**-49	76.3	72.5

In this chapter we report only phone classification performance rather than phone recognition. The baseline phone classification performance of the single global DNN is shown in Table 7.3. The results are on TIMIT full and core test sets, averaged over 20 DNNs with different training initialisations. The 40 phone set was used when calculating classification accuracies. We obtained results using all frames of the TIMIT corpus, and on only centre frames of phone segments. As expected, the classification accuracies on the "centre frames only" data is considerably higher than that on the "all frames" data. The method for training and testing a DNN on only centre frames is described in Section 7.2.3.

Table 7.3: *Global DNN phone classification results on the TIMIT test set. These are baseline results for experiments in this Chapter.*

Test data	#Frame tested	%Acc
Full TEST, all frames	410920	67.70
Full TEST, centre frames	51680	76.57
Core TEST, all frames	57919	67.60
Core TEST, centre frames	7333	76.81

7.3.2 Comparisons between the two proposed structures

The 8 non-overlapping BPCs are used to train local BPC-dependent DNNs. Following the baseline DNN structure, we also use three hidden layers in the BPC-dependent

DNNs, where the last hidden layer is a bottleneck layer containing 9 neurons. When deciding the size of non-bottleneck layers, we tried to keep the overall number of parameters in the different systems as close as possible.

In this experiment, we train 8 parallel BPC-dependent DNNs with a bottleneck layer, for both Structure I and Structure II. The size of BPC-dependent DNNs is set to 286-256-256-9-X, where X is 49 in the system of Structure I and $K_i + 1$ in the system of Structure II, as described in Section 7.2.1. The structure of the BPC classification neural network in Structure I is 286-256-256-256-8, and the structure of the fusion network in Structure II is Y-32-49, where $Y = \sum_i K_i + 8$. The fusion network has only one small hidden layer because there was no benefit observed when enlarging this hidden layer or using more layers.

The phone classification performance of the two structures are shown in Table 7.4. We can see that the Structure II outperforms the Structure I in the phone classification task, with fewer parameters.

Table 7.4: *Phone classification performance on the TIMIT core test set.*

Structure	%Acc (all frames)	%Acc (centre frames)	# System parameters
Baseline	67.60	76.81	1,353,203
Structure I	66.85	75.57	1,343,888
Structure II	69.38	77.45	1,136,659

We use the Structure II for our phone classification system in the following experiments.

7.3.3 Changing the ratio of in-group/out-group data

As described in Section 7.2.1, the BPC-dependent DNNs in Structure II are trained on all the training data, regardless of which broad class that network is focussing on. However, as the "out-of-class" data is usually more than the "in-class" data[2], the BPC-dependent DNNs might be overfitting the "out-of-class" data. Is it beneficial to

[2]The proportion of data in each BPC is shown in Figure 3.1 in Section 3.1.

increase the proportion of in-class data when training the BPC-dependent DNNs? We try to answer this question in this subsection.

We tried different proportions of in-class and out-of-class data for each BPC-dependent DNN. This was achieved by either repeating the in-class data (double or four times) while keeping the amount of out-of-class data unchanged, or decreasing the out-of-class data (half, quarter, and 1/8) while keeping the amount of in-class data unchanged. The overall system classification performance was tested and the accuracies are shown in Figure 7.4. The classification performance of the individual BPC-dependent DNNs were also tested. The results are included in Appendix A.

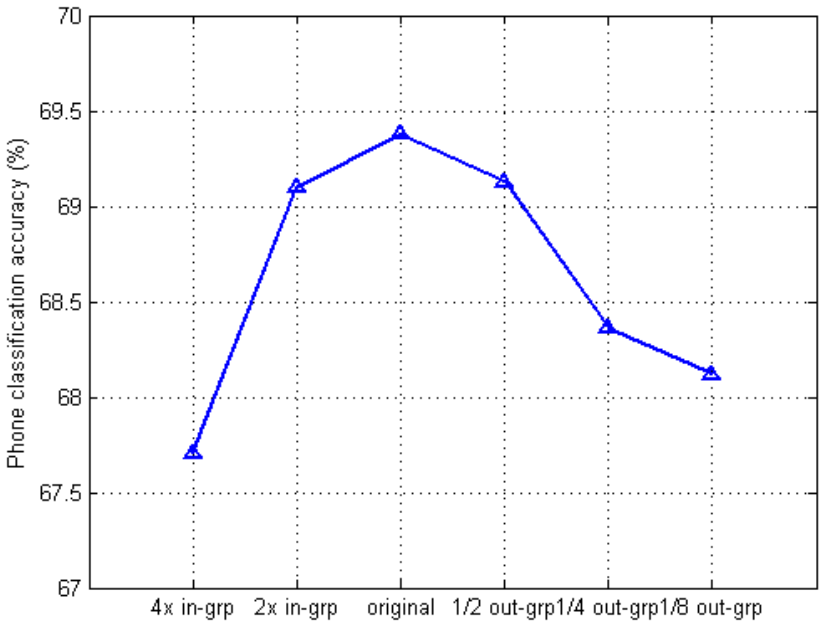


Figure 7.4: *Overall phone classification accuracy when varying the ratio of in/out group data.*

From Figure 7.4 we see the overall performance of phone classification, i.e. the result of the fusion network, goes down whether the proportion of the "in-group" data is increased or that of the "out-of-the-group" data is decreased. One explanation could be that the proportion of in-class data for all BPC-dependent DNNs is small in the test set, thus the system needs the BPC-dependent DNNs in the first level to be capable of well classifying "out-of-class" data.

Note that the differences in classification accuracies are actually quite small and may not indicate a disadvantage of increasing the ratio of in/out group data. Multiple runs with different initialisations may be needed to see if the differences are significant. However, we have enough evidence to decide to keep the original ratio of in/out group data in the following experiments in this chapter.

7.3.4 Including "super" broad classes

In this subsection, we explore if the use of "super" broad phone class could improve the phone classification system. We consider five different BPC groups, D_1 to D_5 (Table 7.5). These determine the sets of local BPC-dependent DNNs.

D_1 consists of the 8 non-overlapping BPCs Q_1 to Q_8 (Table 7.1). D_2 consists of D_1 plus the additional class Q_9 which combines the three vowel sub-categories. D_3 consists of D_2 plus Q_{10}, which combines plosives with weak fricatives. D_5 is the same as that used in (Huang et al., 2016), with additional classes (Q_{11}, Q_{12} and Q_{13}) for different combinations of vowel sub-categories plus a global class Q_{14} containing all phones. D_4 is D_5, without the global class Q_{14}.

Table 7.5: *The sets $D_1, ..., D_5$ of BPCs used to train local BPC-dependent DNNs in the two-level system.*

Broad phone class	Experimental setup				
	D_1	D_2	D_3	D_4	D_5
$Q_1 - Q_8$	X	X	X	X	X
Q_9		X	X		
Q_{10}			X	X	X
Q_{11}				X	X
Q_{12}				X	X
Q_{13}				X	X
Q_{14}					X
# local DNNs	8	9	10	12	13

Table 7.6 shows classification results for all frames and centre frames only for the single global DNN and the two-level manifold structures corresponding to $D_1, ..., D_5$.

The figures are the averages of experiments performed over 20, 10 and 6 random DNN parameter initialisations for the global, local (softmax) and local (BNF), respectively.

First we focus on classification results using all frame feature vectors and taking softmax output from the local DNNs as the input to the fusion network (first column of the results in Table 7.6). The average phone recognition accuracy for the single global DNN is 67.60% with standard deviation of 0.48. The two-level structure with local DNNs gives in all cases better performance, which in many cases presents a significant improvement ("*" indicates that in 95% of pairwise comparisons between global and local networks, the difference in performance is significant at the 0.05 level according to the McNemar's test (Gillick and Cox, 1989).

The two-level systems corresponding to D_2 to D_5 give significant improvement over D_1 setup. The best performance (70.01% accuracy) is obtained with D_5. These results indicate that it is useful to include local network(s) that operate on a union of two or more BPCs, in particular, the union of vowel sub-categories or combination of pairs of vowel sub-categories, and also plosives with weak fricatives. This reflects the similarity in production of these sub-categories. The inclusion of such super-broad class networks may help to account for errors due to possible confusion between broad phone categories. For instance, the results in (Huang et al., 2016) indicate that there was a considerable confusion between these categories when performing classification of BPCs.

The results using only the centre feature vectors of each phone segment show similar performance trends to those observed for all feature vectors, however accuracy is considerably higher. This indicates that the classification error is higher during phone transitions. This is not surprising because those are the regions of phones that are more subject to articulation effects and therefore more confusable, in particular with neighbourhoods.

Table 7.6: *Phone classification accuracy obtained using all signal frames and using only the centre frames of each phone, and in each case using Softmax output and BNF as input to fusion network.*

	All frames		Centre frames	
Global DNN	67.60 (avg)		76.81 (avg)	
	69.05 (avg+3std)		77.58 (avg+3std)	
Local DNNs	Fusion net input		Fusion net input	
	Softmax	BNF	Softmax	BNF
D_1 (avg)	69.05*	68.78	77.45	77.03
D_2 (avg)	69.44*	69.23*	77.85	77.75
D_3 (avg)	69.56*	69.24*	78.31*	78.11*
D_4 (avg)	69.76*	69.31*	78.59*	78.08
D_5 (avg)	70.01*	69.63*	78.93*	78.70*

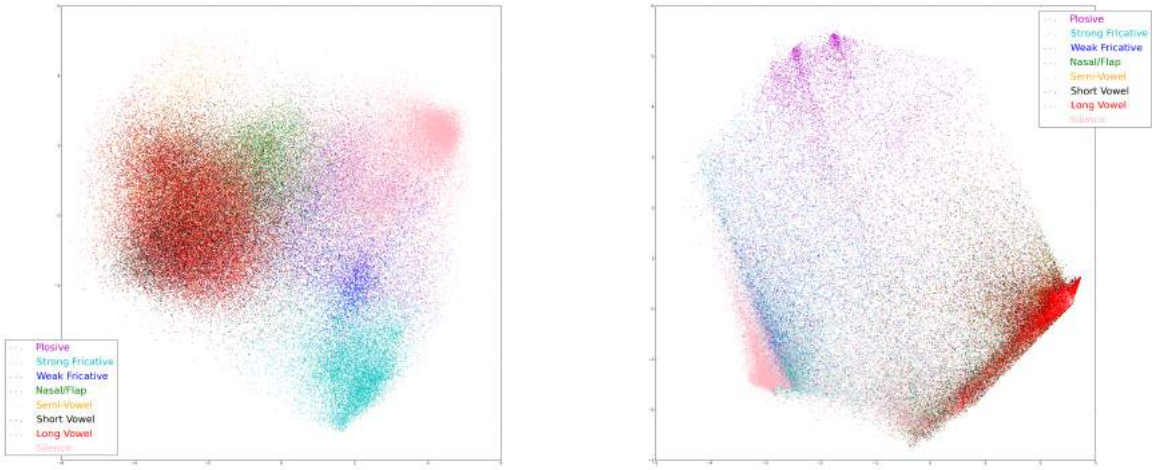
7.3.5 Comparison between different fusing inputs

We also explored passing different information from the local DNNs to the fusion network. We attempted using the bottleneck layer outputs and the softmax layer outputs as the input to the fusion network. The classification results have been listed in Table 7.6 ("Softmax" columns and "BNF" columns). We can see that using the probabilities from the output layer works a little better than using the BNFs obtained from the layer above the output layer. However, they both perform better in the phone classification task than the global structure using a single DNN.

7.3.6 Neural network visualisations with LDA

This section explores visualisations of the structures learned by local BPC-based DNNs. The 9-dimensional BNFs from the local DNNs are projected onto 2-dimensional space using linear discriminant analysis (LDA). This is conducted in the same way as the experiment in Section 5.2.1. As was observed in Section 5.2.1, the visualisations of BNFs for the training set and the test set are again very similar, thus we only show results for the training set. The plots marked (*) only show 10% of the frames in the training set for clarity.

Figure 7.5(a) is the 2-dimensional visualisations of the 1^{st} and the 2^{nd} dimension of LDA projections for the global DNN bottleneck layer (i.e. the baseline DNN). We first

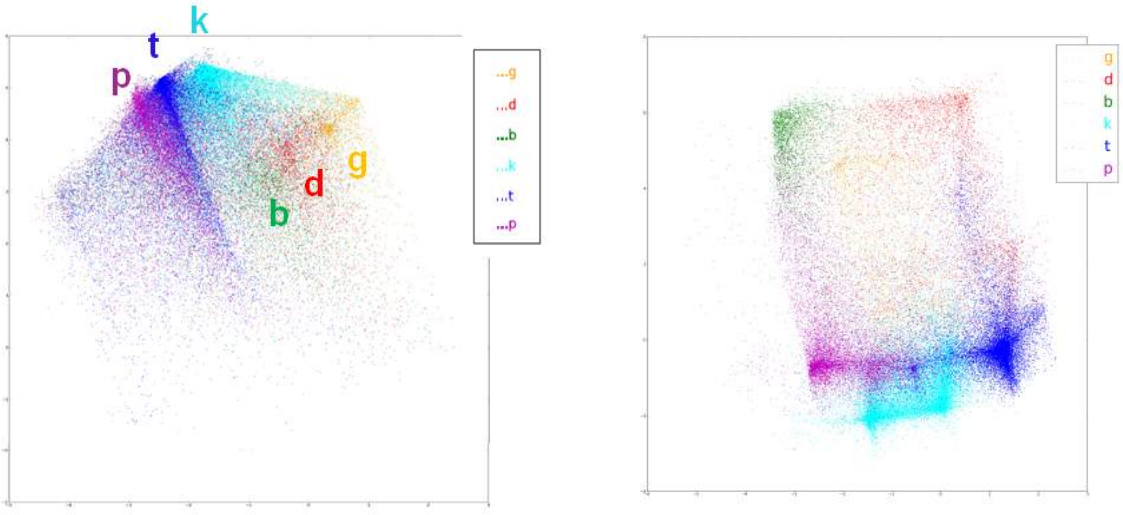


(a) Global DNN (286-1024-1024-9-49)(*) (b) Local DNN for plosive (286-256-256-9-7)(*)

Figure 7.5: *Visualisations of* 1^{st} *vs.* 2^{nd} *dimension of LDA-based projections of 9-dimensional BNFs from a single global DNN (a), and* Q_1 *('plosive') local DNN (b). Plot with data of all phones.*

look at an example 2-dimensional visualisations for the BPC Q_1 (plosives), shown in Figure 7.5(b) and Figure 7.6. Figure 7.5(b) shows the first 2 dimensions of the LDA projections for data from all phones. Plosives are represented in purple. Comparing Figure 7.5 (a) and (b), we can see that the purple plots representing BNFs of plosives is better separated from the other phones in the plosive focused DNN (Figure 7.5(b)), occupying a larger area in the 2-dimensional space. This indicates that the local neural network for plosives focuses more on the plosive class. Interestingly, although the non-plosive data were all assigned to 'out-of-class' category, the network has structured this data in an unsupervised manner according to phonetic categories.

Figure 7.6(a) shows data for "plosive" phones only, with each plosive represented in a different colour. It can be seen from Figure 7.6(a) that /p/, /t/, and /k/ are placed in an order which reflects their place of articulation (lips, teeth and soft palate, respectively). The voiced counterparts /b/, /d/, /g/ are placed in the same order but shifted towards the lower right. Figure 7.6(b) shows the plosive phone data projected onto LDA dimensions 3 and 4. Again, good separation of each plosive class is evident. Dimension 4 now seems to indicate voicing, with the unvoiced plosives placed in the lower part and voiced in the higher part of the figure. Again, the location structure for



(a) Plosive - 1^{st} vs. 2^{nd} dimension

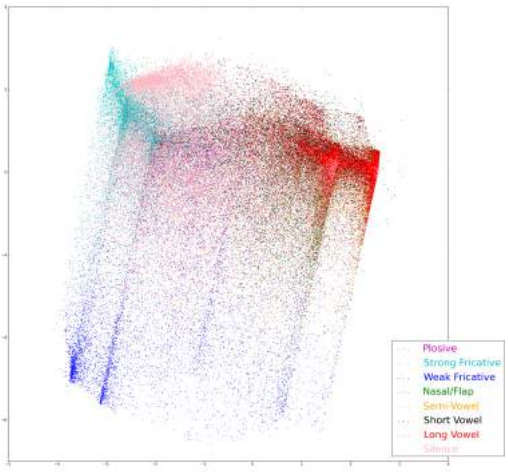
(b) Plosive - 3^{rd} vs. 4^{th} dimension

Figure 7.6: *Visualisations of LDA-based projections of 9-dimensional BNFs from Q_1 ('plosive') local DNN, for data within plosive class only. 1^{st} vs. 2^{nd} dimension (a) and 3^{rd} vs. 4^{th} dimension (b) .*

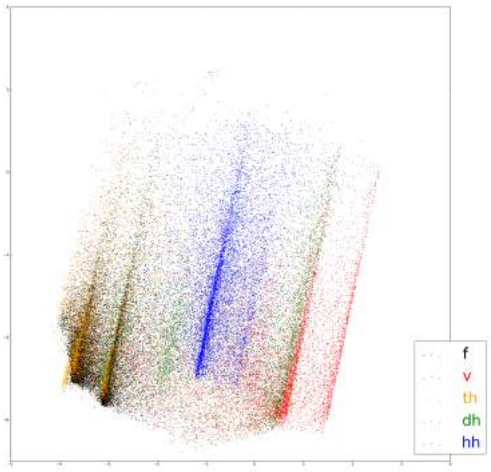
the unvoiced plosives is the same as for the voiced plosives, but shifted in dimension 4 for voicing.

We display the 2-dimensional visualisations (1^{st} and 2^{nd} dimension of the LDA-based projections) for other local DNNs focusing on weak fricatives, strong fricatives, nasals and flaps, semi-vowels, short vowels, long vowels and silences in Figure 7.7. The left-hand figures plot all phones (all comparable to Figure 7.5(a)), one colour for one BPC, whereas the right-hand figures plot only phones within the named BPC.

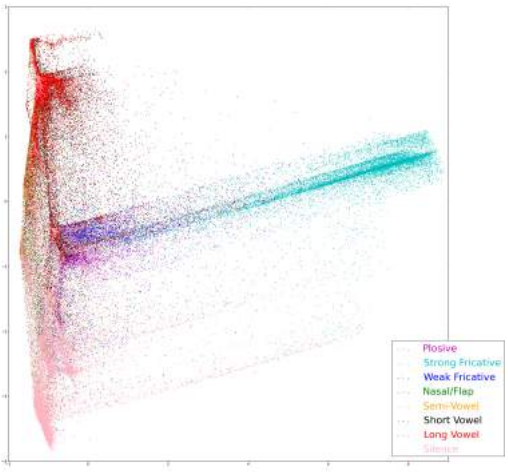
Similar to the local DNN for plosives, in all cases we see a good separation between the phones in the focused BPC and the other phones (figures on the left), and a fairly clear structure within the BPC that can be to some extent related to phonetic features (figures on the right). For example long vowels in Figure 7.7 (k) and (l), plots in (k) show a good separation between the plots of long vowels (in red) and the others; in (l) from bottom to top we observe */iy/*, */uw/*, */ey/*, */er/*, */ay/*, */aw/*, which roughly corresponds to the places of articulations (tongue positions) from high to low.



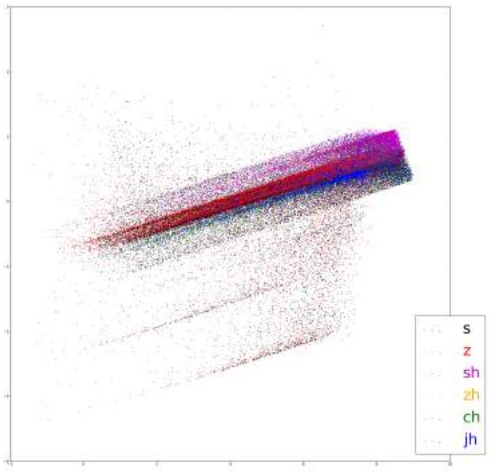
(a) Weak fricative NN (blue) - all phones(*)



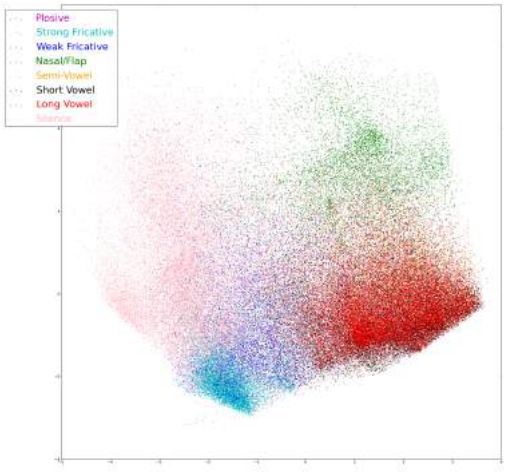
(b) Weak Fricative NN - in-group phones



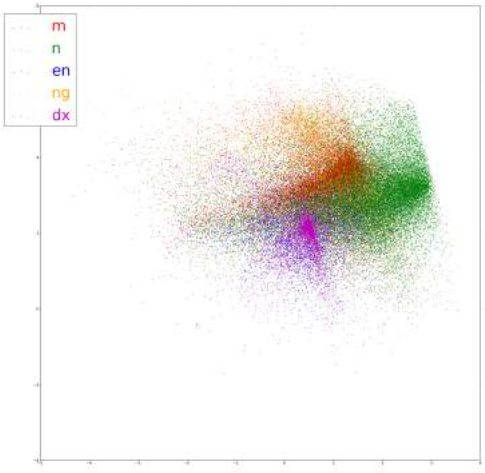
(c) Strong Fricative NN (cyan) - all phones(*)



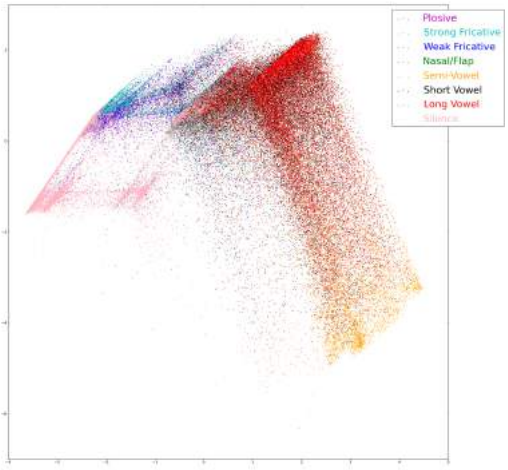
(d) Strong Fricative NN - in-group phones



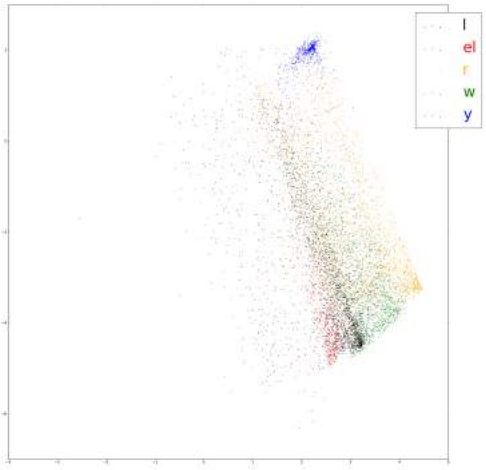
(e) Nasal&Flap NN (green) - all phones(*)



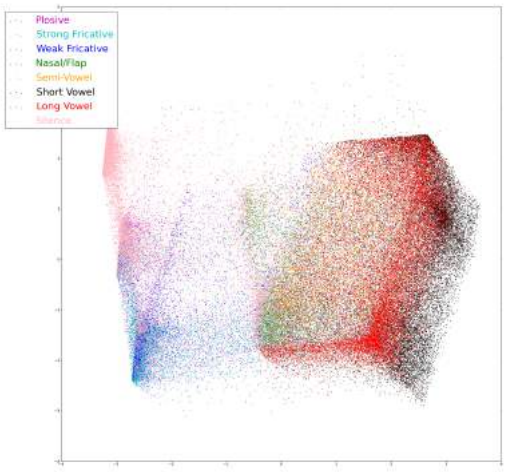
(f) Nasal&Flap NN - in-group phones



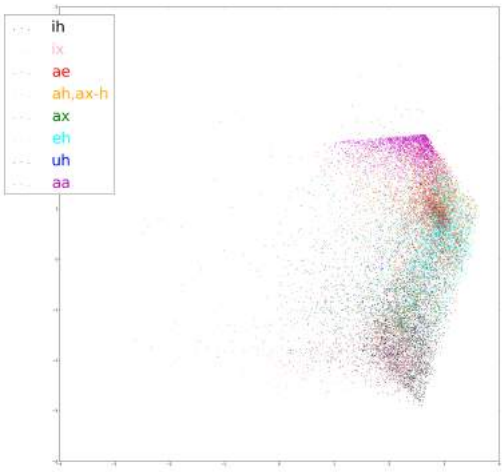
(g) Semi-vowel NN (orange) - all phones(*)



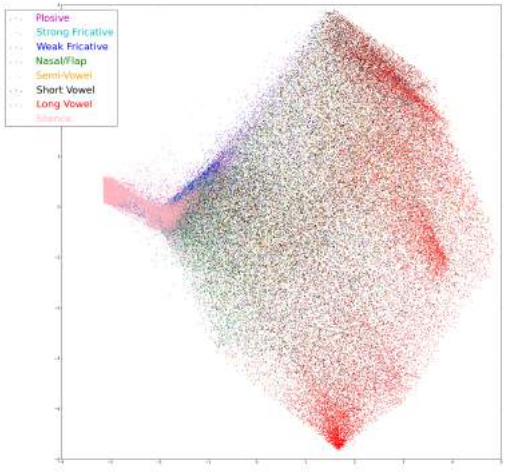
(h) Semi-vowel NN - in-group phones(*)



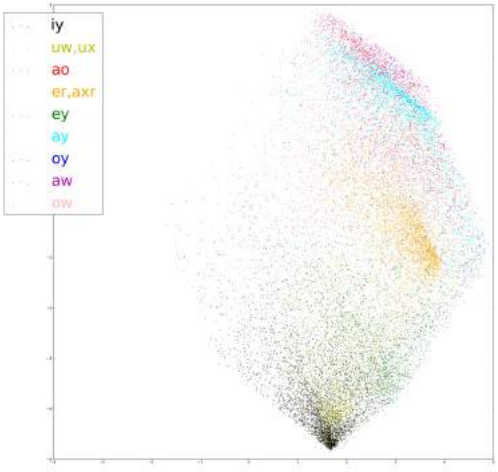
(i) Short vowel NN (black) - all phones(*)



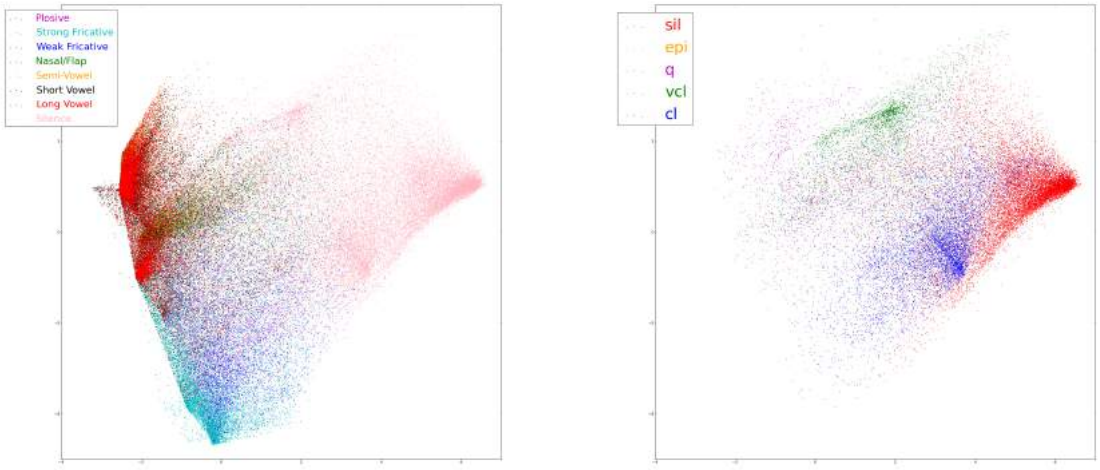
(j) Short vowel NN - in-group phones(*)



(k) Long vowel NN (red) - all phones(*)



(l) Long vowel NN - in-group phones(*)



(m) Silence NN (pink) - all phones(*)

(n) Silence NN - in-group phones(*)

Figure 7.7: *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of the BNFs from local DNNs. Plots are on all phones (left figures) and in-group phones (right figures)*

7.4 Summary and discussion

Most state-of-the-art automatic speech recognition (ASR) systems use a single deep neural network (DNN) to map the acoustic space to the decision space. However, It is generally accepted in the literature that different phonetic classes, corresponding to different production mechanisms, are best described by different types of features. Hence it may be advantageous to replace this single DNN with several phone class dependent DNNs. In this case, the appropriate mathematical structure is a manifold. Previous work has considered systems comprising multiple linear phone class dependent mapping (Huang et al., 2016), and our approach can be regarded as a non-linear extension of this work.

This chapter presents phone classification systems inspired by a non-linear manifold model of speech acoustic space. We presented two structures and chose to use one of them after an initial comparison. The structure used contained multiple BPC-dependent DNNs and a data fusion network. The systems used between 8 and 13 BPC-dependent mappings. Multiple trainings were conducted with different neural network weight initialisations. Pairwise comparisons between classification results from single global DNNs and proposed structures were made using the McNemar's significance test. The results show that using the BPC-dependent DNNs provides a small but significant improvement in phone classification accuracy in comparison to a single global DNN, with an average of 3.6% (comparative) improvement using the best structure obtained. It is demonstrated that in addition to the use of a set of local DNNs corresponding to basic BPCs, it is advantageous to also include local DNNs focusing on a combination of some BPCs, especially, vowel sub-categories. The use of the softmax outputs as input to the fusion network provides slightly better results than the bottleneck outputs in this experiment. Visualisations of the structures learned by the local DNNs indicate a relationship to speech production mechanisms.

Chapter 8

Conclusion

8.1 Contributions

The first aim of the work presented in this thesis is to find compact speech features that can be reliably estimated for all speech sounds and are suitable for segmental models of speech that parsimoniously model speech dynamics. In Chapter 4 we show that phone discrimination bottleneck neural networks can be used to extract very low-dimensional features containing sufficient information to support high-accuracy phone recognition. Specifically, 9-dimensional BNFs extracted from a phone discrimination bottleneck neural network provide better ASR phone accuracies than 39-dimensional MFCCs. In addition, BNF-based features outperform formant-based features of the same dimensions, with an averaged improvement of 33.7% (comparative) in ASR phone recognition accuracies. All the experiments were conducted using a conventional GMM-HMM recogniser. Subjectively, very low-dimensional BNFs well fit the assumptions of the Continuous-State HMM model, and can be obtained consistently for all phones.

Despite the remarkable improvements in speech recognition accuracy resulting from DNNs, our understanding of how (deep) neural networks work and what BNFs mean is limited. In Chapter 5 we visualise the 9-dimensional BNFs using LDA and t-SNE. We observe phonetically meaningful clusters in the projected 2-dimensional spaces. Apart

from mapping the 9-dimensional feature into 2-dimensional spaces, we narrowed the bottleneck layer to 2-units and extract 2-dimensional BNFs. Using a back-propagation method that computes the values of a given layer that are optimised for a particular output target, we obtain one "cardinal" 2-dimensional BNF for each phone and interpret these features. The 2-dimensional BNF space shows distinct regions used for each phonetic category, where within each category the organisation of phones appeared to correspond approximately to the phone production mechanisms. We also investigate the neural network neuron responses at the bottleneck layer and visualisations of non-bottleneck layer activations. We demonstrate that different parts of the network are involved in capturing different phonetic information, or information for different phonetic categories.

Neural networks trained with different weight initialisations would result in different BNF sets. Experimental results seem to suggest that NN initialisations do not significantly affect the final ASR results. However, it is a concern that a new initialisation may invalidate any phonetic description of BNFs. In Chapter 6 we explore the relationships between BNF sets resulting from different neural network weight initialisations. We show that the relationship between them is approximately piecewise linear. We present the results of experiments in which hierarchical clustering is applied to phone dependent linear transformations between BNF sets, and show that the combined linear transforms that emerge correspond, in general, to phonetically meaningful phone classes. In addition, we show that the biggest decreases in phone recognition accuracy occur when transforms corresponding to categories that differ significantly in their phonetic properties are combined. This result suggests that the type of feature extraction applied by the network to extract BNFs is different for different phone categories. The network is able to learn and combine multiple phone category dependent feature extraction mappings to optimise a low-dimensional representation for its phone classification task. In addition, this result suggests that any phonetic description of BNFs corresponding to particular phone class will remain valid, to within a linear

transformation, for BNFs resulting from a different initialisation of the same network.

It is generally accepted in the literature that different phonetic classes, corresponding to different production mechanisms, are best described by different types of features. In addition, it has already been noted that the mapping from spectra-in-context to phone posterior probabilities implemented by a DNN is a continuous approximation to a discontinuous function. All of these suggest that it may be advantageous to replace this single DNN with several phone class dependent DNNs. In this case, the appropriate mathematical structure is a manifold. Previous work has considered systems comprising multiple linear phone class dependent mapping (Huang et al., 2016), and our approach can be regarded as a non-linear extension of this work.

In Chapter 7 we present phone classification systems using multiple BPC-dependent DNNs and a data fusion network. The systems use between 8 and 13 BPC-dependent mappings. The structure is inspired by a non-linear manifold model of acoustic speech space. Multiple trainings were conducted with different neural network weight initialisations. Pairwise comparisons between classification results from single global DNNs and from the proposed structure comprising multiple BPC-dependent DNNs were made using the McNemar's significance test. The results show that using the BPC-dependent DNNs provides a small but significant improvement in phone classification accuracy in comparison to a single global DNN, with an average of 3.6% (comparative) improvement using the best structure obtained. It is also demonstrated that in addition to the use of a set of local DNNs corresponding to basic BPCs, it is advantageous to also include local DNNs focusing on a combination of some BPCs, especially vowel sub-categories. Visualisations of the structures learned by the local DNNs indicate a relationship to speech production mechanisms. The local DNNs learn clearer local structures than a global DNN.

8.2 Future work

Although the multiple BPC-dependent DNN structure proposed in this thesis is inspired by the notion of manifold, it does not constitute a true topological manifold. To obtain a true topological manifold, the "local" non-linear mappings ϕ_i should explicitly map subsets A_i of the acoustic space A into the BNF space B, rather than being determined by sub-classes Q_i of phones. This requires a better understanding of the topology of A and the relationships between its subsets and BPCs, which might be obtained through topological data analysis. In addition the ϕ_is should satisfy the consistency condition in Section 2.5. The latter could be investigated using "reconstruction" DNNs in which the targets are equal to the inputs, although this might compromise the utility of the BNFs for classification.

Experiments need to be conducted to confirm that the benefits of the local DNN structure for frame-level phone classification transfer to full ASR. Some possible approaches are discussed in the following paragraphs.

Experiments in this thesis used neural networks learning phone posterior probabilities. We can train neural networks to learn HMM state probabilities instead, which could be monophone states or triphone states. Therefore the multiple BPC-dependent DNN model studied in this thesis can be used to create the state-level probabilities required by a decoder. However, it is not clear that this structure is compatible with existing ASR toolkits such as Kaldi. The alternative is to develop a custom Viterbi decoder.

Currently the most successful ASR systems use various types of recurrent neural networks (RNNs) and convolutional neural networks (CNNs). It would be interesting to discover whether BNFs created using these types of networks would deliver superior ASR performance and to what extent they lend themselves to phonetic interpretation.

The experiments in this thesis could be extended to more practical speech data, for example noisy conversational speech.

Appendix A

Supplementary Figures

In this part, supplementary figures that results from experiments presented in Chapter 4 to 7 are included.

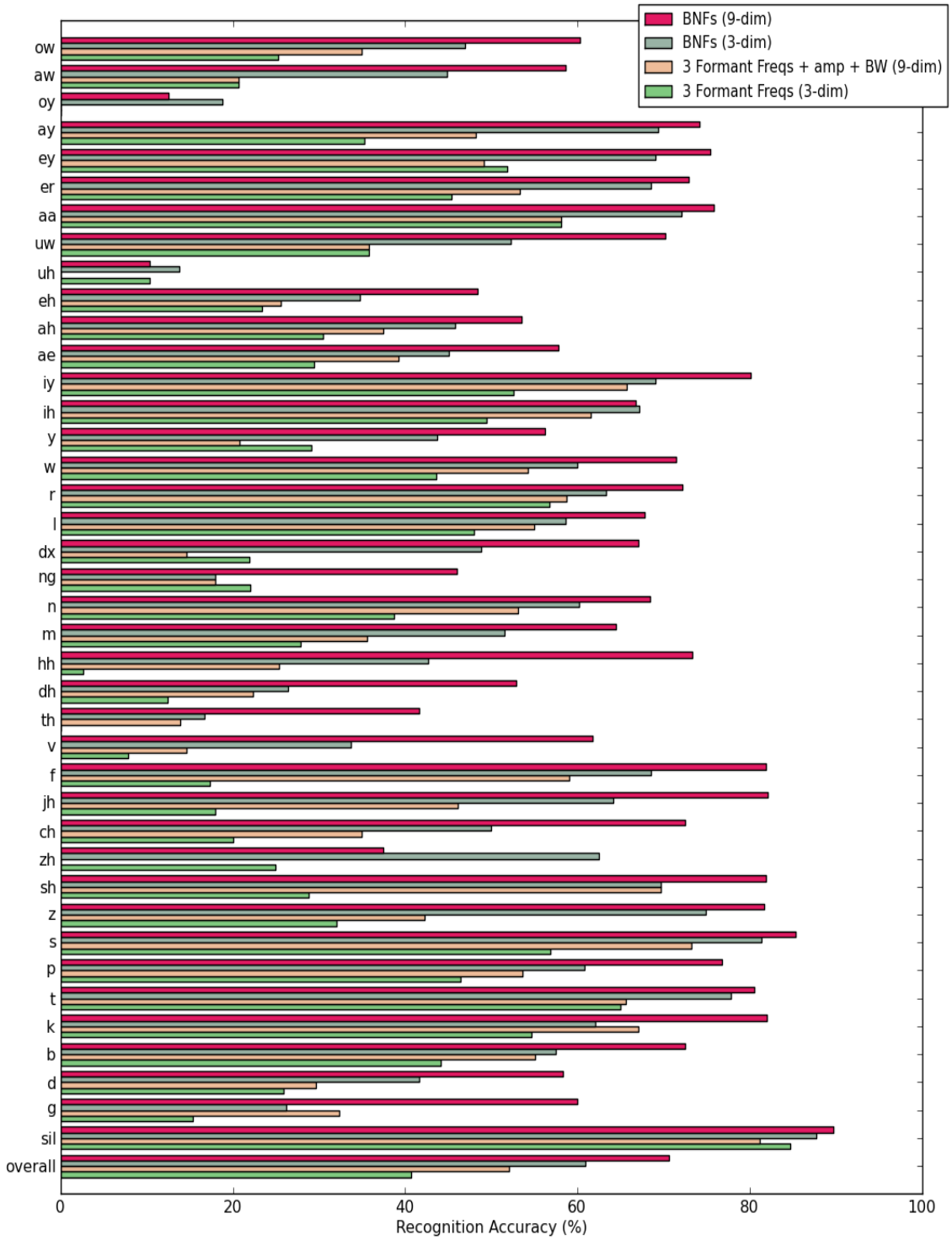


Figure A.1: *Comparing BNFs and formant features with GMM-HMM recognisers. For almost all phones, the ASR accuracy using these four types of features: 9-dimensional BNF > 3-dimensional BNF > 3 formant frequencies+ 3 amplitudes+ 3 bandwidths (9-dimensional) > 3 formant frequencies (3-dimensional)*

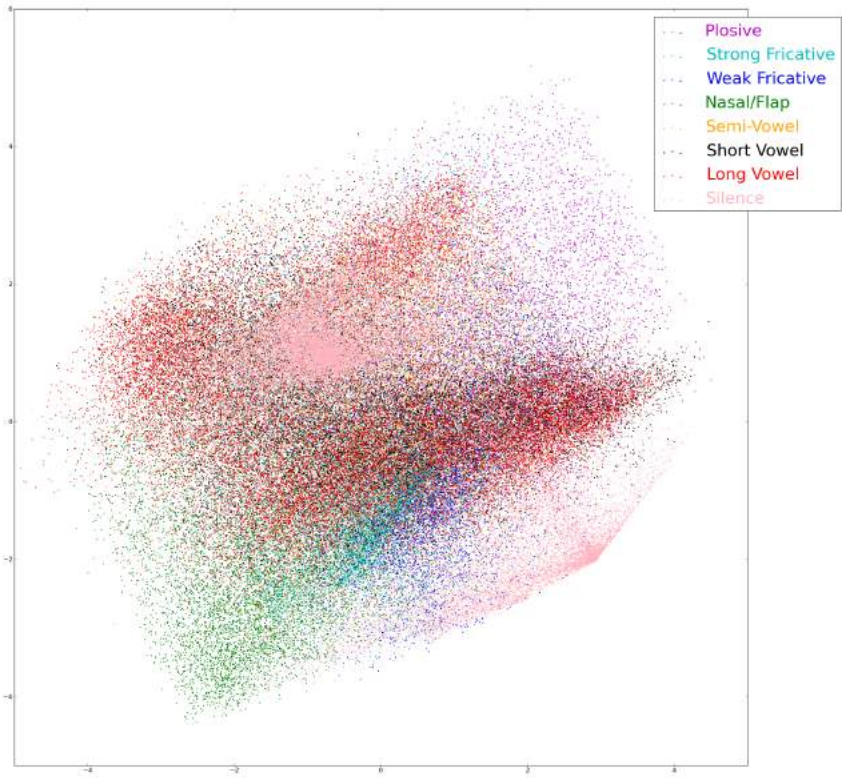
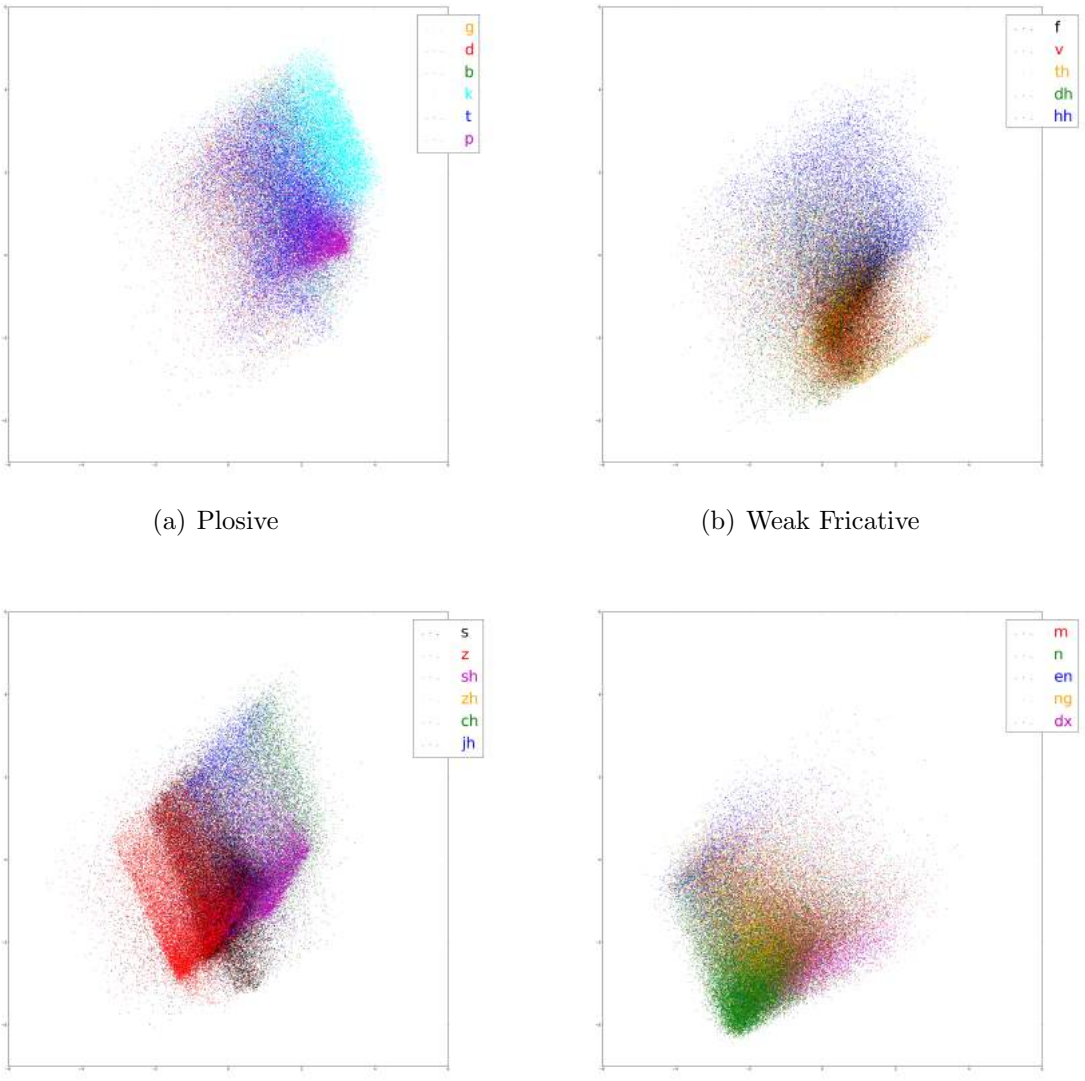


Figure A.2: *Visualisations of LDA-based projections (3^{rd} and 4^{th} dimension) of 9-dimensional BNFs from a phone classification DNN of structure 286-512-9-512-49, on the 10% subset of the training set. Plots are coloured by their corresponding phone classes. The visualisation of the 3^{rd} and the 4^{th} dimension of the LDA projections does not show separations between broad phone classes.*



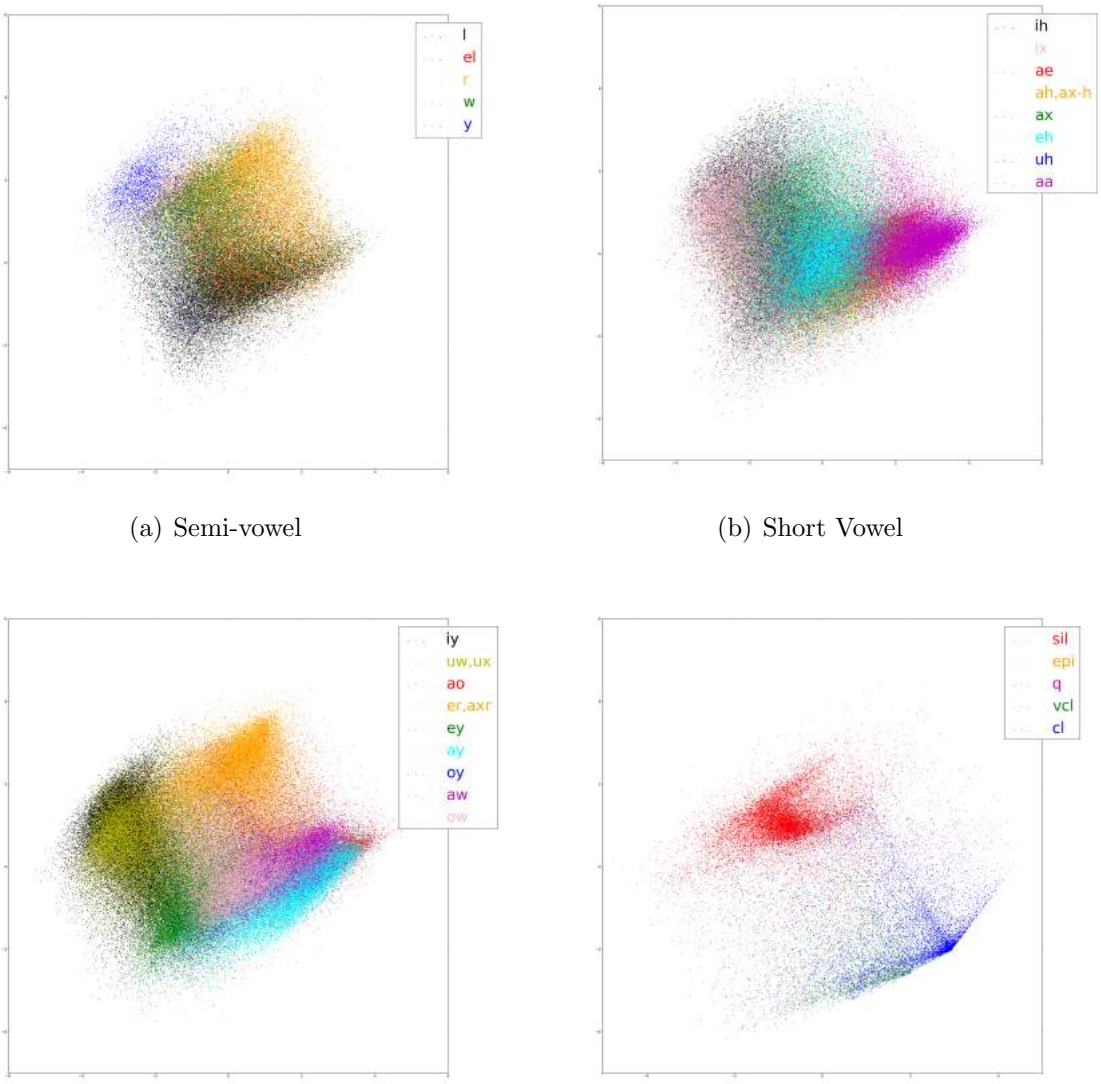
(a) Plosive

(b) Weak Fricative

(c) Strong Fricative

(d) Nasal, Flap

Figure A.3: *Visualisations of LDA-based projections (3^{rd} vs. 4^{th} dimension) of 9-dimensional BNFs from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure. (*)For silence only 10% of the data are plotted or clarity. We can observe separations within phone categories such as "strong fricative", "semi-vowel", "short vowel", "long vowel", and "silence"*



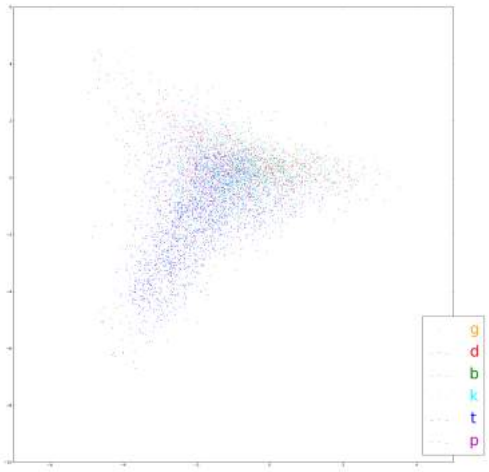
(a) Semi-vowel

(b) Short Vowel

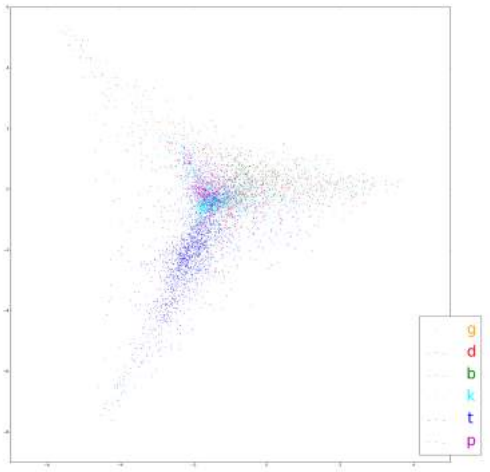
(c) Long Vowel

(d) Silence (*)

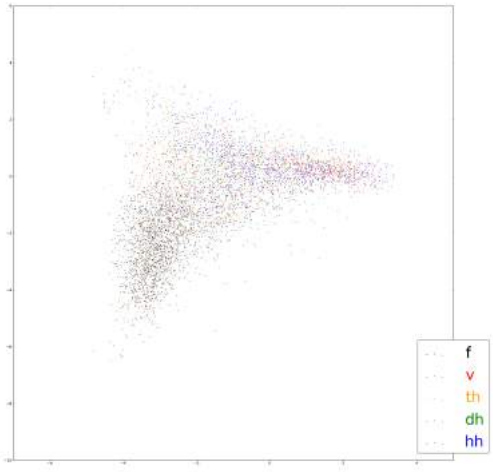
Figure A.3 (cont.): *Visualisations of LDA-based projections (3^{rd} vs. 4^{th} dimension) of 9-dimensional BNFs from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure. (*)For silence only 10% of the data are plotted or clarity. We can observe separations within phone categories such as "strong fricative", "semi-vowel", "short vowel", "long vowel", and "silence".*



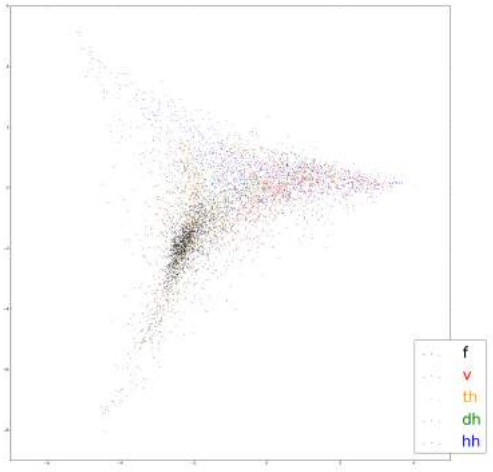
(e) Plosive (1^{st} Hidden Layer)



(f) Plosive (3^{rd} Hidden Layer)

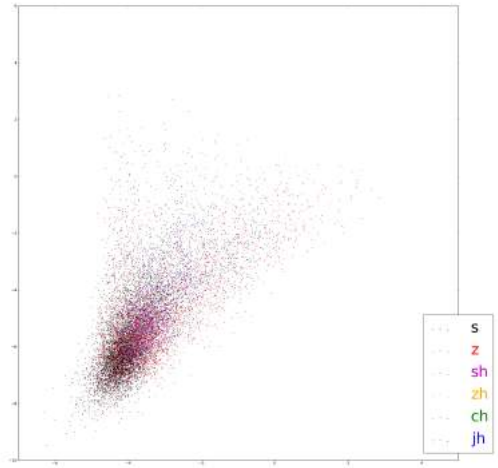


(g) Weak Fricative (1^{st} Hidden Layer)

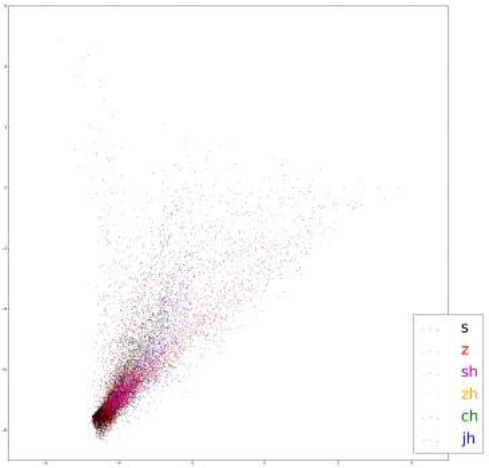


(h) Weak Fricative (3^{rd} Hidden Layer)

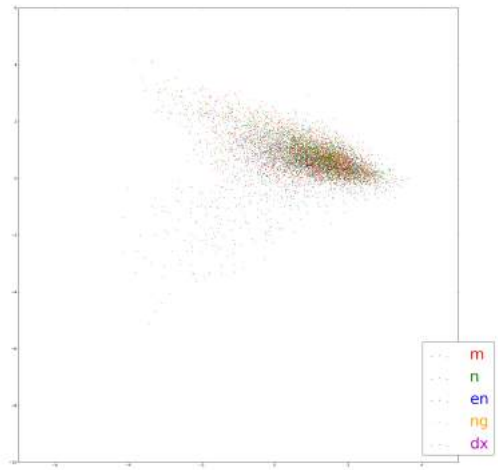
Figure A.4: *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of the first (left column) and the (3^{rd} hidden layer (right column) activations from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure. The visualisation of the 3^{rd} layer is less fuzzy than that of the 1^{st} layer.*



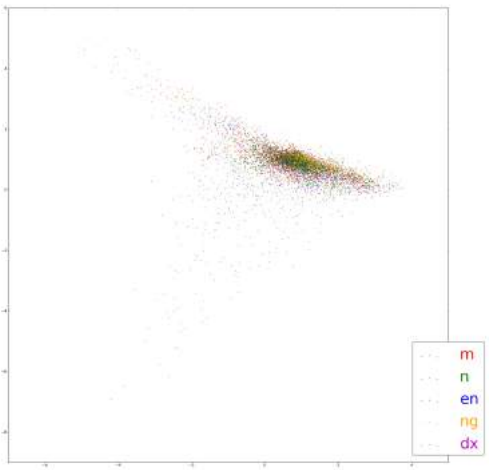
(a) Strong Fricative (1^{st} Hidden Layer)



(b) Strong Fricative (3^{rd} Hidden Layer)

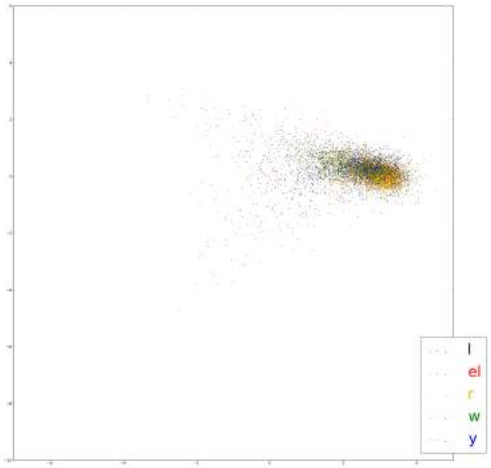


(c) Nasal,Flap (1^{st} Hidden Layer)

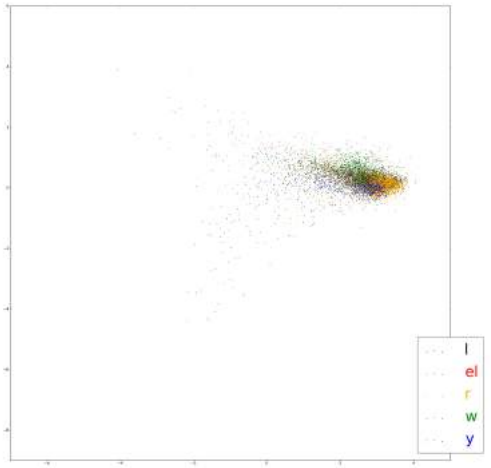


(d) Nasal,Flap (3^{rd} Hidden Layer)

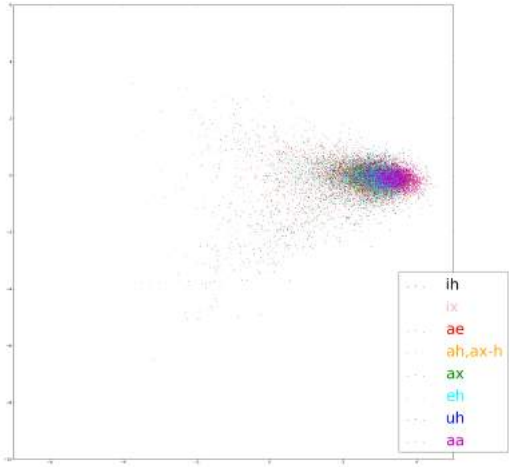
Figure A.4 (cont.): *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of the first (left column) and the (3^{rd} hidden layer (right column) activations from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure. The visualisation of the 3^{rd} layer is less fuzzy than that of the 1^{st} layer.*



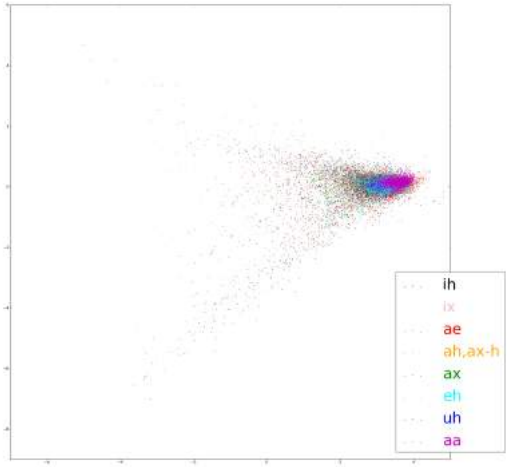
(e) Semi-vowel (1^{st} Hidden Layer)



(f) Semi-vowel (3^{rd} Hidden Layer)

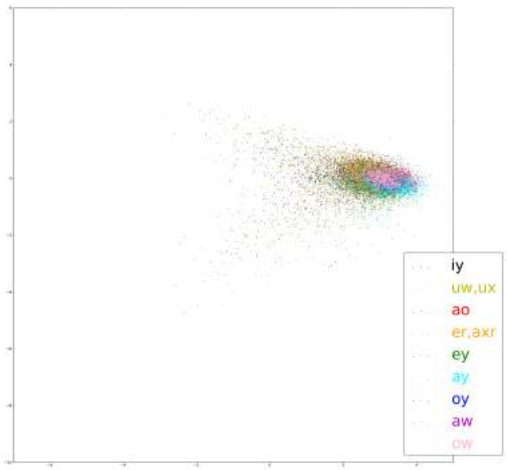


(g) Short Vowel (1^{st} Hidden Layer)

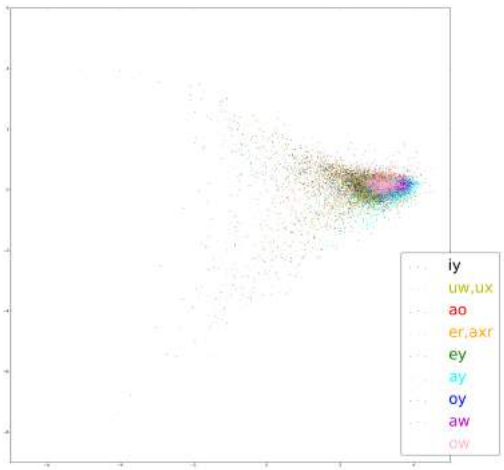


(h) Short Vowel (3^{rd} Hidden Layer)

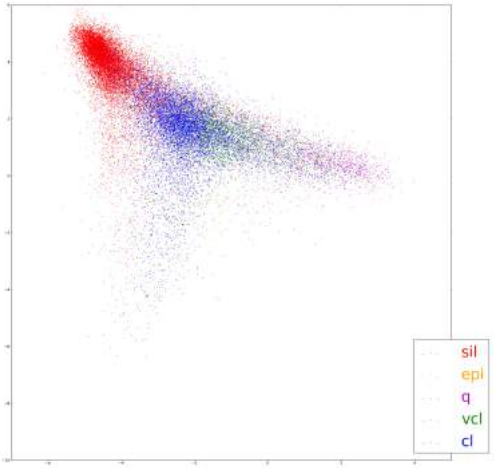
Figure A.4 (cont.): *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of the first (left column) and the (3^{rd} hidden layer (right column) activations from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure. The visualisation of the 3^{rd} layer is less fuzzy than that of the 1^{st} layer.*



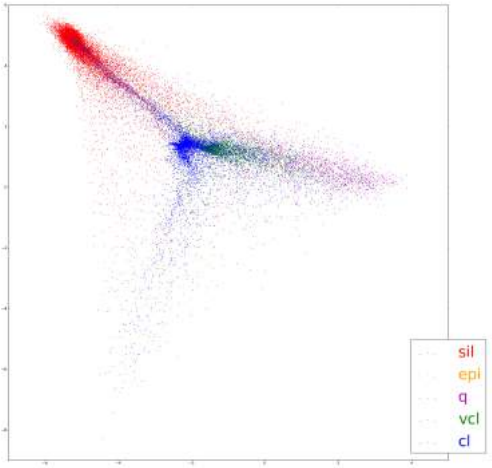
(i) Long Vowel (1^{st} Hidden Layer)



(j) Long Vowel (3^{rd} Hidden Layer)

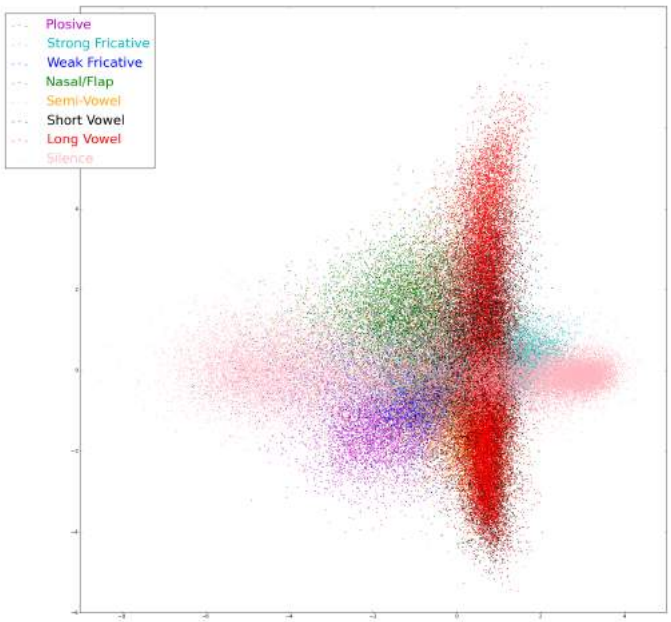


(k) Silence (1^{st} Hidden Layer)

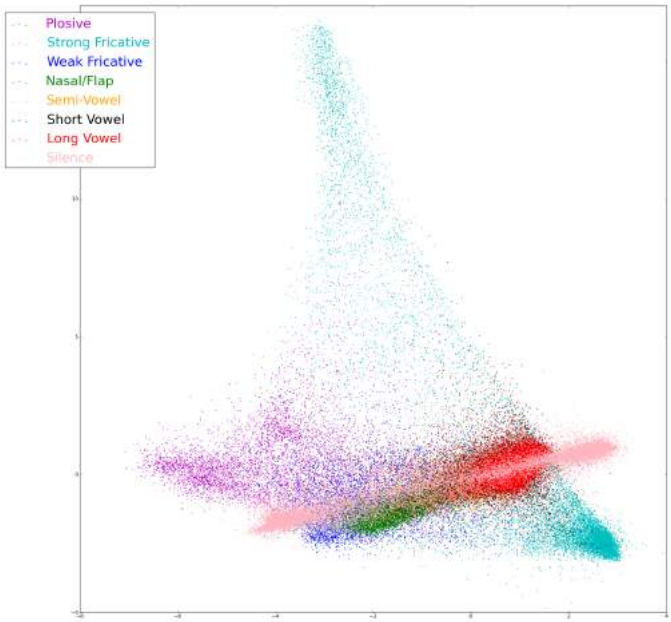


(l) Silence (3^{rd} Hidden Layer)

Figure A.4 (cont.): *Visualisations of LDA-based projections (1^{st} vs. 2^{nd} dimension) of the first (left column) and the (3^{rd} hidden layer (right column) activations from a phone classification DNN of structure 286-512-9-512-49. Plot on one phone category in each figure. The visualisation of the 3^{rd} layer is less fuzzy than that of the 1^{st} layer.*

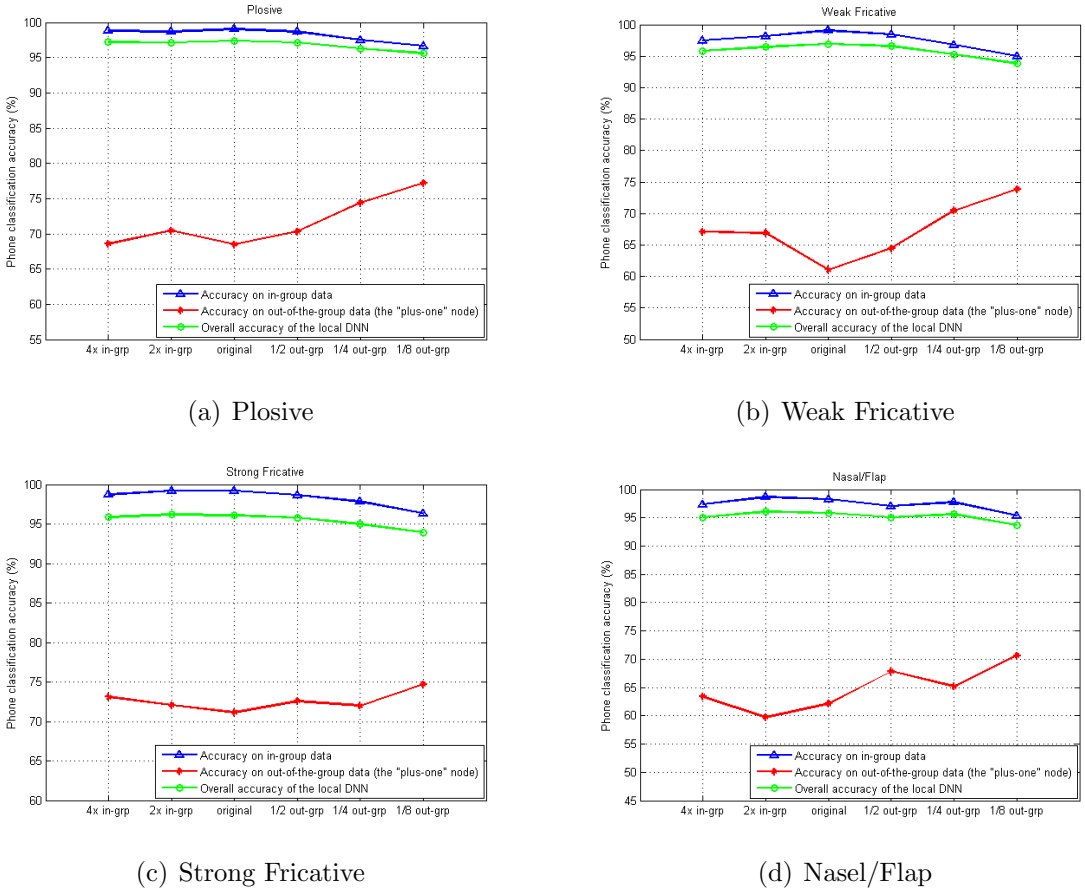


(m) 3^{rd} vs. 4^{th} dimension, 1^{st} hidden layer



(n) 3^{rd} vs. 4^{th} dimension, 3^{rd} hidden layer

Figure A.5: *Visualisations of LDA-based projections (3^{rd} vs. 4^{th} dimension) of the 1^{st} (a) and the 3^{rd} (b) hidden layer activations from a phone classification DNN of structure 286-512-9-512-49.*



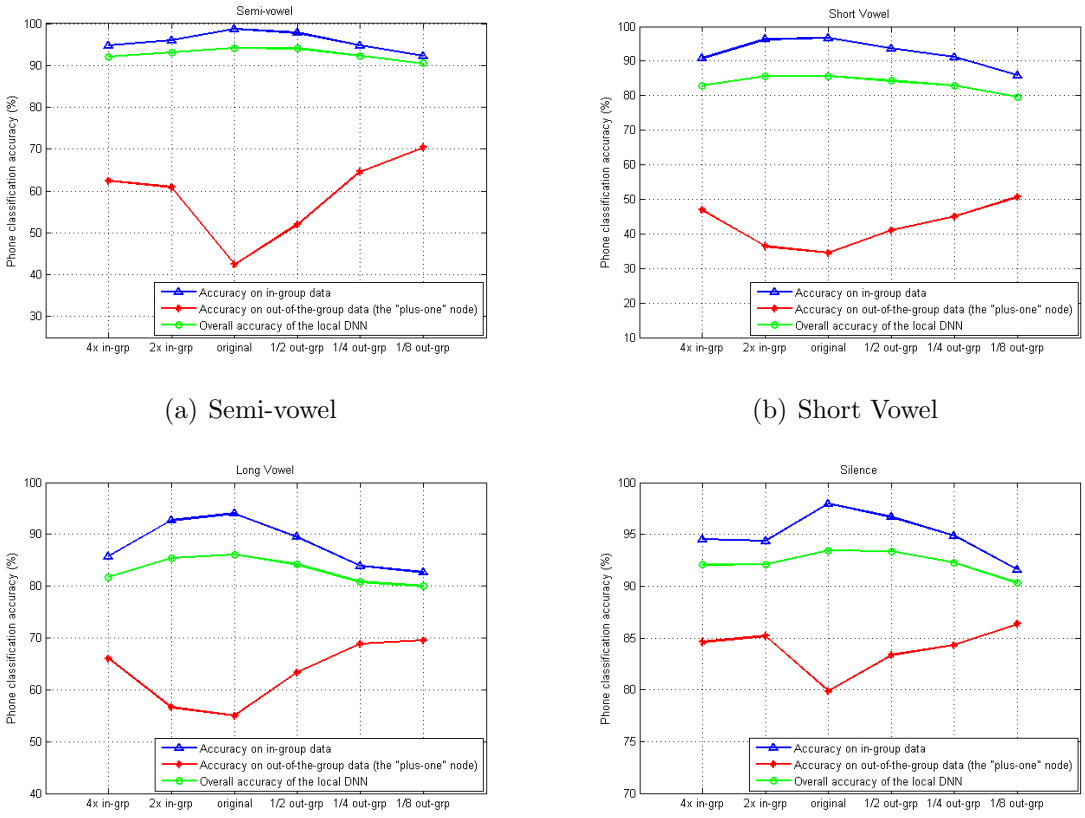
(a) Plosive

(b) Weak Fricative

(c) Strong Fricative

(d) Nasel/Flap

Figure A.6: *Phone/class classification accuracy of local DNNs when varying the ratio of in/out group data. Increasing the proportion of in-class data when training the BPC-dependent DNNs (from the "original" point to both directions along the horizonal axis) usually improves their abilities of classifying the in-class data (red plots), however the performance on all frames gets worse (green plots) in most cases. It seems that the benefit of a DNN better at classifying in-class data is not enough to counterweigh the loss of the DNN worse at classifying in-class data.*



(a) Semi-vowel

(b) Short Vowel

(c) Long Vowel

(d) Silence

Figure A.6 (cont.):*Phone/class classification accuracy of local DNNs when varying the ratio of in/out group data. Increasing the proportion of in-class data when training the BPC-dependent DNNs (from the "original" point to both directions along the horizonal axis) usually improves their abilities of classifying the in-class data (red plots), however the performance on all frames gets worse (green plots) in most cases. It seems that the benefit of a DNN better at classifying in-class data is not enough to counterweigh the loss of the DNN worse at classifying in-class data.*

Bibliography

Ackley, D. H., G. E. Hinton, and T. J. Sejnowski (1985). A learning algorithm for Boltzmann machines. *Cognitive science 9*(1), 147–169.

Apostol, T. M. (1974). *Mathematical analysis; 2nd ed.* Addison-Wesley Series in Mathematics. Reading, MA: Addison-Wesley.

Bahl, L. and F. Jelinek (1975). Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition. *IEEE Transactions on Information Theory 21*(4), 404–411.

Bai, L., P. Jančovič, M. Russell, and P. Weber (2015). Analysis of a low-dimensional bottleneck neural network representation of speech for modelling speech dynamics. In *Proc. Interspeech,* Dresden, Germany, pp. 583–587.

Bai, L., P. Jančovič, M. Russell, P. Weber, and S. Houghton (2017). Phone classification using a non-linear manifold with broad phone class dependent DNNs. In *Proc. Interspeech,* Stockholm, Sweden, pp. 319–323.

Baker, J. K. Stochastic modeling for automatic speech recognition.

Bastien, F., P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio (2012). Theano: new features and speed improvements. *Deep Learning and Unsupervised Feature Learning NIPS Workshop*.

Baum, L. E. and J. A. Eagon (1967). An inequality with applications to statistical

estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society 73*(3), 360–363.

Bergstra, J., O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio (2010, June). Theano: a CPU and GPU math expression compiler. In *Proc. of the Python for Scientific Computing Conference (SciPy)*.

Bilmes, J. A. et al. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute 4*(510), 126.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford, UK: OUP.

Bishop, C. M. et al. (2006). *Pattern recognition and machine learning*, Volume 1. springer New York.

Boersma, P. and D. Weenik (Retrieved January 29, 2013). Praat: Doing phonetics by computer (version 5.3. 39) [computer program].

Bourlard, H. and N. Morgan (1993). Continuous speech recognition by connectionist statistical methods. *IEEE Transactions on Neural Networks 4*(6), 893–909.

Champion, C. J. and S. M. Houghton (2015). Application of Continuous State Hidden Markov Models to a classical problem in speech recognition. *Computer Speech and Language,* doi:10.1016/j.csl.2015.05.001.

Czech Technical University in Prague (2008). Examples: Statistical pattern recognition toolbox. `https://cmp.felk.cvut.cz/cmp/software/stprtool/examples.html`.

Denes, P. and E. Pinson (1993). *The speech chain* (2nd ed.). New York: W.H. Freeman and Company.

Deng, L. (2006). *Dynamic Speech Models, Theory, Algorithms, and Applications*. Morgan & Claypool.

Deng, L. and J. Chen (2014). Sequence classification using the high-level features extracted from deep neural networks. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Florence, Italy, pp. 6844–6848.

Deng, L., G. Hinton, and B. Kingsbury (2013). New types of deep neural network learning for speech recognition and related applications: An overview. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Vancouver, BC, Canada, pp. 8599–8603. IEEE.

Deng, L. and J. Ma (2000). Spontaneous speech recognition using a statistical coarticulatory model for the vocal-tract-resonance dynamics. *J. Acoust. Soc. Am. 108*(6), 3036–3048.

Doddipatla, R. (2016). Speaker adaptive training in deep neural networks using speaker dependent bottleneck features. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* New Orleans, Louisiana, USA, pp. 5290–5294. IEEE.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics 7*(2), 179–188.

Gales, M. J. F. and S. J. Young (1993). Segmental hidden Markov models. In *Proc. Eurospeech '93,* Berlin, Germany, pp. 1579–1582.

Garofolo, J., L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue (1993). *TIMIT Acoustic-Phonetic Continuous Speech Corpus.* Univ. Pennsylvania, Philadelphia, PA: Linguistic Data Consortium.

Gers, F. A., J. Schmidhuber, and F. Cummins (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation 12*(10), 2451–2471.

Gillick, L. and S. J. Cox (1989). Some statistical issues in the comparison of speech recognition algorithms. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Glasgow, Scotland, pp. 532–535. IEEE.

Golub, G. and W. Kahan (1965). Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis 2*(2), 205–224.

Graves, A. et al. (2012). *Supervised sequence labelling with recurrent neural networks*, Volume 385. Springer.

Graves, A., S. Fernández, F. Gomez, and J. Schmidhuber (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 369–376. ACM.

Graves, A. and N. Jaitly (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772.

Grézl, F., M. Karafiát, S. Kontár, and J. Cernocky (2007). Probabilistic and bottleneck features for LVCSR of meetings. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Honolulu, Hawai, Volume 4, pp. 757–760. IEEE.

Halberstadt, A. K. and J. R. Glass (1997). Heterogeneous acoustic measurements for phonetic classification 1. In *Proc. Eurospeech '93,* Berlin, Germany, pp. 401–404.

Heinz, J. M. and K. N. Stevens (1961). On the properties of voiceless fricative consonants. *The Journal of the Acoustical Society of America 33*(5), 589–596.

Henter, G. E., M. R. Frean, and W. B. Kleijn (2012). Gaussian Process dynamical models for nonparametric speech representation and synthesis. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Kyoto, Japan, pp. 4505–4508.

Henter, G. E. and W. B. Kleijn (2011). Intermediate-state HMMs to capture

continuously-changing signal features. In *Proc. Interspeech,* Florence, Italy, pp. 1828–1831.

Hermansky, H., D. P. Ellis, and S. Sharma (2000). Tandem connectionist feature extraction for conventional HMM systems. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Istanbul, Turkey, Volume 3, pp. 1635–1638.

Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine 29*(6), 82–97.

Hinton, G. E. (2006). Training products of experts by minimizing contrastive divergence. *Training 14*(8).

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in cognitive sciences 11*(10), 428–434.

Hinton, G. E. (2012). A practical guide to training restricted Boltzmann machines. In *Neural networks: Tricks of the trade*, pp. 599–619. Springer.

Hinton, G. E., S. Osindero, and Y.-W. Teh (2006). A fast learning algorithm for deep belief nets. *Neural computation 18*(7), 1527–1554.

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

Holmes, J. and W. Holmes (2001). *Speech synthesis and recognition* (2nd ed.). London and New York: Taylor and Francis.

Holmes, J. N., W. J. Holmes, and P. N. Garner (1997). Using formant frequencies in speech recognition. In *Proc. Eurospeech '97,* Rhodes, Greece, pp. 2083–2086.

Holmes, J. N., I. G. Mattingly, and J. N. Shearme (1964). Speech synthesis by rule. *Language & Speech 7*, 127–143.

Holmes, W. J. (1997). *Modelling Segmental Variability for Automatic Speech Recognition.* Ph. D. thesis, University of London.

Hori, T., S. Watanabe, Y. Zhang, and W. Chan (2017). Advances in joint CTC-Attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. In *Proc. Interspeech 2017*, pp. 949–953.

Huang, H., Y. Liu, L. ten Bosch, B. Cranena, and L. Boves (2016). Locally learning heterogeneous manifolds for phonetic classification. *Computer Speech and Language 38*, 28–45.

Jiang, B., Y. Song, S. Wei, J.-H. Liu, I. V. McLoughlin, and L.-R. Dai (2014). Deep bottleneck features for spoken language identification. *PloS one 9*(7), e100795.

Johns, D. (1975). *An outline of English phonetics* (9th ed.). Cambridge Univ. Press.

Karpathy, A., J. Johnson, and F. Li (2015). Visualizing and understanding recurrent networks. *CoRR abs/1506.02078*.

Khasanova, A., J. Cole, and M. Hasegawa-Johnson (2014). Detecting articulatory compensation in acoustic data through linear regression modeling. In *Proc. Interspeech,* Singapore.

Kreyszig, E. (2000). *Advanced Engineering Mathematics: Maple Computer Guide* (8th ed.). New York, NY, USA: John Wiley & Sons, Inc.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature 521*(7553), 436–444.

Lee, J. (2010). *Introduction to topological manifolds* (Second ed.). Graduate texts in Mathematics 202. New York Dordrecht Heidelberg London: Springer.

Lee, K.-F. and H.-W. Hon (1989). Speaker-independent phone recognition using hidden Markov models. *IEEE Trans. on Acoustics, Speech and Signal Processing 37*(11), 1641–1648.

Li, F., A. Menon, and J. Allen (2010). A psychoacoustic method to find the perceptual cues of stop consonants in natural speech. *The Journal of the Acoustical Society of America 127*(4), 2599–2610.

Li, F., A. Trevino, A. Menon, and J. B. Allen (2012). A psychoacoustic method for studying the necessary and sufficient perceptual cues of American English fricative consonants in noise. *The Journal of the Acoustical Society of America 132*(4), 2663–2675.

Liu, D., S. Wei, W. Guo, Y. Bao, S. Xiong, and L. Dai (2014). Lattice based optimization of bottleneck feature extractor with linear transformation. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Florence, Italy.

Lu, L., X. Zhang, K. Cho, and S. Renals (2015). A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition. pp. 3249–3253.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*(4), 115–133.

Nagamine, T., M. L. Seltzer, and M. N (2015). Exploring how deep neural networks form phonemic categories. In *Interspeech*, pp. 1912–1916.

Ostendorf, M., V. V. Digalakis, and O. A. Kimball (1996). From HMM's to segmental models: a unified view of stochastic modeling for speech recognition. *IEEE Trans. on Spch. & Aud. Proc. 4*(5), 360–378.

Petridis, S. and M. Pantic (2016). Deep complementary bottleneck features for visual speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* New Orleans, Louisiana, USA.

Pundak, G. and T. N. Sainath (2017). Highway-LSTM and recurrent highway networks for speech recognition. In *Proc. Interspeech 2017*, pp. 1303–1307.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*(2), 257–286.

Rao, C. R. (1948). The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological) 10*(2), 159–203.

Raphael, L. (1972). Preceding vowel duration as a cue to the perception of the voicing characteristic of word-final consonants in American English. *The Journal of the Acoustical Society of America 51*(4B), 1296–1303.

Richards, H. B. and J. S. Bridle (1999). The HDM: a segmental Hidden Dynamic Model of coarticulation. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Phoenix, AZ, pp. 357–360.

Rosenblatt, F. (1962). Principles of neurodynamics.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR abs/1609.04747*.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

Sak, H., A. Senior, and F. Beaufays (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Scanlon, P., D. Ellis, and R. Reilly (2007). Using broad phonetic group experts for improved speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing 15*(3), 803–812.

Schmidhuber, J. (2017). Recurrent neural networks. `http://people.idsia.ch/~juergen/rnn.html`.

Shoup, J. E. (1980). Phonological aspects of speech recognition. *Trends in Speech Recognition*, 125–138.

Sjölander, K. and J. Beskow (2000). Wavesurfer - an open source speech tool. In *Interspeech*, pp. 464–467.

Stevens, K. and S. Blumstein (1978). Invariant cues for place of articulation in stop consonants. *The Journal of the Acoustical Society of America 64*(5), 1358–1368.

Stevens, K. N. (2002). Toward a model for lexical access based on acoustic landmarks and distinctive features. *The Journal of the Acoustical Society of America 111*(4), 1872–1891.

Tan, S., K. C. Sim, and M. Gales (2015). Improving the interpretability of deep neural networks with stimulated learning. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pp. 617–623. IEEE.

University of California Berkeley (2011). Voices of berkeley. `http://voicesof.berkeley.edu/vowel_space.php`.

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

Wattenberg, M., F. Vigas, and I. Johnson (2016). How to use t-SNE effectively. *Distill*.

Weber, P., L. Bai, S. Houghton, P. Jančovič, and M. Russell (2016). Progress on phoneme recognition with a Continuous-State HMM. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* New Orleans, Louisiana, USA.

Weber, P., L. Bai, M. Russell, P. Jančovič, and S. Houghton (2016). Interpretation of low dimensional neural network bottleneck features in terms of human perception and production. In *Proc. Interspeech,* San Francisco, CA, USA, pp. 3384–3388.

Weber, P., S. M. Houghton, C. J. Champion, M. J. Russell, and P. Jančovič (2014). Trajectory analysis of speech using Continuous-State Hidden Markov Models. In

Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, pp. 3042–3046.

Wilde, L. (1995). *Analysis and synthesis of fricative consonants.* Ph. D. thesis, Massachusetts Institute of Technology.

Yang, L. (2007). Data embedding research. `https://cs.wmich.edu/~yang/research/dembed/`.

Young, S. J., J. Odell, D. Ollason, V. Valtchev, and P. Woodland (1997). *The HTK Book* (v2.1 ed.). Cambridge, UK: Entropic Camb. Res. Lab.

Yu, D. and L. Deng (2014). *Automatic speech recognition: A deep learning approach.* Springer.

Zeiler, M. D. and R. Fergus (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer.

Zhang, Y., W. Chan, and N. Jaitly (2017). Very deep convolutional networks for end-to-end speech recognition. IEEE.