Rijuban Rangslang

Segment phoneme classification from speech under noisy conditions:Using amplitude-frequency modulation based two-dimensional auto-regressive features with deep neural networks

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 03.08.2016

Thesis supervisor:

Prof. Paavo Alku

Thesis advisor:

Dhanajaya Gowda(Ph.D.)

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Rijuban Rangslang

Title: Segment phoneme classification from speech under noisy conditions:Using amplitude-frequency modulation based two-dimensional auto-regressive features with deep neural networks

Date: 03.08.2016 Language: English Number of pages: 6+64

Department of Signal Processing and Acoustics

Professorship: Speech communication technology Code: S-89

Supervisor: Prof. Paavo Alku

Advisor: Dhanajaya Gowda(Ph.D.)

This thesis investigates at the acoustic-phonetic level the noise robustness of features derived using the AM-FM analysis of speech signals. The analysis on the noise robustness of these features is done using various neural network models and is based on the segment classification of phonemes. This analysis is also extended and the robustness of the AM-FM based features is compared under similar noise conditions with the traditional features such as the Mel-frequency cepstral coefficients(MFCC).

We begin with an important aspect of segment phoneme classification experiments which is the study of architectural and training strategies of the various neural network models used. The results of these experiments showed that there is a difference in the training pattern adopted by the various neural network models. Before over-fitting, models that undergo pre-training are seen to train for many epochs more than their opposite models that do not undergo pre-training. Taking this difference in training pattern into perspective and based on phoneme classification rate the Gaussian restricted Boltzmann machine and the single layer perceptron are selected as the best performing model of the two groups, respectively.

Using the two best performing models for classification, segment phoneme classification experiments under different noise conditions are performed for both the AM-FM based and traditional features. The experiments showed that AM-FM based frequency domain linear prediction features with or without feature compensation are more robust in the classification of 61 phonemes under white noise and 0 dB signal-to-noise ratio(SNR) conditions compared to the traditional features. However, when the phonemes are folded to 39 phonemes, the results are ambiguous under all noise conditions and there is no unanimous conclusion as to which feature is most robust.

Keywords: Robust speech recognition system, AM-FM based features, Neural networks

Preface

This thesis work is undertaken for fulfillment of the requirements of the Master's Degree Program in Communications Engineering at Aalto University School of Electrical Engineering, Finland. The task was performed at the Department of Signal Processing and Acoustics, School of Electrical Engineering, Aalto University.

First and foremost I would like to express my sincere gratitude to Prof. Paavo Alku for giving me an opportunity to work with him and his team. Secondly, I would like express my gratitude to my instructor Dhananjaya Gowda for all the encouragement and advice he has given me throughout the course of the thesis. I would also like to thank Bajibabu Bollepali for giving me valuable advice pertaining to the subject of neural networks. I would also like to thank all the staff members that are responsible in maintaining the triton accounts and also Tarmo Simonen for helping me out with all the IT related issues. I would also like to thank all my colleagues and friends who were part of the foosball games in the piano bar.

Lastly, I would like to thank my family especially my sister for being my constant source of inspiration and support these last couple of years.

Contents

Symbols and abbreviations

$H_j(\boldsymbol{x})$	Neural network posterior probability at output unit j for input $\boldsymbol{x}$
$E(v, h)$	Energy function of a joint configuration of visible and hidden units
E_{AE}	Error function of an auto-encoder
E_{sparse}	Error function of a sparse auto-encoder
E_{CAE}	Error function of a contractive auto-encoder
$H(z)$	Transfer function of a simple resonator
$A_i[n]$	Amplitude modulation component of the i^{th} narrowband signal
$Error(h, S)$	Segment phoneme classification error

RASTA	Relative spectra
MLLR	Maximum likelihood linear regression
MAP	Maximum a-posterior
MCE	Minimum classification error
MMI	Maximum mutual information
LP	Linear prediction
ASR	Automatic Speech recognition
AM-FM	Amplitude modulation-frequency modulation
AM-FM FDLP	AM-FM Frequency domain linear prediction
AM-FM 2D AR	AM-FM two dimensional auto-regressive
MFCC	Mel-frequency cepsral coefficient
PLP	Perceptual linear prediction
MLP	Multi-layered perceptron
ReLU	Rectified linear unit
GRBM	Gaussian Restricted Boltzmann machine
GPU	Graphic processor units
PCA	Principal component analysis
DNN	Deep neural networks
GMM/HMM	Gaussian mixture models/Hidden Markov models
CMN	Cepstral mean normalization
CMVN	Cepstral mean variance normalization

1 Introduction

1.1 Overview

Over the last few decades automatic speech recognition system(ASR) has undergone considerable improvement. This has lead to widespread use of the technology in areas such as automated call centers, personalized assistant in mobile phones, hands free computing, speaker dependent recognition system in many home automated appliances, etc. However, the inability of the automatic speech recognition system to reach human level performance has restricted its coverage. This is especially true in areas such as defense systems where the margin of error is small.

The performance of the speech recognition system further aggravates under noisy conditions. Noise influences the speaker as well as the system. The noise corrupts the speech signal and also leads to the Lombard effect [1]. The Lombard effect results in changes to the pitch, the duration and amplitude of syllables. Most speech recognition system are inherently designed or trained under clean conditions [2]. Hence, such changes to the input speech leads to substantial degradation in the performance of the automatic speech recognition system.

Various speech enhancement techniques are used to reduce the effect of the noise. With additive noise speech enhancement methods include Wiener filtering [3], power bias subtraction [4] and missing data reconstruction [5]. For convolutive noise, methods such as feature warping [6], RASTA processing [7] and cepstral mean subtraction [8] are often used. As for acoustic models, models that are derivative of Gaussian mixture models and trained generatively are made to adapt to the noise environment using adaptation technique such the maximum likelihood linear regression(MLLR) [9] and maximum a posterior(MAP) [10]. Discriminative trained acoustic models on the other hand are adapted using minimum classification error(MCE) [11] and maximum mutual information(MMI) [12] .

Most of the speech enhancement technique mentioned work by assuming the type of noise and estimating the noise properties. Hence, these methods do not generalize well to other types of degradations [13]. Similarly with the models, the adaptation techniques mentioned are trained using data that represent a subset of the test set domain. But collecting a reasonable amount of data for all test conditions is not always possible. Thus, to ensure an all round robustness in the system, signal analysis techniques for feature extraction should concentrate on regions less affected by noise. Also, the system should adapt and generalize well to all test conditions even when trained using clean speech signal.

Over the years a considerable amount of research has been done to arrive at signal analysis techniques that are robust under various degradation conditions. These methods involve modeling the speech signal in terms of its envelope and phase modulation and are known as the AM-FM model of speech [14]. Improvements are reported in [15] for various speech recognition task when using features that are derivative of the auto-regressive modeling of the envelope of the speech signal.

From the model point of view, deep neural networks(DNN) have replaced Gaussian mixture models in the acoustic model of the ASR system. Such acoustic models

achieved considerable improvement compared to Gaussian mixture models with respect to various speech recognition task. Using deep neural networks have led to improvement in phoneme recognition as reported in [16]. Deep neural networks without any form of adaptation are also extended to modeling of robust acoustic models. This is achieved by a different training approach such as multi-condition training. The deep architecture allows the discovery of representations that are stable to variations in the input signal as discussed in [17]. Deep neural nets are also trained as de-noising auto-encoder and used for speech enhancement [18]. Overall there has been considerable improvement in the speech recognition system under noisy conditions using such deep neural networks.

1.2 Objective

The objective of the thesis is to study the robustness of the AM-FM based features under different noise conditions. This study uses various neural network models and is based on segment phoneme classification experiments. We begin with an important aspect of segment phoneme classification experiments which is the study of the functionality of various neural network models with regards to classification. The various neural network models used can be divided into two groups. The first group include networks which do not undergo any initial pre-training such as the single layer perceptron and the multi-layered perceptron(MLP) with rectified linear(REL) units that are integrated with a model averaging technique called the dropout. Also included are neural networks models that are initially pre-trained as stacks of restricted Boltzmann machine and auto-encoder before being fine-tuned as an MLP for classification.

After the study of the functionality of the various network models we choose based on the phoneme classification rate the best neural network models; one from each group. These neural networks models serves as network classification models that are used in studying the robustness of the AM-FM based features under different additive degradation. The degradation types included are the white, babble and factory noise with signal-to-noise(SNR) ratio of 0,10 and 20 dB. The study is also conducted using features that are compensated with RASTA filtering. Also, the robustness of the AM-FM based features with regards to phoneme classification is also compared under similar conditions with the traditional features. The AM-FM based features included are the AM-FM FDLP and the AM-FM 2D AR features. The traditional features used include mel-frequency cepstral coefficients(MFCC) and the perceptual linear prediction(PLP) features.

2 Introduction on feature extraction and pattern classification

2.1 Overview

The process of transcribing speech features to words in a speech recognition system happens at three levels. The first involve using various signal processing techniques for extracting speech features. Secondly, the acoustic model establishes at the acoustic-phonetic level the relationship between the features and the smallest linguistic unit namely, the phoneme. This in turn is used when representing larger speech units such as words or phrases. Finally, the language model decides on the sequence of words. In this chapter, we will review the front-end method of feature extraction. We will also look at an important aspect of the acoustic model which is the mapping of the features to the respective phonemes.

2.2 Feature Extraction

The first step in an automatic speech recognition system involves pre-processing of the speech signal and extracting discriminative features. These discriminative features are representative of the speech characteristics that are stable over time and remain unaffected by reasonable background noise. These features also represent speech characteristics that are discriminative between speaker while being tolerant to intra-speaker variability such as health and emotion. Most state of the art features do not incorporate all of these speech characteristics. For research and certain practical conditions, features that partially represent these speech characteristic are considered [19]. The most popular features include the mel-frequency cepstral coefficients(MFCC) and the perceptual linear predictions(PLP) features. Here, to give an example to the generation of speech feature we will discuss the generation of the MFCC features.

Speech is the sound generated as it gets filtered through the vocal tract including the tongue and teeth. The vocal tract manifest itself in terms of the short-time power spectrum envelope. MFCC represent to a certain accuracy the envelope of the power spectrum [20]. There are usually 39 dimensions in the MFCC feature vector. These include 12 static features, 1 energy feature, the first and second order derivative of the static features [21]. The procedure for extraction of MFCC is summarized as follows.

1. Pre-emphasis :- The spectrum associated with speech signal decreases at a rate of $-20dB$ per decade. Thus, high frequency regions of the speech signal have smaller amplitude [22]. Pre-emphasis is intended to offset this natural slope of the speech spectrum by flattening out the spectrum of the speech signal [23]. Pre-emphasis is achieved by filtering the speech signal through a finite impulse

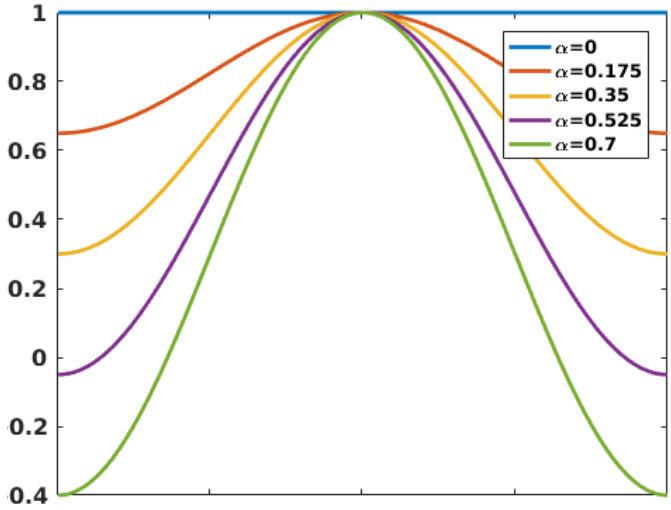


Figure 1: Hamming window

response(FIR) filter defined as shown below

$$H_{pre}(z) = 1 + az^{-1} \tag{1}$$

where the typical range for a is $[-1.0, -0.4]$.

2. Framing :- The human speech signal represent a slowly time-varying signal [21]. Under short time frames, it is treated as a stationary process [24]. This treatment of the speech signal as a stationary time process is essential for the spectral analysis of the speech signal. Normally, the frame size is considered to be $20-40ms$ with an frame step of $10ms$. If we consider a speech signal sampled at 16000 Hz, a frame size of $25\,ms$ will result in a frame of $0.025 \times 16000 = 400$ samples. A frame-step of $10\,ms$ will result in 160 samples. There is an overlap with the first frame extending from 0-400 samples, while the next frame starts at 160 and extend for another 400 samples.

3. Window:- Windowing serves as a necessary pre-cursor before the spectral analysis using Fourier transform. The Fourier transform assumes a finite set of data that is one period of a periodic signal. Thus, there is a circular topology with the beginning and the end point of the signal being connected. However, if such conditions are not met and the signal is discontinuous, these discontinuities show up as high frequency component in the Fourier transform [25]. To prevent this, the signal is filtered using a window function. The most commonly used window in speech signal analysis is the Hamming window which is defined by $w(n)$ as shown below:

$$w(n, \alpha) = (1 - \alpha) - \alpha cos(2\pi n/(N-1)), 0 < n < N-1 \tag{2}$$

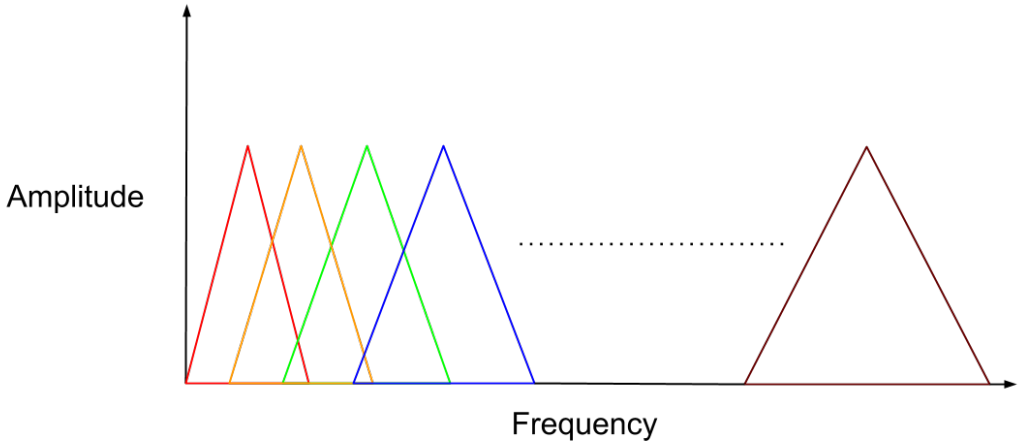


Figure 2: A sketch of the mel-filter bank

where different values of α correspond to different curves for the Hamming window and the window length is $N + 1$

4. Discrete Fourier transform to each frame :- After windowing, the samples present in each frame are converted from the time domain to the frequency domain. This spectral analysis of each frame is achieved using the fast Fourier transform(FFT). These frequency domain coefficients are complex numbers that are representative of the magnitude and the phase of the speech signal. In the case of the MFCC, only the magnitude of the spectral coefficients is considered [21].

5. Mel filter-banks:- Human hearing is not equally sensitive to all frequency bands. To incorporate this non-linear perception of frequency, the spectrum of the speech signal is filtered by a group of triangular filter-banks. These mel filter-banks are arranged on a mel-frequency scale. The relation between mel-scale frequencies to the linear scale frequencies(Hertz) is given by

$$f_{mel} = 2595 \ log(1 + f/700) \tag{3}$$

where f_{mel} and f is the frequency in the mel-scale and linear scale, respectively. On the mel-scale the windows are evenly distributed. The bandwidth of such windows are narrow at low frequencies and gradually increases as it approach the higher frequencies regions. In order to get an estimate of the amount of energy in each filter bank, the spectrum is multiplied with the triangular filter and the coefficients in each of the filter bank are added [21].

6. Applying the natural logarithm:- The human auditory system also exhibit non-linear loudness characteristics. This is approximated in the feature extraction method by considering the logarithmic scale. Also, using this logarithm scale

ensures that the convolutive distortion is additive and can be removed using simplified speech enhancement techniques.

7. Discrete cosine transform :- The discrete cosine transform is meant to de-correlate and convert the log-mel spectrum back to the time domain. This time domain representation represent the MFCC features. Only a fraction of the de-correlated Mel coefficient are considered for feature extraction.

8. Log energy :- The Log energy is calculated directly from the time domain. It represent the energy of each frame. The log energy sometimes replaces the 0^{th} cepstral coefficient for various speech recognition task.

9. Delta and the delta-deltas:- The MFCC coefficients described so far represents only the power spectral envelope of the speech signal [20]. However, the speech signal also has information that is loaded in the trajectories of these coefficients over a period of time. These trajectories are calculated in terms of the delta coefficients that are defined as

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{4}$$

where d_t is the delta coefficient computed from frame t. It is computed in terms of the static cepstral coefficients c_{t+n} to c_{t-n}. A typical value of $N = 2$. Delta-deltas coefficients are calculated using the same formula as shown above, replacing the static coefficients with the delta coefficients. These dynamic speech features are appended to the static features and used in various speech recognition problems.

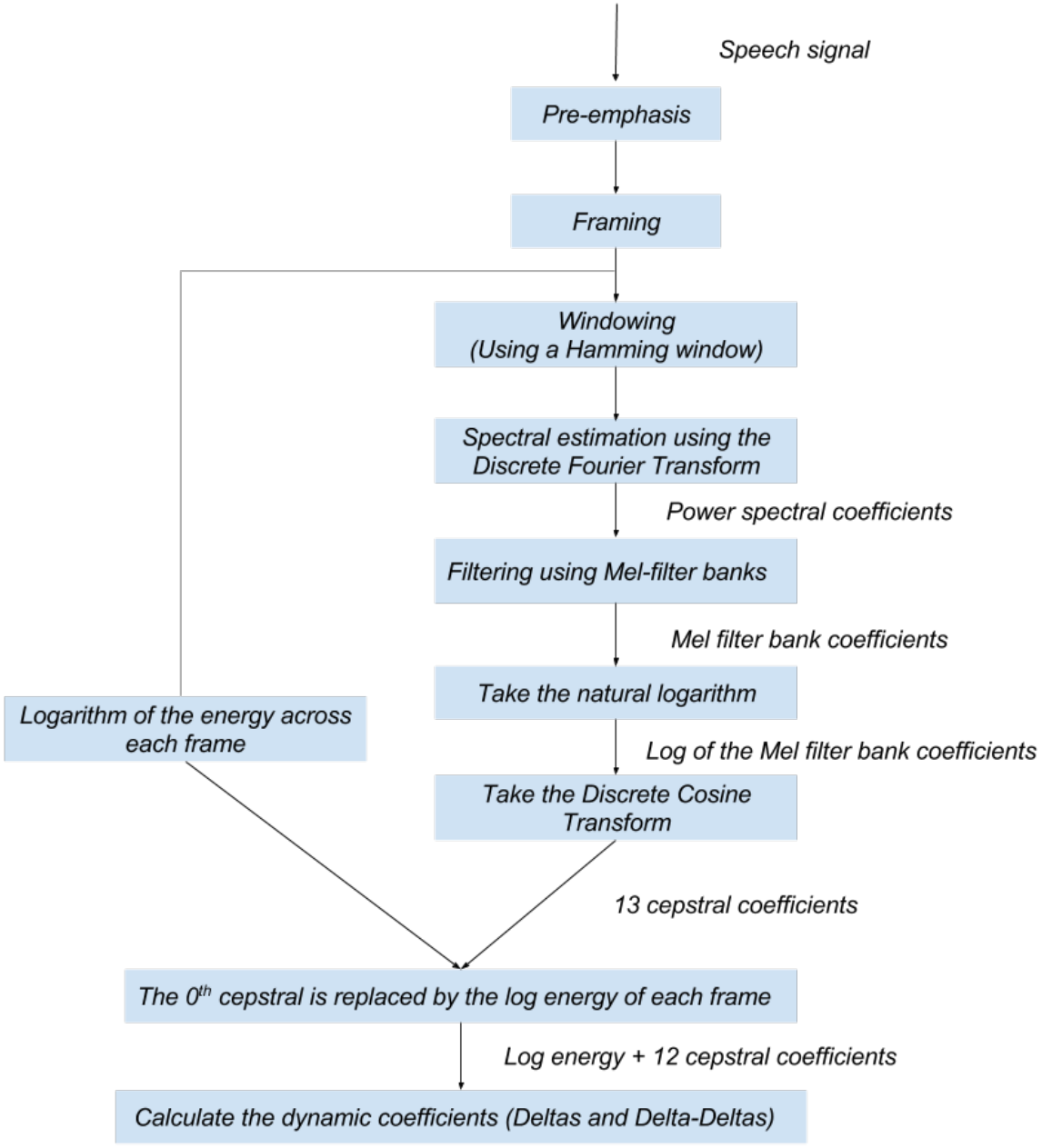


Figure 3: Flowchart showing the step by step process in the extraction of MFCC features

2.3 Statistical pattern classification

Since the start of speech recognition research many methods for transcribing speech to words have been proposed. These include template matching and statistical methods based techniques. In template matching techniques a prototype or a pattern of the speech utterance(usually of a 2-D shape) is available. Template matching involves a direct comparison between the unknown pattern and the training pattern taking into account all available pose such as translation, rotation and scale changes [26]. Statistical methods on the other hand include well formulated probabilistic mathematical models. These mathematical models are particularly suitable for speech recognition system as they can model the uncertainty or incomplete information that can arise from contextual effects,confusable sound etc. [27]. Statistical methods such as the Hidden Markov Model continue to be in use even today delivering commendable results on many aspect of speech recognition.

An important aspect of speech recognition considered in this thesis is that of phoneme classification. Phoneme classification is described as the task of tagging a speech utterance to its respective phoneme. Various statistical methods are considered for phoneme classification. The Elena project represent one such study where using real world datasets statistical methods such as the neural networks are used for task of phoneme classification [28].

In the coming section we give an overview of various classification methods. We begin with the Bayesian classification methods. We then proceed to discussing neural networks and also deep neural networks based classification.

2.3.1 Bayesian Classification

The Bayesian classification approach forms the basis of many statistical classification methods. Based on the objective of establishing decision boundaries Bayesian classification methods can be sub-divided into two section. The probabilistic and the discriminant analysis classification approach. In the probabilistic approach, the class conditional probability distribution of a pattern is estimated. Then a discriminant function that specifies the decision boundaries is established. A number of decision rules is associated with the probabilistic approach. These include the Bayes decision rule, the maximum likelihood and the Neyman-Pearson rule. Most of the decision rule are derivatives of the Bayes decision rule but have distinct training strategies [28].

The discriminant analysis approach establishes the decision boundaries directly and is supported by Vapnik's philosophy. Vapnik's philosophy states that it is better to solve the problem directly rather than considering an intermediate step especially if there exist a limited amount of information for solving the problem. The discriminant analysis approach follows a parametric form of the decision boundary. The parameters of the decision boundary are estimated by minimizing a cost function [26]. The cost function can be an error criteria such as the mean square error between the true class assignment values and the predicted class values.

In the coming section we begin with an explanation of the probabilistic approach and consider the Bayes decision rule and the likelihood decision rule. Then we proceed to explaining discriminant analysis approach for classification.

The Bayes decision rule approach can be summarized as follows: Consider that a given feature vector $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ of d dimensions that needs to be assigned to one of the c categories $w_1, w_2, ..., w_c$. If $P(w_j)$ is the prior probability of group j and $P(x|w_j)$ is the probability density function . Then as per Bayes' rule

$$P(w_j|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|w_j)P(w_j)}{f(\boldsymbol{x})} \tag{5}$$

where $P(w_j|\boldsymbol{x})$ is the posterior probability of group j and the marginal distribution $f(\boldsymbol{x}) = \sum_{i=1}^{C} f(\boldsymbol{x}|w_j)P(w_j)$.

The training step for the Bayes decision rule involves estimating the class conditional densities $P(\boldsymbol{x}|w_j)$ from the training data. A classic parametric approach is to model the class conditional densities as multivariate normal distributions or as a mixture of some standard probability densities. Hence, the training step involves estimating the parameters of the density function using the maximum likelihood approach or using the expectation maximization algorithm for the mixture models. In the non-parametric case, the k-NN method represents a popular non-parametric density estimation. After estimating the parameters of the class conditional densities and assuming that prior for each class $P(w_j)$ is known, the Bayes decision rule assigns the input feature vector $\boldsymbol{x}$ to a class w_i with the minimum misclassification risk. The class dependent misclassification risk is defined as shown below

$$R(w_i|\boldsymbol{x}) = \sum_{j=1}^{c} L(w_i, w_j) * P(w_j|\boldsymbol{x}) \tag{6}$$

where $L(w_i, w_j)$ is the loss incurred in deciding w_i when the true class is w_j. Thus as per the Bayes decision rule, feature vector $\boldsymbol{x}$ is assigned to a class w_i when

$$R(w_i|\boldsymbol{x}) = min_{i=1,2,...C} R(w_i|\boldsymbol{x}) \tag{7}$$

In the case of the 0/1 loss function where each class misclassification is assumed equal then

$$L(w_i, w_j) = \begin{cases} 0, & \text{if} \quad i = j \\ 1, & \text{if} \quad i \neq j \end{cases}$$

The Bayes decision rule simplify to equation(8) with the feature vector $\boldsymbol{x}$ assigned to class w_i when

$$\begin{gathered} P(w_i|\boldsymbol{x}) > P(w_j|\boldsymbol{x}) \quad for \quad all \quad j \neq i \\ or \\ P(\boldsymbol{x}|w_i)P(w_i) > P(\boldsymbol{x}|w_j)P(w_j) \end{gathered} \tag{8}$$

This represent the optimal Bayes decision rule with the vector $\boldsymbol{x}$ assigned to a class with the maximum posterior probability.

For the explanation of the maximum likelihood decision rule, we again assume that the feature vector $\boldsymbol{x}$ needs to be assigned to one of the c class $w_1, w_2, ..., w_c$. The likelihood decision rule is an extension of the Bayes rule with a 0/1 loss function. It also makes an assumption that the prior $P(w_j)$ across all the classes are equal. Thus, the decision rule allocates the feature vector $\boldsymbol{x}$ according to equation(8) with the decision rule dependent on the posterior probability [29]. The training step for the maximum likelihood rule aims to estimate this posterior probability directly. For a parametric maximum likelihood method such as the linear or logistic regression, the parameters that can define the posterior probability are estimated using the gradient descent method with a least square or cross entropy criterion fitting function. The cross entropy criteria is preferred as it considers a binomial distributed error. Also, the categorical cross entropy is extended to multi-class classification problems with the posterior probability modeled as a soft-max function

Discriminant analysis method makes no assumption on the conditional class probability density. The Fisher linear discriminant analysis method as described in [30]represent one such method. The Fisher linear discriminant finds a linear projection $w^T\boldsymbol{x}$ for the training data $\boldsymbol{x}$, with w being the linear transformation matrix. This linear function aims to maximizes the ratio of the between class separation to the within class separation and is thus derived by maximizing the following objective function

$$J(w) = \underset{w}{\text{argmax}} \quad \frac{w^T S_B w}{w^T S_W w} \tag{9}$$

where S_B is the between class scatter matrix" and S_W is the "within class scatter matrix". The definition of the scatter matrix are

$$\begin{aligned} S_B &= \sum_i (\mu_i - \hat{\boldsymbol{x}})(\mu_i - \hat{\boldsymbol{x}})' \\ S_W &= \sum_i \sum_{j \epsilon c} (x_j - \mu_i)(x_j - \mu_i)^T \end{aligned} \tag{10}$$

where $\hat{\boldsymbol{x}}$ is the overall mean of the data.

For a feature vector $\boldsymbol{x}$ that is to be classified to one of the c classes $w_1, w_2, ..., w_c$. The average score $w^T\mu_c$ is calculated for each of the class $i = 1, ...c$.The decision rule assigns the feature vector x to class w_j when

$$|w^T\hat{\boldsymbol{x}} - w^T\mu_i| < |w^T\hat{\boldsymbol{x}} - w^T\mu_j| \quad for\ all\ i \neq j \tag{11}$$

2.3.2 Neural network based classification

Statistical models discussed so far are built on the Bayes Decision theory which considers calculating the posterior probabilities using various training strategies. Most of the statistical classification techniques have an underlying assumption of the conditional class and prior distributions. These techniques generally prevail when the model assumptions and underlying conditions are met.

Neural networks have emerged as an important alternative to traditional statistical models. Neural network unlike statistical models make no assumption regarding the

conditional class and prior distribution [29]. A neural network for classification can be defined as a mapping function $H : \boldsymbol{x} \rightarrow \boldsymbol{y}$ of a feature vector $\boldsymbol{x}$ to the output vector $\boldsymbol{y}$. This mapping function H as per the least square estimation theory is given by

$$H_j(\boldsymbol{x}) = E[y_j|\boldsymbol{x}] \tag{12}$$

where $E[y_j|\boldsymbol{x}]$ is conditional expectation of y_j given $\boldsymbol{x}$.For a binary output vector $\boldsymbol{y}$ with a j^{th} basis vector $e_j = (0, ..., 1, 0, ..., 0)$ if $\boldsymbol{x}\epsilon$ class w_j the mapping function is defined as the posterior probability as shown below

$$H_j(\boldsymbol{x}) = E[y_j|\boldsymbol{x}] = P(w_j|\boldsymbol{x}) \tag{13}$$

where $P(w_j|\boldsymbol{x})$ is the posterior probability of the class w_j given $\boldsymbol{x}$.

Neural networks are also universal approximators. As already stated neural networks provide estimates of the posterior probability. An explicit connection between the neural networks and the traditional statistical classifiers is arrived based on the posterior probabilities. Neural networks as discussed in [31] are equivalent to seven statistical classifier including the Fischer linear discriminant, the minimum empirical error classifier etc.. Also, neural network that are trained to minimize the cross-entropy cost function approximate the logistic regression model. Thus, neural network can approximate the posterior probability distribution and also bears resemblance to many statistical methods.

In the coming section we will begin with a definition of a neural network. Then proceed towards the various architectures and different training algorithms associated with each architecture. Then we will review deep neural networks and finally address the issue of learning and generalization that is normally encountered while training neural network.

Basic definition of an artificial neural network

Human beings have an inherent ability to easily recognize patterns. This ability has drawn scientist to design recognition systems that are inspired by the human cognitive system. An artificial neural network represent one such system. The basic units in an artificial neural network are the artificial neurons. Artificial neurons are highly abstract model of the natural neurons shown in Figure.4 that constitute the human cognitive system. Such an abstract model of the artificial neural network helps at discovering characteristics of the neurons that are most cognitively relevant.

The first model of the artificial neuron was presented by physiologists, McCulloch and Pitts in 1943. This neuron model considers unweighted or fixed excitatory or inhibitory connections between the binary inputs and outputs. For an input signal $x_1, x_2, ...x_n$ and $y_1, y_2, ..., y_n$ appearing at the excitatory and inhibitory input connections respectively. The neuron unit is evaluated based on the following decision rule :-

- A single inhibitory signal $y_i = 1$ deactivates the neuron and result in a 0 binary output.

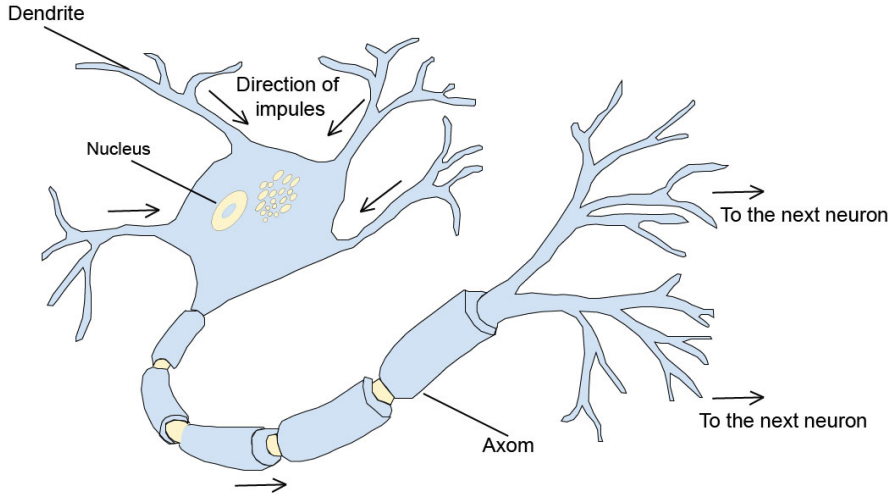


Figure 4: A neuron

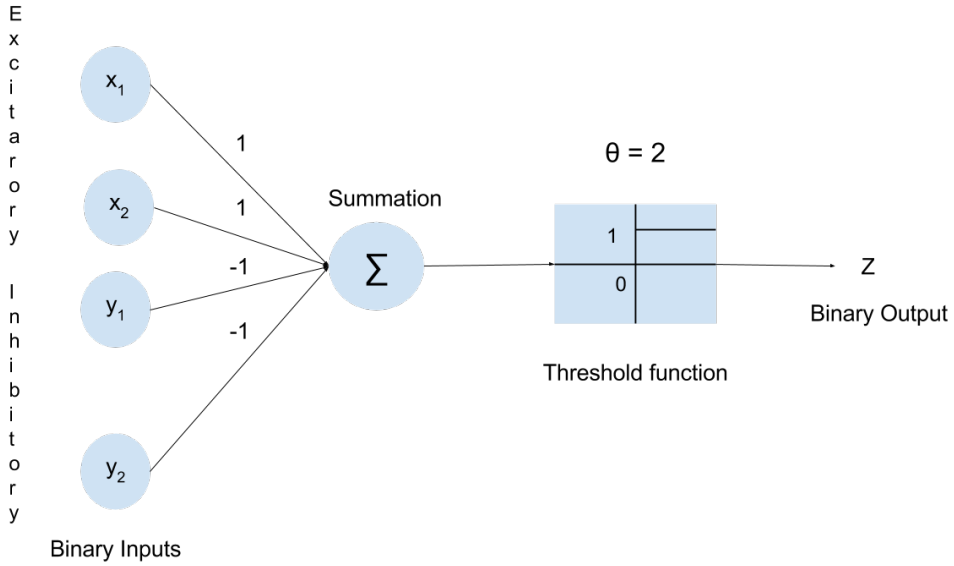


Figure 5: Mc Culloch and Pitts neuron with output $= 1$ when $\sum_{i=1} x_i \geq \theta = 2$ with $y_i = 0$ else output $= 0$ with $y_i = 1$

- If $y_i = 0$ for all $i = 1, ..., n$ then only excitation input connections is considered. The total excitation $x = x_1 + ... + x_n$ is compared with the threshold θ. The neuron is activated only with a binary output 1 if and only if $x \geq \theta$.

This neuron model is biologically relevant as it is models the electrochemical process that goes on inside a natural neuron [32]. Also, under different threshold considerations this neuron resembles most logic circuits.

The neuron presented in Figure.5 has fixed weight which are set at the very

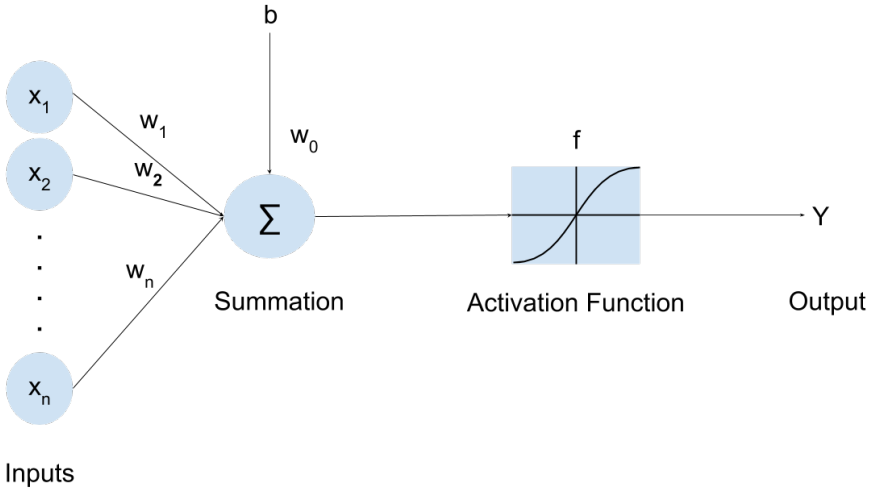


Figure 6: Single layer perceptron.

beginning. The concept of 'learning' is not available for such kind of neuron model. To improve upon this Rosenblatt in 1958 developed the perceptron neuron model. In the classic perceptron model weights are introduced which are adjusted using a trial and error method. Minsky and Papert made further improvement to the classical perceptron model. The model in Figure.6 represent the perceptron model introduced by Minsky and Papert. Mathematically this model can be represented as shown below

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{14}$$

where w_i are the adjustable weights, x_i are the inputs, f is the activation function and b is the bias

An essential element of the perceptron model seen in Figure.6 is the introduction of the activation function. The activation function allows the neuron to be able to generalizes and solve different kinds of problem. The neuron is made to resemble the McCulloch and Pitts neuron when a binary activation function is used. Activation functions such as the differentiable sigmoid activation has led to the development of various learning methods for the perceptron. Learning methods such as the least mean square and the back-propagation algorithm are possible because of the differentiable activation functions. Another key feature of the activation function is that it provides a probabilistic set-up for the neuron by projecting the output values within ranges $[0, 1]$ [33]. Lastly, the activation function also introduces non-linearity in the neuron which has helped in mapping the non-linear relationship between the inputs and the outputs.

Neural network Architecture

Neural networks have been applied to many real world classification problems. These include application in speech recognition [[34],[35]], medical diagnosis [36] etc.. The scope of neural network architecture for classification can extends to many different type. These include network architecture such the single layer perceptron, the multilayer perceptron [36], competitive models such as the self-organizing maps [37], recurrent networks [34] and energy based Hop-field networks.

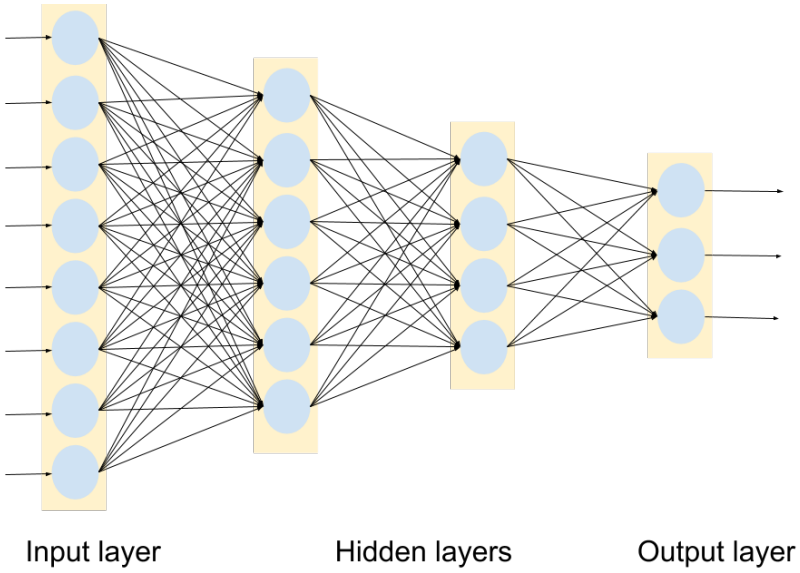


Figure 7: Multi-layer perceptron

The multi-layered perceptron with its feed-forward architecture is commonly used in classification problems. This network shown in Figure.7 consist of an input layer, one or more hidden layers and an output layer. The neuron in each layer are not connected to each other. The flow of information is unidirectional with the input undergoing a series of non-linear transformation from the input layer to the output layer.

Self organizing maps shown in Figure.8 is a fully connected single-layer network with the output layer organized in a two dimensional arrangement of nodes [37]. Using a soft competitive learning algorithm the high dimensional input is mapped to the two-dimensional array of node. Generally the self organizing map network is not intended for classification. However, in the presence of class specific data self organizing maps with the learning vector quantization can be used for classification.

Hopfield networks are symmetrically connected recurrent networks with binary units as shown in Figure.9. These binary units serves both as input as well as the output units. Hopfield nets are defined by the configuration of the binary units which is reflected in the energy function. Under stable binary configuration and minimum energy they serve as associative memories with the network able to memorize certain states and patterns [32]. An extension of the Hop-filed network is the restricted

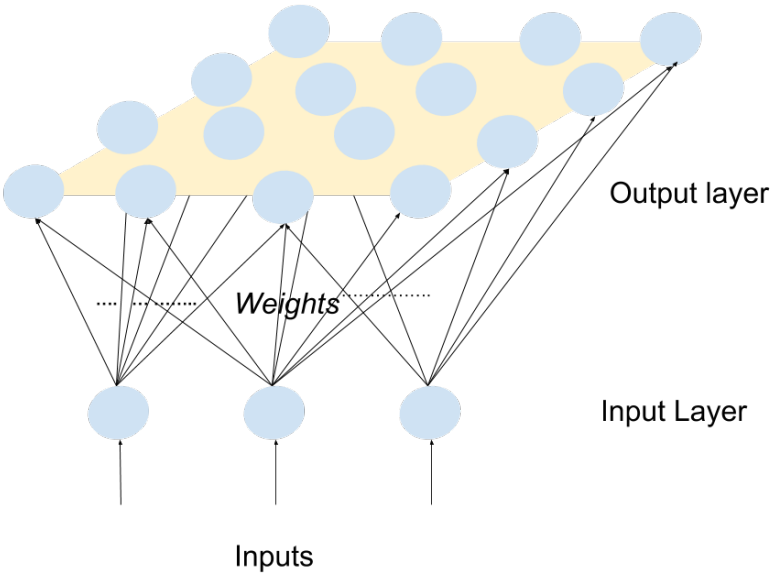


Figure 8: Self organizing maps

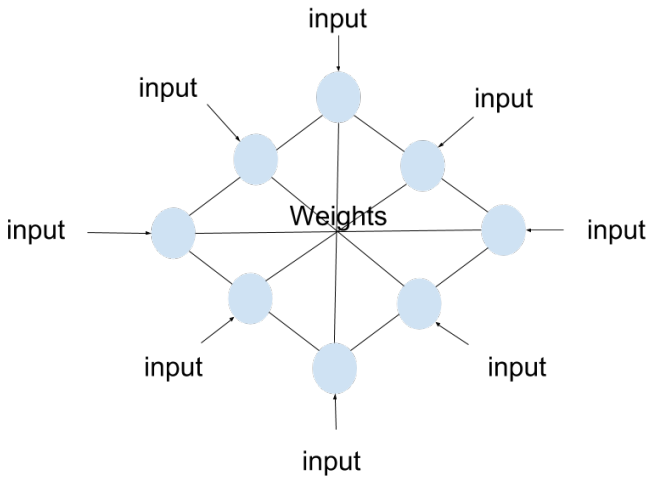


Figure 9: Hopfield networks

Boltzmann machine. The restricted Boltzmann machine considers hidden binary units while serving as associative memories. Restricted Boltzmann machines form the basis of the early deep learning research suggested by Geoffrey Hinton back in 2006. These machines when fine-trained as a multi-layered perceptron can perform classification.

Training methods in neural networks

The neural network architectures presented in the previous section were all seen to solve the classification problem. The presence of different neural network architecture has necessitate the formulation of different training methods. However, these training methods are mostly derived from the Hebbian learning rule. This Hebbian learning rule generally allows to fine-tune the variable weights between the network units in each of their respective networks.

The Hebbian learning rule is defined as follows. If two network units i and j are considered, as per the Hebbian rule the connection weight w_{ij} is strengthen when the two units i and j are simultaneously active. Thus, if the network is directed from i to j. The Hebbian learning rule updates the weight w_{ij} as shown below

$$\Delta w_{ij} = \gamma x_i x_j \tag{15}$$

where γ is the learning rate, with x_i and x_j are the network unit values of units i and j respectively.

The Hebbian learning rule presented in equation(15) is considered unstable. There is no threshold level to the increase of the weight w_{ij} between the units i and the unit j. The Oja's learning rule is an improvement of the Hebbian learning rule and is defined as shown below

$$\Delta w_{ij} = \gamma x_i [x_j - w_{ij} x_i] \tag{16}$$

Thus in equation(16) while the first update term follows the Hebbian learning, the other serves as a regulator keeping the norm of the weight vector w_{ij} close to unity.

Learning in a neural network can also be achieved by optimizing the loss function. Such learning methods are associated with multi-layer perceptron models and include the back-propagation algorithm. The back-propagation algorithm is an instantaneous stochastic gradient algorithm which tries to minimize the mean-square error between the desired output and the real output. Let us consider a multi-perceptron model with two hidden layers as shown in Figure.10. The network consist of three layers. The first layer is between the input and the first hidden layer, the second layer between the two hidden layers and the third between the hidden layer and the output.

$W^{(k)}, f^{(k)}, v^{(k)} = W^k x_{out}^{(k-1)}, x_{out}^{(k)} = f^{(k)}(v^{(k)})$ for $k = 1, 2, 3$ are the weights, activation function, linear response and output associated with each of the three layers. Also, the weight $W^{(k)}$ for each of the k layers consists of units $w_{ij}^{(k)}$ which represent connection weights between element i of layer $k-1$ and element j of layer k. For an training pair $\{\boldsymbol{x}, \boldsymbol{d}\}$, the output of the network is given as shown below

$$\boldsymbol{y} = f^{(3)}[W^{(3)} f^{(2)}[W^{(2)} f^{(1)}[W^{(1)} \boldsymbol{x}]]] \tag{17}$$

where $\boldsymbol{y}$ is the output of the network. Hence, the mean square error between the desired response $\boldsymbol{d}$ and the output $\boldsymbol{y}$ is given by

$$J_{MSE} = \frac{1}{2}\{||\boldsymbol{d} - \boldsymbol{y}||^2\} \tag{18}$$

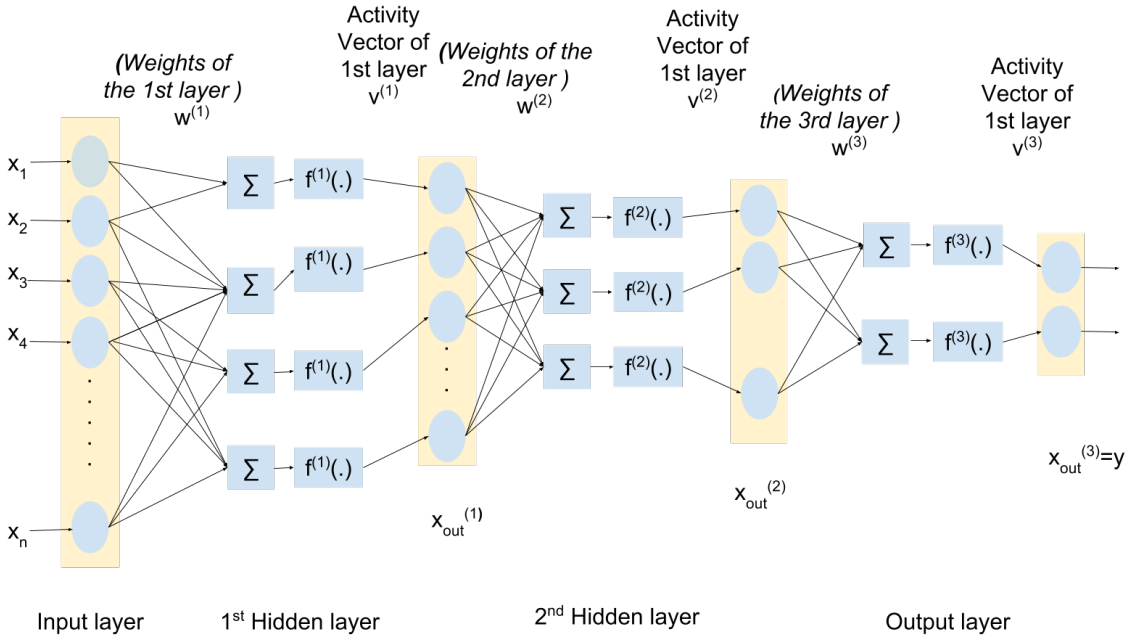


Figure 10: Multi-layer perceptron

Using the gradient descent approach for the mean square error, the update rule for a weight $w_{ji}^{(k)}$ in any of the k layers, is as shown below

$$\Delta w_{ji}^{(k)} = -\mu^{(k)} \frac{\partial J_{MSE}}{\partial w_{ij}^{(k)}} \tag{19}$$

where $\mu^{(k)}$ is the learning parameter.

The update rule can also be defined in term of the local error. To give a clear definition of this local error, assume a component of the input vector x_i is present on the i^{th} neuron of the first layer. The local error is defined as shown below

$$\delta_j^{(3)} = \frac{\partial J_{MSE}}{\partial v^{(3)}} = (d_j - y_j) g(v_j^{(3)}) \tag{20}$$

where g is the derivative of the activation function f. In general for the hidden layers the local error is given by

$$\delta_j^k = (\sum_{h=1}^{n_k+1} \delta_h^{(k+1)} w_{hj}^{(k+1)}) g(v_j^k) \tag{21}$$

Thus, the update rule for the weights in all the layer is as shown below

$$\Delta w_{ji}^{(k)} = \mu^{(k)} \delta_j^{(k)} x_{out,i}^{(k-1)} \tag{22}$$

The cost function defined by the mean square error leads to a regression model where the errors are normally distributed with the update rule being susceptible to a

learning slowdown[28]. Thus, in most cases the multi-layered perceptron is trained considering the cross-entropy cost function. This cost function uses a statistically correct binomial distributed error and is resistant to a learning slowdown [29]. The cross-entropy cost function is defined as shown below

$$J_{Cross-entropy} = \sum_{j} d_j \, log(y_j) \tag{23}$$

where $\boldsymbol{d}$ is defined as a basis vector and the output y_j for the perceptron model shown in Figure.10 is as shown

$$y_j = \frac{\exp(v_j^3)}{\sum_{j=1}^{c} \exp(v_j^3)} \tag{24}$$

where c is the number of output neurons with the output activation function $f^{(3)}$ of the perceptron model defined as a softmax function.

Neural networks such as the multi-layer perceptron discussed above are referred to as $'shallow'$ networks. The absence of a proper training algorithm limits their size to not more than two layers. If we consider a network with many hidden layers, the back-propagation algorithm would progresses extremely slow from one layer to the next. It would halt altogether without making an significant update on the weights[38].

In the coming chapter we will look at efficient ways of training neural networks with many hidden layers. These networks with their respective training algorithms are meant to overcome the under-fitting problems associated with the gradient descent algorithm [39]. We will consider using energy based stochastic Hopfield networks, auto-associators networks etc in training deep neural networks.

2.3.3 Deep neural networks based classification

Neural network models can approximate any function to any level of precision. $'Shallow'$ network models however require more hidden units and are less efficient in approximating such functions. Deep neural network are coherent with the complexity theory of circuits. A deeper neural networks architecture ensure an efficient model both in terms of the number of parameters and elements required to represent functions. Evidence also suggests that deep networks with many levels of non-linearity can handle more complex task provided, there is enough data to capture the complexity [38].

From the AI perspective, the goal is to develop systems that can mimic the human brain in its ability to learn, sense, remember and recognize. This is translated to a AI system that can learn and represent the meaningful representation of input data [40]. $'Shallow'$ networks to some extend are able to learn and solve AI problem such as classification. However, they are unable to represent features inherent to the data. Also, $'shallow'$ networks cannot work with unlabeled data.

The need for algorithm that can train deep neural networks has led researches to many different network architectures. These network architectures include convolution neural network and the restricted Boltzmann machine. These network generally follow their respective training algorithm. However, a key element observed in all these architecture is the use of an unsupervised initial training [41]. This training yields a good starting point for the parameters before they are further updated based on the task at hand.

In the coming section we will look at training deep neural networks using stacks of restricted Boltzmann machines(RBM) and using stacks of auto-encoders. Both follow a similar training schedule with the auto-encoder considered easier to train than the RBM [42]. Finally, we included in this section the training of multi-layered perceptrons with has more layers than a $'shallow'$ network. These multi-layered perceptron consider a activations function other than the sigmoid activation. With these activation function, the training algorithm is improved and hence such networks can be extended to many layers.

Training using Restricted Boltzmann Machines

The idea of training deep neural networks using stacks of restricted Boltzmann machines was proposed by Geoffrey Hinton in 2006. Hinton derived his inspiration from training deep belief networks from deep belief networks that also consists of stacks of restricted Boltzmann machines or auto-encoders. Hinton and his student Yee-Whye Teh made an observation describe in [43] that an individual layer of the deep belief network can be trained greedily one layer at a time in an unsupervised manner. Also the trained lower hidden layer served as input to the next sub-network. This algorithm was later extended as the first effective method to training deep neural networks.

Deep neural networks consisting of RBM undergo two stages of training

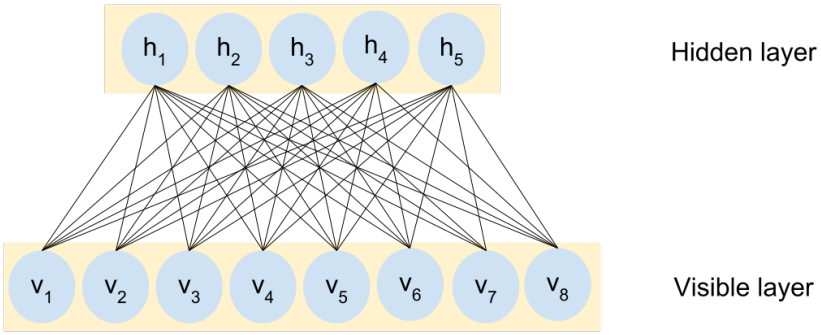


Figure 11: Restricted Boltzmann machines

- A greedy layer wise pre-training phase one layer at a time
- A fine-tuning of the network with respect to a cost function which is dependent on the criteria of interest.

In order to describe the pre-training phase it is important to define what an RBM is and then discussed how unsupervised training is performed on the RBMs.

A Restricted Boltzmann machine shown in Figure.11 is an extension of the Hopfield networks. Besides, the layer of visible units that are not connected to each other. It also consists of un-connected binary hidden units that share an undirected symmetrical connections with the visible units. Like the Hopfield networks, the configuration of the binary units in an RBM is defined in term of an energy function. The energy function $E(v, h)$ of a joint configuration of the visible v and the hidden units h is as shown below.

$$E(v,h) = -\sum_i b_i v_i - \sum_j c_j h_j - \sum_i \sum_j v_i w_{ij} h_j \tag{25}$$

which is expressed in vector form as

$$E(\boldsymbol{v},\boldsymbol{h}) = -\boldsymbol{b}^T\boldsymbol{v} - \boldsymbol{c}^T\boldsymbol{h} - \boldsymbol{h}^{\boldsymbol{T}}W\boldsymbol{v} \tag{26}$$

where W represent the weight and $\boldsymbol{b}, \boldsymbol{c}$ are the respective biases of the visible and hidden layer.

Energy based models such as the RBM also define a probabilistic distribution through this energy function

$$p(\boldsymbol{v},\boldsymbol{h}) = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z} \tag{27}$$

with the normalizing factor Z also called the partition function defined as a sum over all the configuration of the visible and hidden binary units.

$$Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{28}$$

Subsequently, the marginal distribution of the visible units over the hidden units is

$$p(\boldsymbol{v}) = \sum_{h} p(\boldsymbol{v}, \boldsymbol{h}) = \sum_{\boldsymbol{h}} \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z} \tag{29}$$

If we define a free energy function as shown below

$$F(\boldsymbol{v}) = -log \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{30}$$

then the marginal distribution reduces to :

$$p(\boldsymbol{v}) = \frac{e^{F(\boldsymbol{v})}}{Z} \quad with \quad Z = \sum_{\boldsymbol{v}} e^{F(\boldsymbol{v})} \tag{31}$$

Also, the RBM has no intra-layer connections,there is a connection only between the visible units and the hidden units. Hence, a binary hidden unit is set to 1 with the following probability for an input vector v

$$p(h_j = 1|\boldsymbol{v}) = \sigma(b_j + \sum_{i} v_i w_{ij}) \tag{32}$$

where σ is the activation function. Similarly a visible unit is set to 1 given the hidden vector h with the following probability

$$p(v_i = 1|\boldsymbol{h}) = \sigma(c_i + \sum_{j} h_j w_{ij}) \tag{33}$$

Energy based model can be learned by performing stochastic gradient descent on the likelihood of the training data

$$\frac{\partial log p(\boldsymbol{v})}{\partial \theta} = E[v_i h_j]_{data} - E[v_i h_j]_{model} \tag{34}$$

where the $E[\,]_{data}, E[\,]_{model}$ denotes the expectation over the training data and model respectively. Thus the update rule for the weight and the bias parameter is as follows

$$\Delta\theta = \epsilon(E[v_i h_j]_{data} - E[v_i h_j]_{model}) \tag{35}$$

Getting an unbiased sample of $E[v_i h_j]_{data}$ is relatively easy. In contrast as explained by Hinton in [44] it is difficult to get an unbiased sample from $E[v_i h_j]_{model}$ as that the visible units are set at a random state. Generally, it takes performing alternate Gibbs sampling for a long time to get a reliable estimate of the $E[v_i h_j]_{model}$ sample. A faster and reliable training can be achieved using the contrastive divergence method. The states of the visible units are set to that of the training vector. The hidden units are computed as per equation(34). Once the hidden units states are set, a 'reconstruction' is performed such that the hidden units are updated as per equation(35). Thus, the update rule for the parameter reduces to

$$\Delta\theta = \gamma(E[v_i h_j]_{data} - E[v_i h_j]_{reconstruction}) \tag{36}$$

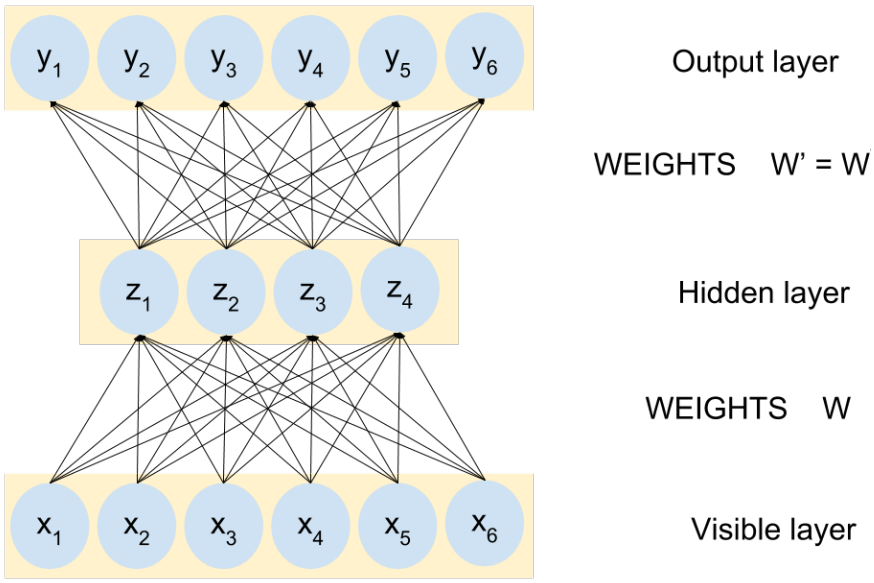


Figure 12: Auto-encoder

Contrastive divergence results in overall improvement in the training even as measured in the cross-entropy term even though it just an approximation of the log-likelihood [44]. This learning rule improves greatly if more alternating steps of Gibbs sampling are considered before $E[v_i h_j]_{reconstruction}$ is considered.

After the pre-training step, the network of stack RBMs is considered wholly. An extra layer is added on top of the stack RBM. Generally a softmax layer is consider for multi-class classification. The network is further fine-tuned using a gradient descent algorithm to minimize an error or cost function. A cross-entropy cost function is preferred.

Auto encoder

Deep neural networks are also trained using stacks of auto-encoders. Auto-encoder based deep neural networks undergo an initial greedy layer-wise pre-training and are further fine-tuned based on the task of interest.

An auto-encoder shown in Figure.12 is a three layer neural network consisting of an input, a hidden and an output layer. Auto-encoders are auto-association network with the input encoded in the hidden layer units. For an input vector $\boldsymbol{x}$, the auto-encoder maps it to a hidden representation $\boldsymbol{z}$ as shown below

$$\boldsymbol{z} = f(W\boldsymbol{x} + \boldsymbol{b}) \tag{37}$$

where W are the weights, $\boldsymbol{b}$ is the bias and f is the activation function. This latent representation $\boldsymbol{z}$ of the input $\boldsymbol{x}$ is further mapped to the output layer as shown

$$\boldsymbol{y} = f(W'\boldsymbol{z} + \boldsymbol{b'}) \tag{38}$$

where with tied weights i.e $W' = W^T$ it is meant to represent a $'reconstruction'$ of the input vector $\boldsymbol{x}$.

Training an auto-encoder involves minimizing the error between the original input $\boldsymbol{x}$ and the reconstructed input $\boldsymbol{y}$. A number of error criteria can be considered including the mean square error $E_{AE}(\boldsymbol{x}, \boldsymbol{y}) = ||\boldsymbol{x} - \boldsymbol{y}||^2$ or the cross entropy $E_{AE}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}log\boldsymbol{y} + (1 - \boldsymbol{x})log(1 - \boldsymbol{y})$. A gradient descent algorithm is considered to find parameters that minimize this error.

As already mentioned, the trained auto-encoder is meant to represent a compressed representation of the input in the hidden layers. This is true only when the number of hidden units is lesser than that of input units. Actually, for a linear activation function the auto-coder performs dimensionality reduction like the principal component analysis(PCA). An issue arises if the number of hidden units is larger or equal to the number of input units. In such circumstances the auto-encoder exactly approximates the identity function and the input is mapped directly to the hidden layer [42].

In order to ensure that the auto-encoder with large hidden units learns interesting features of the input, constraints are added to the auto-encoder network. Some of the constraints are discussed below:-

1. **De-noising auto-encoder** - This auto-encoder is trained to reconstruct a clean input from a partially corrupted input. A clean input $\boldsymbol{x}$ is partially corrupted using a corruption process $q(\boldsymbol{x}|\hat{\boldsymbol{x}})$ to get a distorted version $\hat{\boldsymbol{x}}$. For a desired distortion proportion v, distortion is achieved considering $q(\boldsymbol{x}|\hat{\boldsymbol{x}})$ as a process of randomly setting a fixed number of the input units to zero[41]. Thus, the network is trained with the distorted input $\hat{\boldsymbol{x}} = v \times \boldsymbol{x}$ while the error measure is defined between the reconstructed input at the output layer $\boldsymbol{y}$ and the clean input $\boldsymbol{x}$.

2. **Sparse auto-encoder** - For auto-encoder network with large number of hidden units; the introduction of a sparsity constraint allows them to learn interesting features of the input. Such sparsity constraint include the L1 regularization of the weights units. This regularization ensures only a few weight units are active for a given input.
   Another sparsity constraint as discussed in [45]uses the Kullback-Leibler divergence and measures the difference between set number of active hidden units and the actual number of active hidden units. If we consider the set number of active hidden units or the sparsity parameter and the actual number of active hidden units as random variables. The sparsity parameter can be defined as a Bernoulli distribution with mean ρ. Similarly, the actual number of active hidden units can be defined as a Bernoulli distribution with mean $\hat{\rho}$. Thus, the KL-divergence between the random variable with mean ρ and $\hat{\rho}$ is as shown below
   $$KL(\rho||\hat{\rho}) = \rho log\frac{\rho}{\hat{\rho}} + (1 - \rho)log\frac{1 - \rho}{1 - \hat{\rho}} \tag{39}$$
   For a auto-encoder network defined with x_i as the input units, the activation of the hidden unit z_j for each of the k training sample with $k = 1, ..., m$ is as

shown below

$$z_j^{(k)} = f(w_{ij}x_i^{(k)} + b_j) \tag{40}$$

where f is the activation function,w_{ij} is the weight connecting the i^{th} input unit to the j^{th} hidden unit.
Hence, the mean activation $\hat{\rho}_j$ for each of the hidden units is

$$\hat{\rho_j} = \frac{1}{m}\sum_{k=1}^{m} z_j^{(k)} \tag{41}$$

Usually, when training the auto-encoder using a stochastic gradient descent the value of m is limited to the batch size. The mean of the sparsity parameter is set to be very small; say $\rho = 0.05$. When the sparsity constraint is enforce on the network hidden unit the loss function is given by

$$E_{sparse}(\boldsymbol{x}, \boldsymbol{y}) = E_{AE}(\boldsymbol{x}, \boldsymbol{y}) + \sum_{j}^{c} KL(\rho||\hat{\rho}_j) \tag{42}$$

where c is the number of hidden units. The above loss function is optimized such that the over all error decreases and at the same time limiting the number of active hidden units; with $\hat{\rho}$ close to ρ.

3. **Contractive auto-encoder** - The Contractive auto-encoder is defined by the addition of a penalty term to the cost function $E_{AE}(\boldsymbol{x}, \boldsymbol{y})$ of the original encoder. This penalty term is the Frobenius norm of the Jacobian of the hidden features with respect to the original input.
Again, for a auto-encoder network defined with the input units x_i and hidden units z_j, for each of the k training sample with $k = 1, ..., m$ the penalty term is as shown below

$$||J_h(\boldsymbol{x^{(k)}})||_F^2 = \sum_{i,j}(\frac{\partial z_j(x_i^{(k)})}{\partial x_i})^2 \tag{43}$$

Hence, the overall cost function of the contractive auto-encoder is defined as

$$E_{CAE}(\boldsymbol{x}, \boldsymbol{y}) = E_{AE}(\boldsymbol{x}, \boldsymbol{y}) + \sum_{k=1}^{m} \lambda||J_h(x^{(k)})||_F^2 \tag{44}$$

where λ is a parameter that takes values between 0 and 1. Generally the value of m is limited to the batch size when batch stochastic gradient descent is used to optimized the cost function.
The penalty term introduce for the contractive auto-encoder penalizes the $'sensitivity'$ of the hidden layer representation with respect to the input. With a sigmoid activation function, this penalty term encourages the hidden layer values towards the left asymptote of the activation function. This regions result in near zeros hidden values with a minimum rate of change with respect to the input. Thus, the penalty term can be considered as one that result in more sparse and robust representation of the input in hidden layer [46].

Auto-encoder with different constraints can also be combined. As shown in [47] a combination of a de-noising auto-encoder with a contractive auto-encoder can achieve better results than when a constraint is used singly.

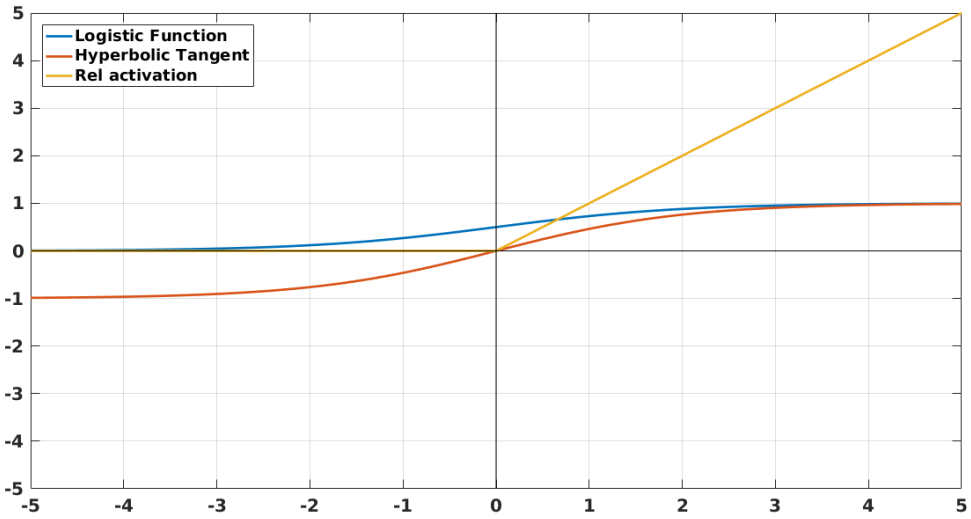


Figure 13: Rectified linear activation

Multi-layer Perceptron

The previous section has seen how deep neural networks can be trained using stacks of restricted Boltzmann machines and auto-encoder. These neural networks undergo an initial greedy unsupervised training. This is followed by further fine-tuning of the network parameters to optimize a cost function based on the task. In our case the task involves classification and the parameters initialized by the greedy unsupervised training are fine-tuned to optimize a mean square or cross entropy cost function between the real and predicted values.

Training deep neural using RBMs and auto-encoder is time-consuming even with specialized hardware like GPUs [48]. Also, if we consider the case of training of deep network using RBM. The log-likelihood which is optimize in the pre-training phase is intractable and can only be approximated. Thus, there is no certain metric to measure the progress of the training [49]. Deep neural networks can also trained without considering an initial unsupervised pre-training. This is achieved by considering an activation function such as the Rectifier Linear(ReL) activation shown in Figure.13. Mathematical this activation function at a hidden unit $h^{(i)}$ is given by

$$h^{(i)} = max(w^{(i)T}x, 0) \tag{45}$$

or

$$h^{(i)} = \begin{cases} w^{(i)T}x, & \text{if} \quad w^{(i)T}x \geq 0 \\ 0, & \text{if} \quad else \end{cases}$$

where x is the input and $w^{(i)}$ is the weight vector of the i^{th} hidden unit.

For ReL non-linearity hidden units that are activated(above zero values), the partial derivative with respect to the network parameters is always linear [48]. Unlike sigmoid activation unit which involve exponential or division; vanishing gradients

does not exist along paths of a network with activated with ReL non-linearity hidden units and are faster to train [50]. Also, the neural network with many active hidden units reduces to a linear convex system. Hence the optimization is straightforward even with first order optimizers.

2.3.4 Issues related to learning and generalization

So far we have discussed neural networks, including the deep neural networks and their various training regime. In this section we define the performance metric that are monitored while training these neural networks. These metrics include; the ability of the network to learn from the training data. It also include its ability to generalize to unseen data. Also discussed in this section; the various regularization methods that results in overall improvement of these network metrics.

Learning and generalization are important metrics that need to be considered while training neural network models;especially deep neural networks. During training, these neural network model should learn to approximate only the underlying behavior of the training data and remain insusceptible to minor fluctuations in the training data. Network models especially those trained as classifiers should also have an inherent ability to generalize well to data that is beyond the training data.

Learning and generalization are correlated albeit negatively. This is well analyzed using the bias-variance decomposition of the error function of the network model [28]. Neural networks models that tend to over-fit the training data are describe as models with a low bias and high variance. According to the bias-variance decomposition a good network model should balance between these two metrics and also achieve minimum prediction error.

The network ability to learn and generalize is data dependent. Generally, the input data is encourage to be of low dimensionality(curse of dimensionality) [51]. Also, a network model with regularization parameters such as the L1 and L2 norm, the right number of hidden unit is intended to prevent the network from over-fitting.

Deep neural networks that undergo a initial unsupervised pre-training results in better generalization. Pre-training results in an increases the magnitude of the weights which in turn result in a complicated cost function [52]. The gradient descent algorithm such as the stochastic gradient descent cannot easily transverse this complicated cost function. Hence the network parameters are regularized and undergo only small changes in the fine-tuning phase. Various regulation methods are also included in deep neural network to help prevent the network from over-fitting. These include the method of early stopping and dropout which are discussed in more details below.

1. **Dropout**

   Dropout presented in [53]is motivated by model combination which is a well known theory in machine learning. Model combination as the name suggest is a combination of models with different architecture or with different parameter setting. These models are also trained considering different training data. Model combination always results in improved performance compared to when

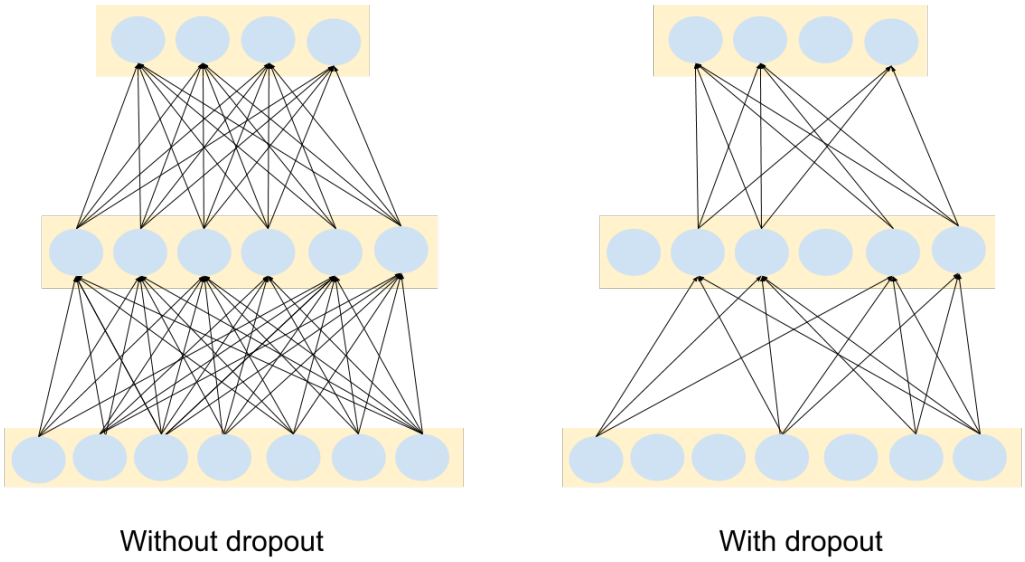


Figure 14: Dropout

individual method are used separately.
Model combination is in-efficient for deep neural networks with many hidden layers. Training each individual network is time consuming and computational expensive. Also, each network require a lot of training data which is very difficult to source.
In order to have an efficient model combination in deep neural network we use dropout. In dropout a hidden unit along with its incoming and outgoing connections are temporary removed as shown in Figure.14. This is done on random hidden units. Usually, dropout rates are fixed at 0.5 which means each hidden units has a 50 percent chance of being dropped at any given time. The dropout rates for the input units are usually lesser say at around 0.2.
Dropout amounts to sampling "*thinned*" networks from a large network with many hidden units. For a large network with n hidden units, there can be as many as 2^n thin network. Each $'thinned'$ network consist of units that have survived the "*dropping*" out. Also, as the input units are dropped so we can say that the input is somewhat different for each "*thinned*" network.
Model combination or averaging over an exponential number of $'thinned'$ networks can be difficult. Thus, in the testing phase all the $'thinned'$ networks are treated as one large network without dropout. The weights of this large network is scaled down depending on the dropout parameter associated with their respective training.
Dropout can also be extended to pre-training of RBMs. It is also related to the de-noising effect introduce in section on auto-encoders.

2. **Early-stopping**

   Early stopping is another method that serves as a regularizer while training deep neural networks. As already stated above the ability of the network model to generalize to unseen data is an important metric that is monitored during the training process. With early stopping we consider a training set as well as a validation set. This validation data is sampled from the training data. While it is expected that the error on the training data will always decrease; as the training progress there is a high chance that over-fitting might set in and error on the validation set may stop decreasing and start increasing. Hence the training is stopped as soon as there is sign of the validation error no longer decreasing.

3 Review of front end noise robust techniques

In this chapter, we begin with a mathematical model of the impact the additive and convolutive distortions has on the clean speech. We will then describe the compensation methods from the feature perspective. Due to the vast nature of the the robust speech recognition history it impossible to describe all the compensation methods. Thus, the scope of this review is limited to single-channel inputs. Also the additive and convolutive distortions are considered more stationary that the original speech signal [54].

3.1 Mathematical model of speech in noise environment

Consider the case where $x(n)$, $n(n)$ and $y(n)$ represent in the time domain the clean speech, the noise and the corrupted speech, respectively. Here n is the time domain index. Let $h(n)$ denote the impulse response of the channel. The mathematical model of speech affected by the channel distortion and the additive noise as shown in Figure.15 is given by :-

$$y(n) = x(n) * h(n) + n(n) \tag{46}$$

with $*$ representing convolution between the clean speech signal and the channel impulse response.

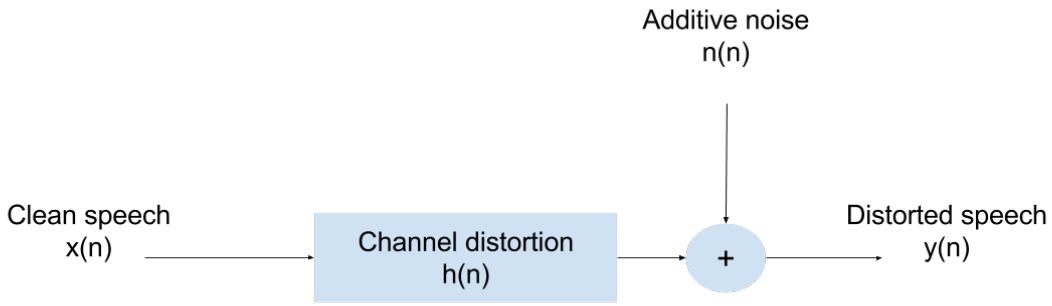


Figure 15: Model of the clean speech affected by channel distortion and additive noise

If we apply the short-time Fourier transform(STFT) we get the spectral or frequency domain representation of equation(46) as shown below :-

$$Y(k) = X(k)H(k) + N(k) \tag{47}$$

where $k = 1, ..., K$ is the Fourier coefficient index. Also $Y(k), H(k), X(k)$ and $N(k)$ are the frequency domain representation of the corrupted signal, channel

distortion, clean signal and additive noise respectively. Now the power spectrum of the corrupted speech is obtained as shown below

$$\begin{aligned} |Y(k)|^2 &= Y(k)Y^*(k) \\ &= |X(k)|^2|H(k)|^2 + |N(k)|^2 + X(k)H(k)N^*(k) + (X(k)H(k)N^*(k))^* \end{aligned} \tag{48}$$

The last two terms of equation(48) are a product of two complex numbers. This product can be represented as the product of their magnitude times the cosine of the angle between them. Thus equation(48) reduces to

$$|Y(k)|^2 = |X(k)|^2|H(k)|^2 + |N(k)|^2 + 2|X(k)||H(k)||N(k)|cos\theta_k \tag{49}$$

$$cos\theta_k = \frac{X(k)H(k)N^*(k)}{|X(k)||H(k)||N(k)|} = \frac{X(k)^*H(k)^*N(k)}{|X(k)||H(k)||N(k)|} \tag{50}$$

where θ_k denotes the angle between the two complex variables $|N^*(k)|$ and $H(k)X(k)$. Assume the case that we are extracting MFCC features. It a common practice to not consider the phase component and thus equation(49) is reduced to

$$|Y(k)|^2 = |X(k)|^2|H(k)|^2 + |N(k)|^2 \tag{51}$$

The extraction of MFCC features involves applying a set of Mel-filter banks to the power spectrum and calculating the power spectrum encompassed by each of the filter bank. Thus the energy for the corrupted speech, clean speech, noise and channel distortion in each of the filter bank is given by

$$\begin{aligned} |Y(l)|^2 &= \sum_k W_k(l)|Y(k)|^2 \\ |X(l)|^2 &= \sum_k W_k(l)|X(k)|^2 \\ |N(l)|^2 &= \sum_k W_k(l)|N(k)|^2 \\ |H(l)|^2 &= \frac{\sum_k W_k(l)|X(k)|^2|H(k)|^2}{|X(l)|^2} \end{aligned} \tag{52}$$

where the l^{th} filter is characterized by the weights $W_k(l)$ $k = 1, ..., K$ and $\sum_k W_k(l) = 1$ Thus in the Mel filter-bank domain equation(51) is given by

$$|Y(l)|^2 = |X(l)|^2|H(l)|^2 + |N(l)|^2 \tag{53}$$

Taking the natural logarithm and DCT on the filter bank coefficients

$$DCT(ln|Y(l)|^2) = DCT(ln(|X(l)|^2|H(l)|^2 + |N(l)|^2)) \tag{54}$$

where DCT represent the discrete cosine transform. To represent equation(54) in a simple way we defined the following vector notations

$$\begin{aligned} \boldsymbol{x} &= DCT(ln|X(l)|^2) \\ \boldsymbol{y} &= DCT(ln|Y(l)|^2) \\ \boldsymbol{n} &= DCT(ln|N(l)|^2) \end{aligned} \tag{55}$$

Hence equation(54) can be written as follows

$$\begin{aligned}\boldsymbol{y} &= DCT\left\{ln\left[|X(l)|^2|H(l)|^2\left(1+\frac{|N(l)|^2}{|X(l)|^2|H(l)|^2}\right)\right]\right\}\\ &= DCT\left\{ln|X(l)|^2\right\}+DCT\left\{ln|H(l)|^2\right\}+DCT\left\{ln\left(1+\frac{|N(l)|^2}{|X(l)|^2|H(l)|^2}\right)\right\}\\ &= \boldsymbol{x}+\boldsymbol{h}+DCT\left\{ln(1+\exp(IDCT[\boldsymbol{n}-\boldsymbol{h}-\boldsymbol{x}]))\right\}\end{aligned} \tag{56}$$

where $IDCT$ is the inverse of the discrete cosine transform. Equation(56) represent the most widely used cepstral domain representation of the corrupted speech signal.

3.2 Feature space approaches for robust speech recognition system

Feature space approaches to robust speech recognition system are usually divided into three sub-categories. These include using signal processing algorithms to provide a refine model of the auditory system [55]. Feature space approaches also include methods that normalize the moment associated with the speech features. Also, feature space approaches include compensation methods where a critical feature of the noise that differentiates it from the speech signal is evaluated and serve as the resource for finally removing the noise.

3.2.1 Noise resistant features

Noise resistant features are derived with a minimal assumption of the underlying noise. These feature extraction methods are more focused on the effects the noise has on the signal [54]. Noise resistant features include features derived from the auditory processing as well as neural network based methods. Features extracted using such methods have shown to be more robust even under severe noise conditions (Kim and Stern, 2012).

a. Auditory features

Auditory features are derived from the auditory processing of speech signal. An auditory model is a complex subsystem of the peripheral hearing system. The physiological functions of hearing system from the basilar membrane up to the cochlear process and the neuron are simulated in the model [56].

Some of the commonly used auditory features include the PLP and its variants. PLP features are based on the psycho-analysis finding. For computational purpose it is interpreted in signal processing terms [57]. A major part of the PLP feature extraction is similar to the MFCC. The power spectrum of the short time Fourier transform is computed. This power spectrum is multiplied with Bark filter-bank instead of the Mel-scale filter-banks. The PLP feature extraction also takes into

account with the equal-loudness pre-emphasis; the frequency sensitivity of the human hearing. This equal-loudness pre-emphasis is achieved on a signal processing level by multiplying the power spectrum coefficient $P(\omega)$ with a weight that depends on the frequency. This weight is given by

$$E_1(f) = \frac{(f^2 + 1.44 * 10^6)f^4}{(f^2 + 1.6 * 10^5)^2(f^2 + 9.61 * 10^6)} \tag{57}$$

where $f = \frac{\omega}{2\pi} f_{sample}$. For signal with frequency greater than 5 kHz, an alternative weighing function is defined by

$$E_2(f) = \frac{(f^2 + 56.8 * 10^6)f^4}{f^2 + 6.3 * 10^6)^2(f^2 + 0.38 * 10^9)} \tag{58}$$

It can be seen that the value of the $E_1(f)$ is similar to the pre-emphasis done in the MFCC feature extraction.

What then distinguishes the PLP from the MFCC and make them more robust. In the feature extraction using MFCC the logarithmic non-linearity is used. This logarithmic non-linearity provides no threshold and small variations in the input can result in large output changes [58]. This characteristic leads to de-gradation in speech recognition accuracy especially in cases when input approaches zero. Small differences in additive noise can produce large differences in the output of the logarithmic non-linearity. Unlike the MFCC, the power spectrum $P(\omega)$ of the PLP are raised to the power of 0.33. With a power-function non-linearity, the output is close to zero if the input is very small. This is also observed in human auditory processing [59] and leads to more robust features.

There are many other feature extraction methods that can be defined as auditory based. These include the power-normalized cepstral coefficient, Gammatone frequency cepstral coefficient, Gabor based filter bank, Amplitude modulation spectrogram and many more. Auditory features however have their drawbacks. Firstly, different feature extraction method consider different set of auditory information. Essentially, there is no universally as to which auditory information that is useful to incorporate in a speech recognition system. Secondly, auditory features are based on the physiological and psychological nature of the auditory system and are complex models. Thus, they are not widely used in noise-robust systems [54]. Also, with reference to a practical transcription process like news broadcasting which include noisy and clean conditions at regular interval. The auditory model is further tweaked to accommodate both conditions. This is essential as such auditory features might perform well under noisy condition; they certainly do not perform better than the likes of the MFCC features under clean conditions.

b. Neural network based features

Unlike the Hybrid connectionist-HMM model where the neural network replaces the GMM acoustic model in estimating the posterior probability given the data [60]. Neural networks based methods provide $features$ that are used with Gaussian mixture modeling. These neural network based methods include the TANDEM

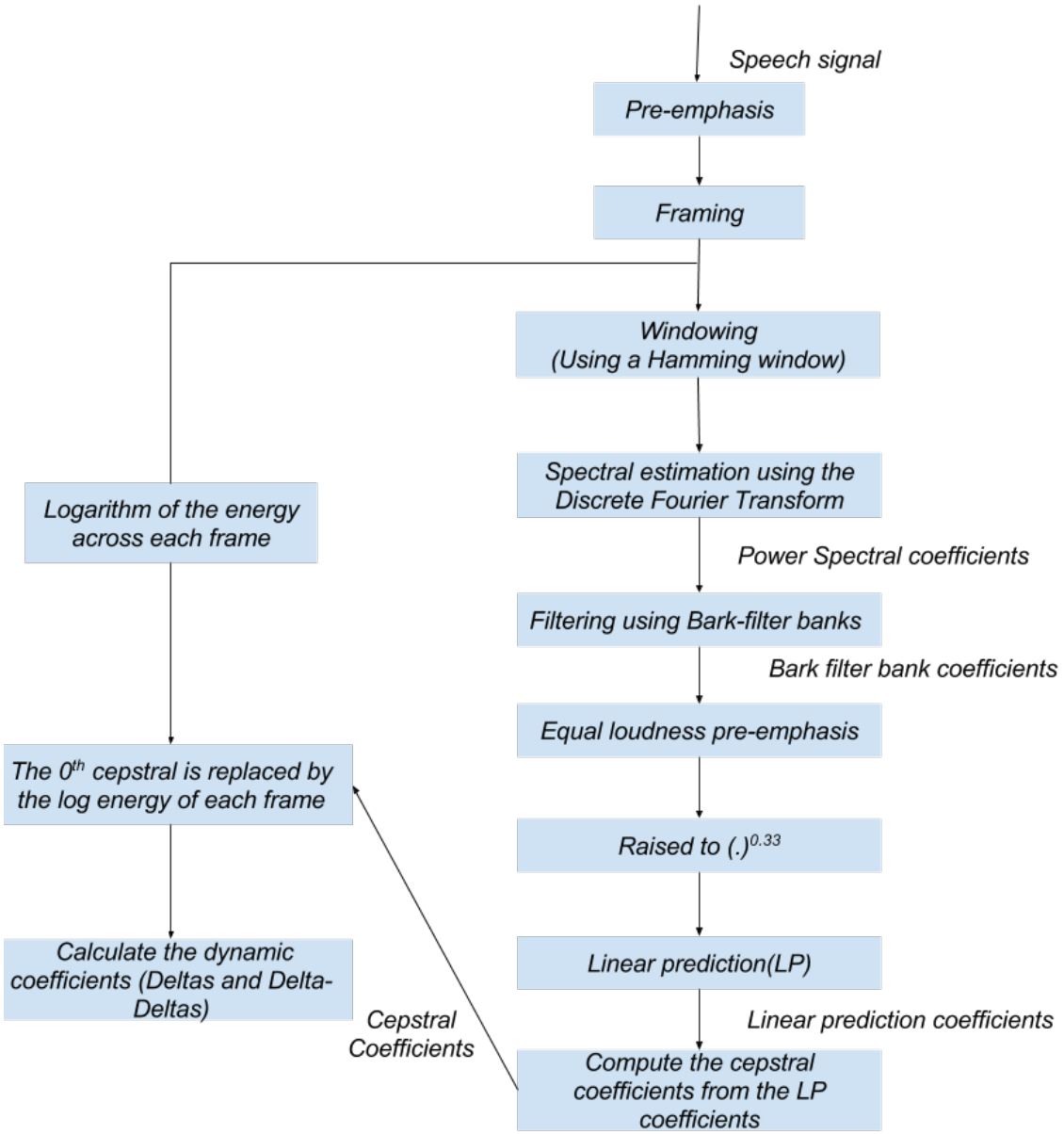


Figure 16: Flowchart showing the step by step process in the extraction of PLP features

connectionist feature extraction approach, the TempoRAL Patterns(TRAPS) that that takes energy pattern over long periods, the bottle-neck approach and the DNN based approach.

In the TANDEM approach the training is done using the back-propagation algorithm. The neural networks however are devoid of a non-linearity at the output layer. This is done to ensure that the emitted probability are not skewed and constitute the log-posterior probability. This log-probability is further PCA de-correlated to finally serve as inputs to a GMM-HMM system. The TANDEM system

with its non-linear hidden units makes it a highly robust system. Its non-linear hidden units is able to normalize data from different sources and also model with good accuracy small regions of feature space that lie on phone boundaries [60].

TempoRAL pattern (TRAP) processing represent another method that can be used to generate features. The two level TRAP system takes in a temporal pattern input of long critical energy from a single frequency band. The temporal pattern is classified by a band conditioned non-linear classifier. The output from all the band conditioned classifier are then merged for the final classification. The TRAP based feature is also robust to noise conditions. The final classification is dependent on the cumulation of band classifiers and does not depend on a single classifier [61].

Bottle neck features of an ANN also serve as features that can be used in conjunction with Gaussian mixture modeling. These features are outputs of the bottle neck layer in the MLP and are different from the probabilistic features derived from the TANDEM approach. The extraction of such features is similar to the TRAP based system. The log energy of each temporal pattern is transformed using a DCT before feeding it to the neural net classifier. Also, Heteroscedastic Linear Discriminant Analysis(HLDA) technique is used instead of the PCA to decorrelate the features before feeding them to a GMM/HMM system [62].

c. Deep neural network based features

Deep neural networks based features are similar to the probabilistic features derived using the TANDEM or the TRAP approach. However unlike the TANDEM or TRAP based approach the network is trained to predict tied context-dependent acoustic states called senones. Unlike the earlier TANDEM or TRAP approach deep neural networks also have many hidden layers [63]. Thus, the extracted features after decorrelation serve as input to a GMM/HMM system. There is an added advantage to using deep neural networks(DNN) to achieve robustness of the speech recognition system. Under multi-training conditions the DNN is seen to achieve comparable results compared to model adapted GMM/HMM models using the Aurora dataset. Also, robustness is further improved considering a model adaptation to the DNN such as the drop-out which was initially design to over-come the over-fitting problem.

3.2.2 Feature moment normalization

Under clean conditions, cepstral features approximates a normal distribution. However, under noisy conditions they take a different profile. Noise result in mean shift and a reduction in variance of the clean normal distribution [64]. Further, the noisy cepstral distribution edges also becomes steeper with a tendency towards a bi-modal distribution and becoming non-Gaussian. It is on this basis that the simple feature moment normalization techniques are proposed. These normalization techniques include the cepstral mean normalization, the cepstral mean and variance normalization and the histogram normalization.

a. Cepstral mean normalization

Cepstral mean normalization(CMN) compensate the channel effects which manifest as the convolution noise. For feature extraction methods such as the MFCC which considers the log-filter domain; the channel effects or convolution noise is additive in the cepstral domain. CMN compensates this channel effect by matching the first order moment of the training and test data and transforming the data to have zero mean [65].

Let $\boldsymbol{x}_t$ denote the cepstral vector at time t. $x_t(i)$ represent the i^{th} component of $\boldsymbol{x}_t$. For an utterance of length T the cepstral data is given by $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t, ..., \boldsymbol{x}_T]$. CMN is performed by computing the maximum-likelihood estimate of the mean(μ), as shown below

$$\mu(i) = \frac{1}{T}\sum_{t=1}^{T} x_t(i) \tag{59}$$

Thus, the estimated cepstral data is now transformed as shown below

$$\hat{x}_t(i) = x_t(i) - \mu(i) \tag{60}$$

CMN considers a whole utterance for the calculation of its mean value. Such is not acceptable for real-time applications. Thus, CMN is designed as a high-pass filter approximation. The cepstral mean is a function of time and the normalized cepstral values are as shown below

$$\begin{aligned} \mu_{x_t} &= \alpha \boldsymbol{x}_t + (1-\alpha)\boldsymbol{\mu}_{x_{t-1}} \\ \hat{\boldsymbol{x}}_t &= \boldsymbol{x}_t - \boldsymbol{\mu}_{x_t} \end{aligned} \tag{61}$$

where μ is a constant and $\mu_{\boldsymbol{x}_{t-1}}$ is the estimate of the mean(μ) upto $t-1$ of the cepstral data. The CMN has found to improve performance under noisy conditions even if there is an absence of channel noise.

As it is suggestive that the noise influence both the mean and variance of the cepstral distribution. We begin looking at another normalization scheme that considers the first and the second order moment namely the cepstral mean variance normalization and finally the histogram normalization.

b. Cepstral mean variance normalization

Cepstral mean variance normalization(CMVN) matches the first and the second order moment of the train and the test data. Thus, the data is transform to have zero mean and unit co-variance. Unlike the mean normalization that removes the channel distortion, CMVN cannot explicitly remove any form of distortion [54]. The main intent is to reduce the first and second order moment mismatch between the train and the test data; thus ensuring distortion brought about by additive noise and the convolutive channel is reduced.

Again, if $\boldsymbol{x}_t$ denote the cepstral vector at time t. $x_t(i)$ represent the i^{th} component of $\boldsymbol{x}_t$. And for an utterance of length T the cepstral data is given by

$X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t, ..., \boldsymbol{x}_T]$. CMVN is performed by computing the mean and the variance in the maximum-likelihood framework as shown below

$$\begin{aligned} \mu(i) &= \frac{1}{T}\sum_{t=1}^{T} x_t(i) \\ \sigma^2(i) &= \frac{1}{T-1}\sum_{t=1}^{T}(x_t(i) - \mu(i))^2 \end{aligned} \tag{62}$$

The normalized cepstral vectors is given as shown below

$$\hat{x}_t(i) = \frac{x_t(i) - \mu(i)}{\sigma(i)} \tag{63}$$

One of the main drawbacks of the CMVN is that it cannot be extended to real-world applications. Like the CMN it considers the whole utterance for the calculation of the mean and the variance. However this can be solved by considering shorter segments of the utterance. Another drawback of the CMVN is that the resultant cepstral vectors after normalization are found to be less discriminant [65]. This in turn results in decrease in performance when such vectors are used in a speech recognition system. To counter this decrease in discriminativeness of the cepstral features, the CMVN is performed using the Bayesian framework for parameter estimation. These Bayesian-CMNV features have non-zero mean and non-unit variance. Also, the mean and variance of the i^{th} and j^{th} dimensions of the cepstral vector $\boldsymbol{x}$ are no longer equal. Such characteristics show that they are greatly more discriminative than the CMVN features.

c. Histogram normalization

Histogram normalization is meant to approximate all the moments associated with the training and test data. It assumes the transformation brought about by the noise on the clean signal is invertible. Noise is not an invertible process but its average effect can be consider invertible. Thus, histogram equalization aims to eliminate this average effect of the noise [66].

If we assume a feature $\boldsymbol{x}$ with a distribution $p_{Test}(\boldsymbol{x})$ derived from the test set. The noise transformation $F(\boldsymbol{x})$ is assume to be a non-decreasing monotonic non-linear function. The non-linear function converts the initial clean test data $\boldsymbol{x}$ to noise data $\boldsymbol{y}$. If the definition of the noise transformation F with respect to the cumulative distribution of the train data and test data C_{Train} and C_{Test} respectively as :

$$\begin{aligned} C_{Train}(F(\boldsymbol{x})) &= C_{Test}(\boldsymbol{x}) \\ F(\boldsymbol{x}) &= C^{-1}_{Train}(C_{Test}(\boldsymbol{x})) \end{aligned} \tag{64}$$

Then, the probability distribution of the noised transformed test data will be equal to that of the train data.

3.2.3 Feature Compensation

Feature compensation methods aims to compensate environmental effects that are both convolutive and additive to the speech signal. These methods include the spectral subtraction and RASTA filtering. For an additive environmental effect, spectral subtraction is preferred. In spectral subtraction the power spectrum of the signal is subtracted from an estimate of the noise power spectrum. Generally, environmental effects that are convolutive to the speech signal also have different temporal properties compared to the speech signal [7]. RASTA filtering is designed to remove this convolutive environmental effect taking advantage of this temporal difference.

a. Spectral subtraction

The spectral subtraction method assumes an additive noise and clean speech that are uncorrelated in the time domain [67]. This method estimates the noise average power spectrum during the non-speech period. If the power spectrum of the noise and the clean speech spectrum is as shown below

$$|Y(k)|^2 = |X(k)|^2 + |N(k)|^2 \tag{65}$$

where $|Y(k)|^2, |X(k)|^2$ and $|N(k)|^2$ are the power spectrum of the corrupted signal, clean signal and noise respectively. Thus the noise power spectrum in N non speech frames is $|\hat{N}(k)|^2 = \frac{1}{N}\sum_{i=0}^{N-1}|Y_i(k)|^2$. The clean speech power spectrum is estimated by subtracting $|N(k)|^2$ from the noisy speech power spectrum as shown below

$$\begin{aligned} |X[k]|^2 &= |Y(k)|^2 - |N(k)|^2 \\ &= |Y(k)|^2 G^2[k] \end{aligned} \tag{66}$$

where

$$G[k] = \sqrt{\frac{SNR(k)}{1 + SNR(k)}} \tag{67}$$

is real valued gain function and

$$SNR(k) = \frac{|y[k]|^2 - |n[k]|^2}{|n[k]|^2} \tag{68}$$

is the frequency dependent signal to noise ratio. Spectral subtraction requires a speech detector from which a reliable noise estimate can be obtained. Also, the subtraction process can result in negative spectral values. This is typically handled by setting the negative spectral values to zero [7].

b. RASTA filtering

Relative spectral processing (RASTA) consists of suppressing environmental effects or noise that are additive in the log-spectral domain. For a short-term auditory spectrum like PLP the process of RASTA filtering is as shown in Figure.17.

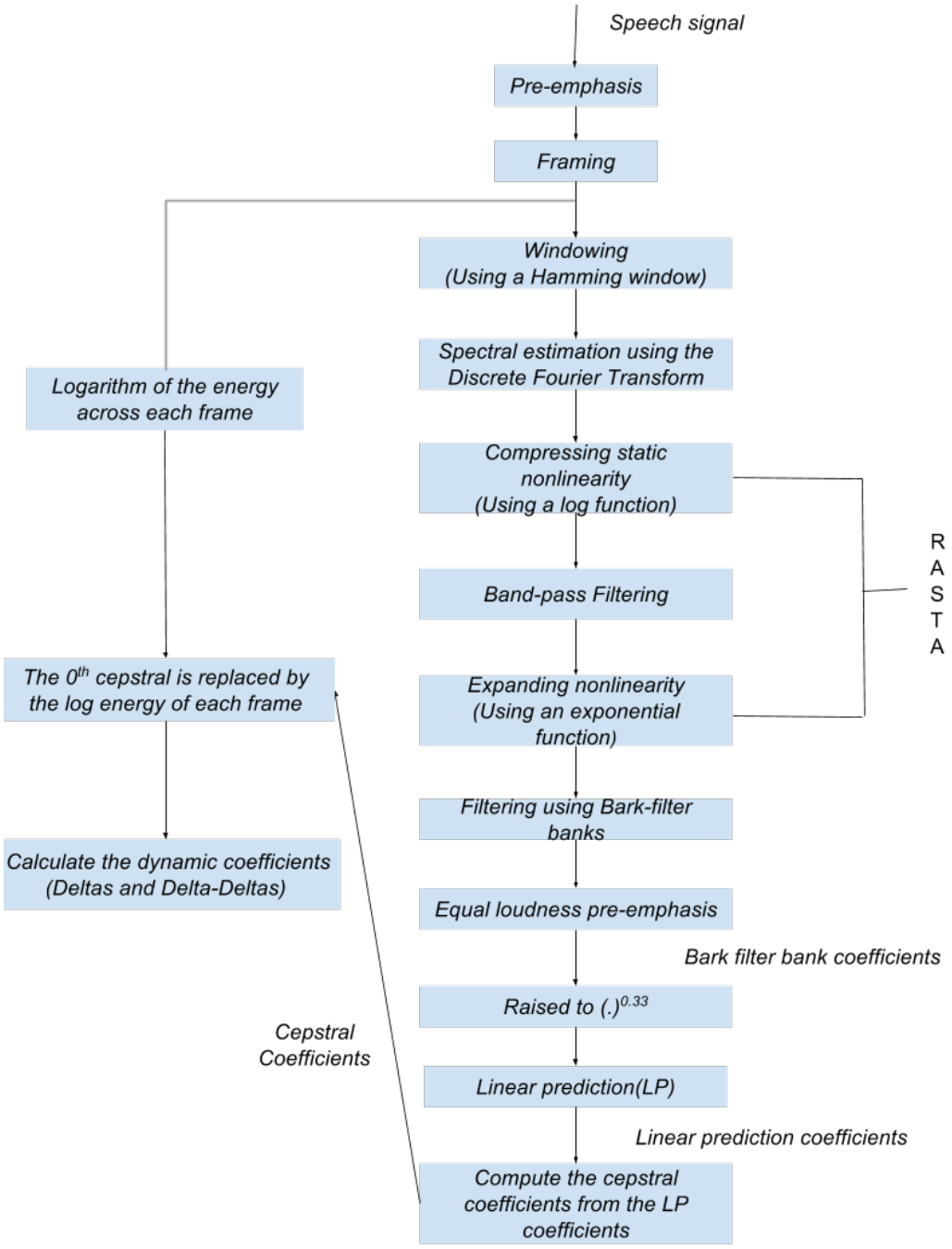


Figure 17: Flowchart showing RASTA filtering integrated in the extraction of PLP features

The critical power spectrum is transformed through the compressing non-linear transformation that is dependent on the SNR . The time trajectories of the spectral component are filtered by a non-causal infinite impulse response (IIR) filter. After the filtering process the power spectrum is transformed back using an exponential

transformation. The filter transfer function is as shown below

$$H_{RASTA}(z) = 0.1z^4 \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.98z^{-1}} \tag{69}$$

This filter has a low cut-off frequency of 0.26 Hz and the filter slope declines at $6dB/oct$ from 12.8 Hz with sharp zeros at 28.9 and 50 Hz. It also considers a long time constant of 500 ms. A pole order of $(z = 0.94)$ which result in a time constant of $160ms$ is also considered optimum [7]. This band-pass filter alleviates the effect of convolution noise and helps to smooth out the fast frame-to-frame spectral changes present in the short term spectrum.

4 Features derived from the AM-FM analysis of speech

In the previous section we have discuss the extraction of a number of robust features. These include auditory based features such as the PLP and neural network based features such as the TRAP and the TANDEM approach.

Conventional speech features such as the PLP and the MFCC are based on the short term spectral analysis of speech signal. The generation of such speech features involves; a short analysis window of the speech signal (10-30 ms) followed by a Bark or a Mel scale integrator and an averaging the power spectral energy [68]. Such methods remain vulnerable to interference and degradation associated with the phone channel [13].

In this section we will look at deriving another set of robust speech features. These features are derived from speech that is analyzed or represented by its envelope and instantaneous frequency [13]; an idea was initial proposed by Gabor. Gabor suggested that the analytical signal is a suitable candidate for the envelope-instantaneous frequency decomposition of speech. The squared magnitude of the analytical signal represent the envelope and the phase component of the analytical signal represent the instantaneous frequency.

The characterization of speech into its envelope-instantaneous frequency is also referred as the AM-FM modeling. AM-FM modeling can also be considered as a sum of sinusoids of varying center frequency and amplitude [14]. For broadband signal such as speech before the AM-FM modeling it is necessary to break the signal into a series on narrow-band signal. This is done using a filter bank designed as a single resonator with two real poles and with the resonating frequency fixed at $\omega = \pi$. Further for each sub-band we adopt an auto-regressive modeling of the envelope or magnitude of amplitude modulation. This is referred to as the frequency domain linear prediction. All this is discussed in details in the coming section.

4.1 AM-FM based FDLP features

As stated above, AM-FM analysis involves modeling the signal as a sum of sinusoids with time varying center frequencies [14]. Thus the AM-FM modeling of a signal $x(t)$ is as shown below

$$x(t) = A(t)cos\phi(t) \tag{70}$$

where $A(t)$ and $\phi(t)$ represent the amplitude modulation and phase modulation respectively. As speech signal represent a broadband signal it is initially decomposed into a series of narrow band signals. The AM-FM analysis of each of the narrow-band signal is as shown below

$$x_i(t) = A_i(t)cos\phi_i(t) \tag{71}$$

where $A_i(t)$ and $\phi_i(t) = 2\pi(f_{c_i} + \int_0^t f_i(t)) + \phi_{i_0}$ and the amplitude and phase modulation component for the narrow band signal $x_i(t)$. Also f_{c_i} is the carrier frequency, $f_i(t)$ is the band-limited instantaneous frequency and ϕ_{i_0} is the initial

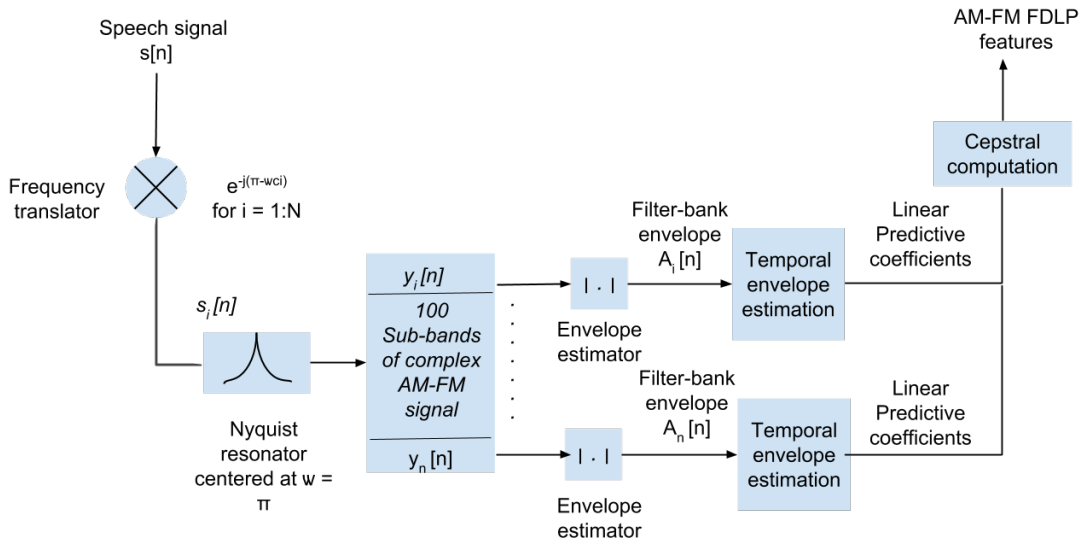


Figure 18: Block diagram for the generation of AM-FM FDLP features

phase of the i^{th} band. The overall speech signal is thus a summation of all the say N narrow band signal

$$x(t) = \sum_{i=0}^{N} x_i(t) \tag{72}$$

An important factor in the AM-FM analysis is the design of the filters that constitute the filter-bank. Here we consider the filter bank that consist of a single filter. This filter is a resonator with two real poles and with the resonating frequency fixed at the Nyquist frequency. The transfer function $H(z)$ and frequency response $H(\omega) = |H(\omega)| \exp^{j\theta(\omega)}$ of the filter with the two real poles at $z = -\beta$ is given by

$$\begin{aligned} H(z) &= \frac{1}{(1+\beta z^{-1})^2} \\ |H(\omega)| &= \frac{1}{1+2\beta cos(\omega)+\beta^2} \\ \theta(\omega) &= \left\{ -2tan^{-1}\frac{-\beta sin(\omega)}{1+\beta cos(\omega)} \right\} \end{aligned} \tag{73}$$

This filter as shown in Figure.20 has a smooth frequency response and the phase response is also linear as $\omega \longrightarrow \pi$. The bandwidth of the filter can also be controlled by varying the proximity of the poles from the unit circle as in Figure.19.

$$B = \frac{\alpha f_s log\beta}{\pi} \tag{74}$$

where B is the bandwidth, f_s is the sampling rate and $\alpha = (2^{\frac{1}{m}} - 1)^{1/2}$ with m as the number of poles. The bandwidth can also be controlled by increasing the number of poles at $\omega = \pi$.

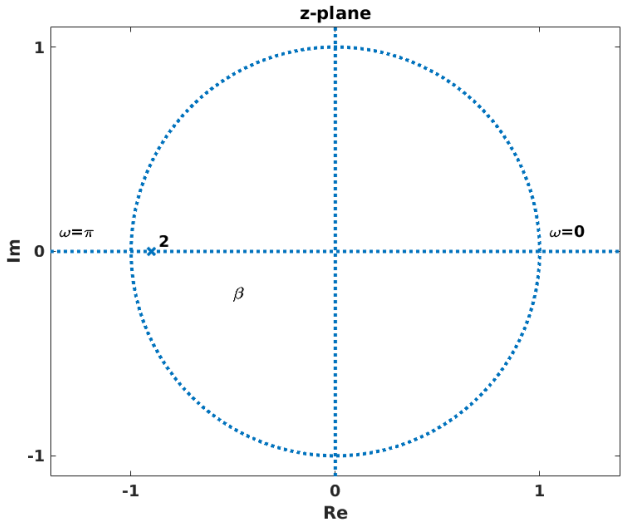


Figure 19: Pole-zero plot of the single resonator

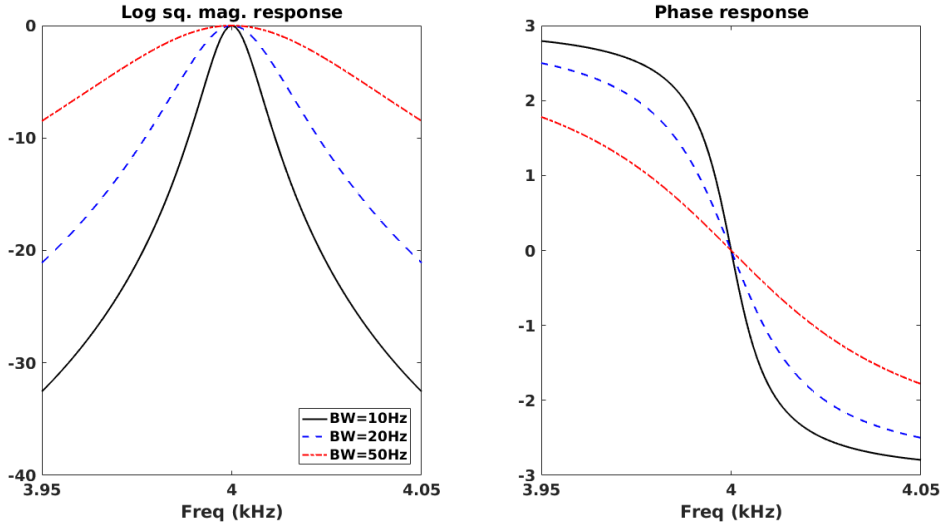


Figure 20: Frequency response of the resonator centered at $\omega = \pi$ for different values of bandwidth

The generation of the narrow band signals involves passing a frequency translated signal through the Nyquist filter.The filter bank divides the spectral range of the speech signal into 100 uniform sub-bands. For a speech signal $s[n]$ which is frequency translated as $s_i[n] = s[n]e^{-j(\pi-\omega_{c_i})n}$ where ω_{c_i} is the center frequency of the desired AM-FM component, narrow band signal obtained after passing through the Nyquist filter is given by

$$y_i = s_i[n] - 2\beta y_i[n-1] - \beta^2 y_i[n-2] \tag{75}$$

where like the complex input $s_i[n]$; the output is $y_i[n] = y_{i_r}[n] + jy_{i_i}[n]$. The discrete

AM-FM model of $y_{i_r}[n]$ as defined by equation(71) is given by

$$y_{i_r}[n] = A_i[n]cos\phi_i[n] \tag{76}$$

The imaginary part of $y_i[n]$ denotes the quadrature phase component of the narrow-band AM-AM signal. The AM component $A_i[n]$ is simply computed as

$$A_i[n] = (y_{i_r}^2[n] + y_{i_i}^2[n])^{1/2} \tag{77}$$

One of the requirement of the AM-FM analysis of a narrow band signal is that there should be large difference between the bandwidth of the AM component compared to the carrier frequency. Also, the bandwidth of the FM component is lesser than the carrier frequency. Thus, the single resonator filter with a resonating frequency fixed at the Nyquist frequency provides the best separation between the AM and FM component.

The next step in the feature extraction method involves approximating the AM component by an auto-regressive model. The auto-regressive modeling of the AM component involves finding a set of parameters a_j for $j = 1, ..., p$ such that in the least square sense

$$A_i[n] = \sum_{j=1}^{p} a_j A[n-j] \tag{78}$$

where p is the model order and is fixed at 40 poles per second.

The auto-regressive modeling of the AM component represent the dual of the linear prediction of the spectral domain as suggested by Kumaresan and Rao. A Fourier transform of the narrow-band signal is taken. A Fourier transform pair exist between the squared magnitude of the AM component and the auto-correlation of the real components of the Fourier transform of the narrow-band signal. Thus, the output at stage is the AM-FM FDLP based features

4.2 AM-FM 2D-AR features

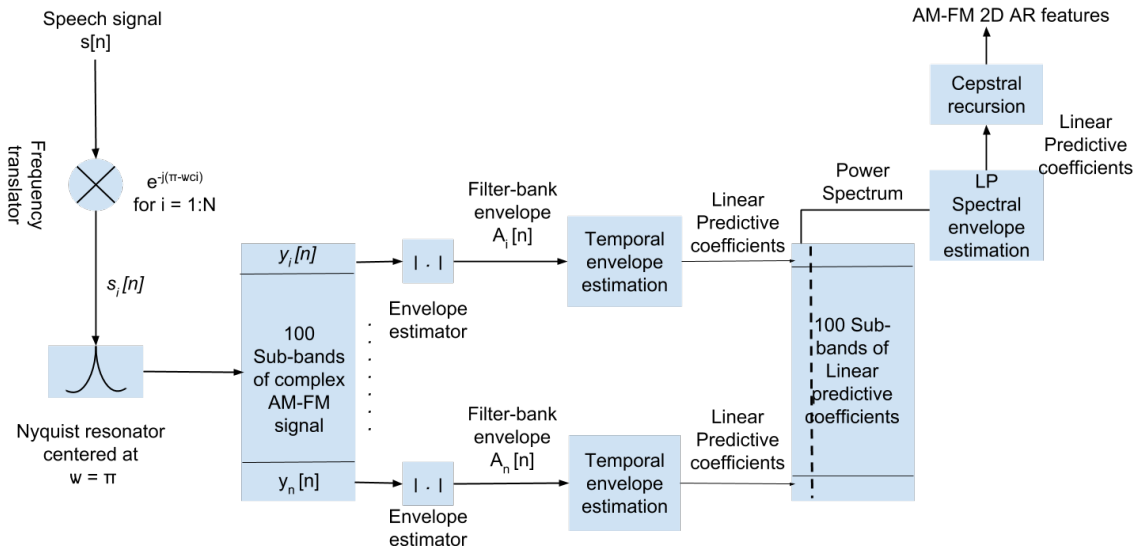


Figure 21: Block diagram for the generation of AM-FM 2D AR features

For the generation of the 2-D auto-regressive features, the FDLP model is further extended. The spectral envelope across each sub-band are converted to short-term energy estimates. Across the various sub-bands these energy estimates are then used as sampled power spectral estimate [15]. The power spectral estimates are inverse Fourier transformed to obtain the auto-correlation sequence. Now, the spectral envelope is model as an auto-regressive model as suggested by Kumaresan and Rao in [13]. The Time domain linear prediction provides an all-pole auto-regressive model of the 100 point power spectrum. The pole order is fixed at 12 and are later transformed to the 13 dimensional cepstral coefficients.

5 Experiments on phoneme classification

This section presents experimental comparison on the frame-wise phoneme classification. Under different noise condition,this experimental comparison is carried out between the robust AM-FM based FDLP features, the AM-FM 2D-AR features and the tradition features such as MFCC and PLP . Various neural network architecture are considered for the frame-wise classification of phonemes. Also considered in these experiments are the front end feature compensation methods.

Frame-wise phoneme classification is an example of a segment classification task as discussed in [35]. In a segment classification task, the positions of the frames measured in terms of the start and stop time are known in advanced. The co-articulation phenomenon seen in speech ensures that the position of a phoneme is dependent on the past as well as on the future phonemes. Essential to the performance of the classification task is the incorporation of the context around each frame. Thus, frames on either side of an input frame are collected and serves as input for the experiments.

In the context of frame-wise phoneme classification, an error measure referred as the frame error rate is considered. This frame error rate represents the number of misclassified segments or frames. For an frame-phoneme tagging pair $(\boldsymbol{x}, \boldsymbol{z})$ drawn from a set S, the frame error rate is defined as shown below

$$Error(h, S) = \frac{1}{Z} \sum_{(\boldsymbol{x}, \boldsymbol{z} \epsilon S)} HD(h(\boldsymbol{x}), \boldsymbol{z}) \tag{79}$$

where Z is the number of segments or frames and HD measures the hamming distance between the real phoneme label $\boldsymbol{z}$ and the predicted output label $h(\boldsymbol{x})$.

The coming section describes the experimental dataset, Theano the specialized python math library for building deep neural networks, the experimental set-up which include a reference of the different noise conditions, the data pre-processing, feature compensation methods used. Also described are the various network architecture along with the training methods. Finally the experimental results for the frame-wise phoneme classification is presented.

5.1 TIMIT dataset

The data used in the experiments is derived from the TIMIT corpus. This corpus contains recordings of ten phonetically rich sentences read out by each of the 630 speakers. These speakers represents eight major dialects of American English. The TIMIT corpus also include for each sentence a time-aligned phonetics and word transcription.

The TIMIT corpus is also partitioned into the training and the test set. Here we will consider only the core test set. The training and the core test set contains 3696 and 192 utterances respectively. No similar speaker or sentence exist in both the training and the test set. A total of 184 randomly chosen training sentences is set aside as the validation set.

5.2 Theano

Theano is a python library that allows to define, optimize and evaluate mathematical expressions involving mathematical arrays. In theano, mathematical expressions are stored as graphs of variables and operation which are pruned at compile time. This graph structure allows automatic computation of the symbolic differentiation of complex expressions, ignore the variables that are not required to compute the final output, reuse partial results to avoid redundant computations, apply mathematical simplifications, compute operations in place when possible to minimize the memory usage, and apply numerical stability optimization to overcome or minimize the error due to hardware approximations [69]. All the various neural network models used in the classification experiments are implemented in theano.

5.3 Experimental Set-up

5.3.1 Different noise conditions

In order to draw a comparison between the performance of the robust speech features and that of the traditional speech features frame-wise phoneme classification is performed. This experimental comparison is done under different noise types and at different signal-to-noise ratio(SNR). The different noise types include white, babble and factory noise. Signal-to-noise ratio of 0,10 and 20 dB is considered.

It should be mentioned that the training is done only with features derived from the clean signal across all of the models. Noise is added only to the test signal and not to the training signal.

5.3.2 Data pre-processing

Across all of the features derived from both the training and the test signal a similar frame or segment size of 25 ms with an overlap of 10 ms is considered. Hence, in all cases the number of frames in the training set and the test set are 1,124,823 and 57400 frames respectively.

The generation of the speech features entails a standard processing of the audio signal. Before the processing, the audio signal is first pre-emphasized by a factor of 0.97. All of the speech features are derived from this pre-emphasized signal. The speech features are characterized by a sequence of 26 vector coefficients. These vector coefficients include 13 static vector; 12 cepstrals coefficients and 1 log-energy and 13 dynamic feature vectors; which are the first derivatives of the static vectors.

The preprocessing step also include various feature compensation methods. These feature compensation methods include RASTA and cepstral mean variance normalization. Feature compensation is performed on both the training and the test data.

5.4 Various network architecture

The following network architecture are considered in our experiments:

1. Single layer perceptron with a hidden layer consisting of 250 sigmoid units.

2. Multi-layer perceptron with three hidden layers. Each hidden layer consist of 1000 units. The rectified linear(ReL) function serves as the activation function.

3. Multi-layer perceptron with drop-out units. The perceptron consist of three hidden layers each of 1000 units. Again, rectified linear function serves as the activation function.

4. Gaussian restricted Boltzmann machine with three layers of 1000 sigmoid units. Drop-out and momentum is implemented both in the pre-training and the fine-tuning phase. Also, a sigmoid activation function is used in both training phases.

5. Sparse de-noising auto-encoder consisting of three layers each of 1000 units. The pre-training involves usage of the soft-plus activation for encoding and sigmoid activation function for decoding with a sparse level of 0.5. The fine-tuning phase uses the sigmoid activation function.

6. Contractive de-noising auto-encoder consisting of three layers each of 1000 units. The pre-training involves usage of the soft-plus and sigmoid activation functionfor encoding and sigmoid activation function and a contraction level of 0.2. The auto-encoder is further fine-tuned considering a sigmoid activation function.

All the network models that we have considered consist of an input layer of 546 units. These 546 units are able to accommodate the 21 features frames that serve as input. Also, besides the single layer perceptron, all the other network models consist of three hidden layers each consisting of 1000 units. Also, the final layer for all network models consist of 61 units each for the 61 phonemes.

For the network models such as the Gaussian RBM and the Auto-encoder that require pre-training. The weights are initialized with random values sampled from uniform distribution. This distribution is set between a range of $[-\sqrt{\frac{6}{n_{hid}+n_{vis}}}, \sqrt{\frac{6}{n_{hid}+n_{vis}}}]$ with n_{hid}, n_{vis} being the number of hidden and visible unit one layer at a time. The weights of the other models such as the multi-layered perceptron are initialized to random values that are sampled from a standard normal distribution with the standard deviation set to 0.01. The bias of all of the networks are initialized to 0. Network model can consider dropout training have a dropout rate fixed at 0.2 for all the layers.

Table 1: Performance of the various models measures in terms of accuracy (%) using MFCC feature with no compensation

	clean	w-20	b-20	f-20	w-10	b-10	f-10	w-0	b-0	f-0
Single layer perceptron	58.12	47.62	51.09	51.87	35.70	40.07	41.21	23.99	22.73	23.52
MLP with Rel activation	57.00	45.91	48.44	49.78	33.94	35.89	38.08	21.76	19.14	20.68
MLP with Rel and dropout	56.67	47.83	49.12	50.16	36.94	37.97	39.58	23.32	20.43	21.95
Gaussian RBM with dropout	61.23	50.10	52.23	53.74	37.21	39.72	42.14	24.31	21.40	23.22
Contractive Auto-encoder	58.60	47.41	50.03	51.32	35.01	36.93	39.28	22.31	19.57	20.95
Sparse Auto-encoder	56.40	47.29	49.99	50.96	34.32	36.62	38.88	21.33	19.05	20.28

5.5 Training methods adopted

All the networks are trained using features derived from a clean training signal. Neural networks such as the Gaussian RBM and the auto-encoder undergo an initial pre-training. The Gaussian RBM are pre-trained to increase the data likelihood. As, we are using the contrastive divergence (CD-1) during training, we monitor the progress of the training using the cross-entropy between the input and the reconstructed output. De-noising auto-encoders and their variants are pre-trained considering a mean square error cost function. Additional penalty term such as the Frobenius norm and a sparsity factor is added to the cost function of the contractive and spare de-noising auto-encoder respectively.

The pre-training networks are further fine-tuned as multi-layered perceptron. In the the training of the multi-layered perceptron and in the fine-tuning of the RBM's and the auto-encoders we consider a batch stochastic gradient algorithm with a cross-entropy cost function defined between the real output and the predicted output. The batch size of is fixed at 60. An initial learning rate of 0.1 with a momentum of 0.9 and a learning rate decay of 0.998 is used. The inputs are randomized after each epoch and to prevent over-fitting an early stopping method is also included.

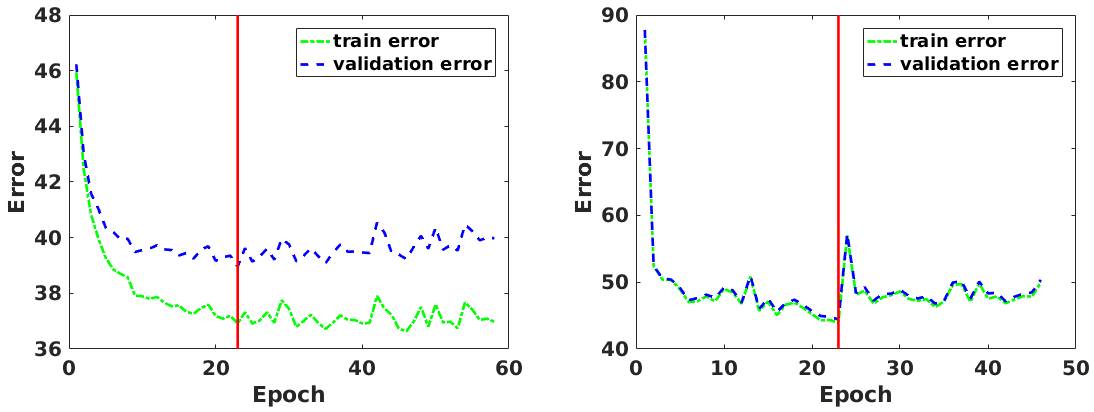
5.6 Results

The following frame-wise classification results are tabulated as the accuracy or (100%− segment classification error) with the segment classification error defined by equation(79). Further, these classification results are reported on the lowest error on the validation set .

5.6.1 Experiments on different neural network architectures

We begin by evaluating the performance of the various models using the MFCC features with no feature compensation. The results are as shown in Table.1

If we were to divide the models presented in Table.1 into two groups. One group will include models that undergo an initial pre-training and another group such as the multilayer perceptron which does not undergo any pre-training. Let us take a



(a) Single layer perceptron (b) MLP with Rel activation and dropout

Figure 22: Training and validation error in (%)for (a) single layer perceptron and (b) Multi-layer perceptron with dropout using MFCC features. The lowest point of the validation error is shown by the intersection of the red line and the trace of the validation error

closer look at the training patterns of these two groups of models with reference to the issue of learning and generalization that is discussed in section 2.3.4.

The two figures shown in Figure.22 represent the training pattern seen in neural network models that belong to the later group such as the single layer perceptron and the MLP with dropout. As can be seen from Figure.22 such models have appropriate generalization and in the case of the MLP with dropout the validation error is seen to follow the training error. However, it is important to note that these two models only under-fit the training data with the training and validation error oscillating within the certain values of the training and validation error. Another model belonging to this group is the MLP without dropout whose training pattern is as shown in Figure.23. In this model of the MLP without the dropout the network can be seen to easily over-fit the training data. Coming back to Table.1 we can see that the best performing model for this group of models is the single layer perceptron followed by the MLP with dropout under most noise and SNR condition.

Neural network models that undergo an initial pre-training have improved initialization of the network parameters such as the weights. This initialization of the network parameters in turn has resulted in models that no longer settle to a poor local minima. As shown in Figure.24 these models are able to train without over-fitting for a greater number of epochs compared to the MLP based models. Also, as seen from Table.1 the best performance of the MFCC features with no feature compensation under different noise and SNR conditions is achieved using the Gaussian RBM model.

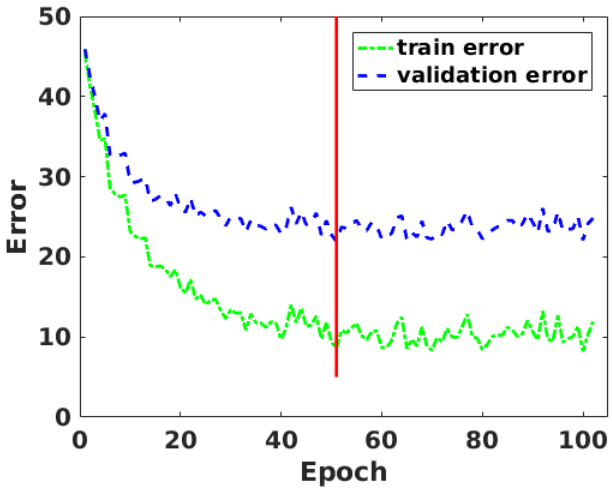
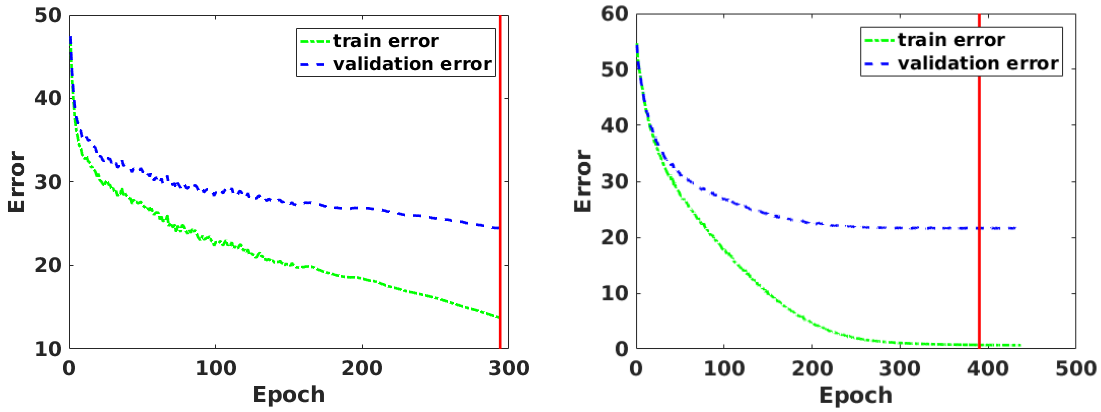


Figure 23: Training and validation error in (%)for MLP with Rel activation.The lowest point of the validation error is shown by the intersection of the red line and the trace of the validation error



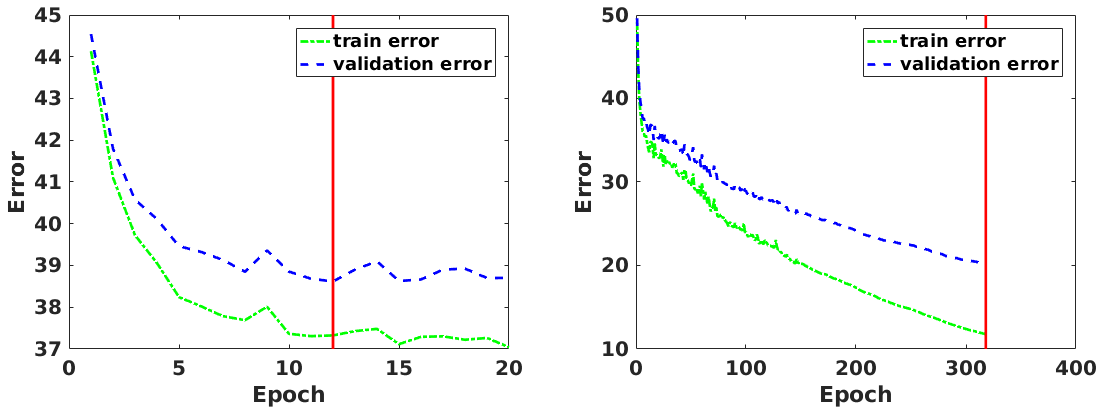
(a) Gaussian RBM with dropout

(b) Contractive auto-encoder

Figure 24: Training and validation error in (%) for (a) Gaussian RBM with dropout and (b) Contractive auto-encoder with dropout using MFCC features.The lowest point of the validation error is shown by the intersection of the red line and the trace of the validation error

5.6.2 Experiment with the different features without feature compensation

A similar training pattern of the models with respect to the issue of learning and generalization is also seen when using different feature vectors such as the AM-FM FDLP as can be seen from Figure.25.



(a) Single layer perceptron (b) Gaussian RBM with dropout

Figure 25: Training and validation error in (%) for (a) Single layer perceptron and (b) Gaussian RBM with dropout using AM-FM FDLP features.The lowest point of the validation error is shown by the intersection of the red line and the trace of the validation error

Let us now see the result of using different features under different noise and SNR condition using the two best models one from each group of neural network *i.e.* single layer perceptron and the Gaussian RBM with dropout.

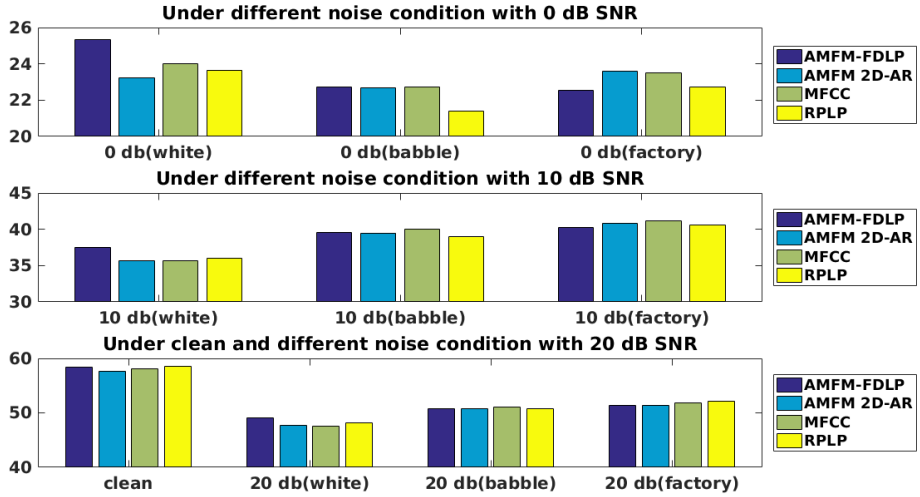


Figure 26: Accuracy in (%) for 61 phonemes under various noise and SNR conditions with a single layer perceptron model for classification

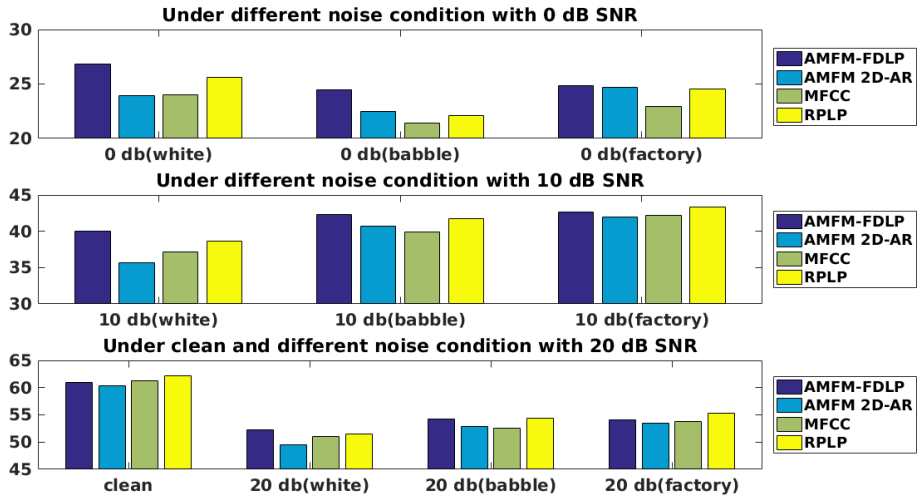


Figure 27: Accuracy in (%) for 61 phonemes under various noise and SNR conditions with a Gaussian RBM with dropout model for classification

From Figure.26 and 27 we can see that under white noise condition and 0 dB SNR the AM-FM FDLP features is seen to have a higher segment phoneme accuracy compared to the other features. This is more prominent in the case of the Gaussian RBM with dropout model and it also extended to other types of noise condition such as the babble and the factory noise. At a greater SNR of 10 dB and 20 dB the results are mix and different feature perform better under different noise and SNR conditions. The same cannot be said in the case of using single layer perceptron. It is true that when using a single layer perceptron the AM-FM FDLP feature performs well under white noise condition and 0 dB SNR. However, under babble noise condition the segment phoneme accuracy is similar to that of using MFCC features. Also, under factory noise condition the AM-FM 2D AR features performs better than the AM-FM FDLP features with the MFCC in the second place. Again, with higher SNR no unanimous decision can be said as to which is the best feature. It largely depends on the noise type and SNR values.

Experiments on phoneme classification for a reduced phoneme set

Now, we will consider collapsing the phonemes from 61 to 39. In order to do this we consider the folding table shown in Table.2

Table 2: Folding the 61 phonemes to 39 as shown in [70]. The phonemes on the right are folded to the phonemes on the left

aa	aa,ao
ah	ah,ax,ax-h
er	er,axr
hh	hh,hv
ih	ih,ix
l	l,el
m	m,em
n	n,en,nx
ng	ng,eng
sh	sh,zh
sil	pcl,tcl,kcl,bcl,dcl,gcl,h # , pau,epi
uw	uw,ux
-	q

Folding has resulted in an increase in accuracy across all the features. For example, with 61 phonemes the accuracy under white noise and 0 dB SNR are 22.93 and 24.85 (%) for the MFCC and AM-FM FDLP features respectively. However, when 39 phonemes are considered the accuracy considering the MFCC and AM-FM FDLP features is reportedly 34.09 and 33.60 (%)respectively. While these is an increase in the accuracy this is not consistent across all of the features. There is greater improvement in the phoneme classification rate of MFCC features compared to the other features with 39 phonemes. A comprehensive view of all the features under various test conditions using the single layer perceptron is shown when we consider 39 phonemes is as shown in Figure.28

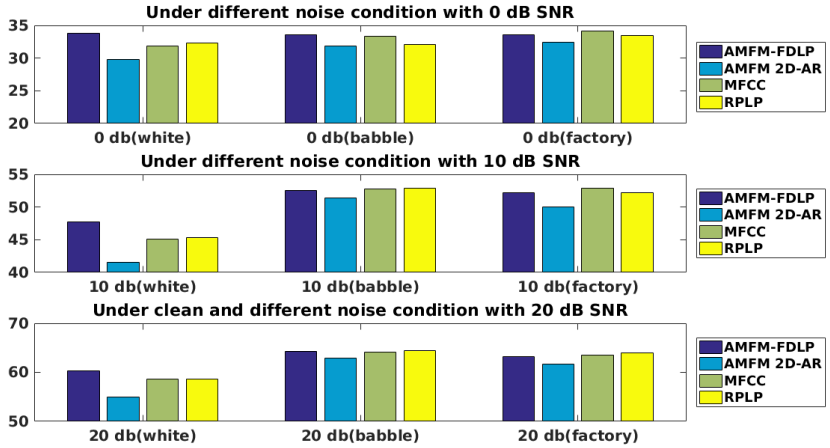


Figure 28: Accuracy in (%) with 39 phonemes under various noise conditions using single layer perceptron

A similar scenario can also observed when using a Gaussian RBM model. In this case also there is a difference in the pattern observed with respect to the accuracy with 39 phonemes. This is given comprehensively in Figure.29.

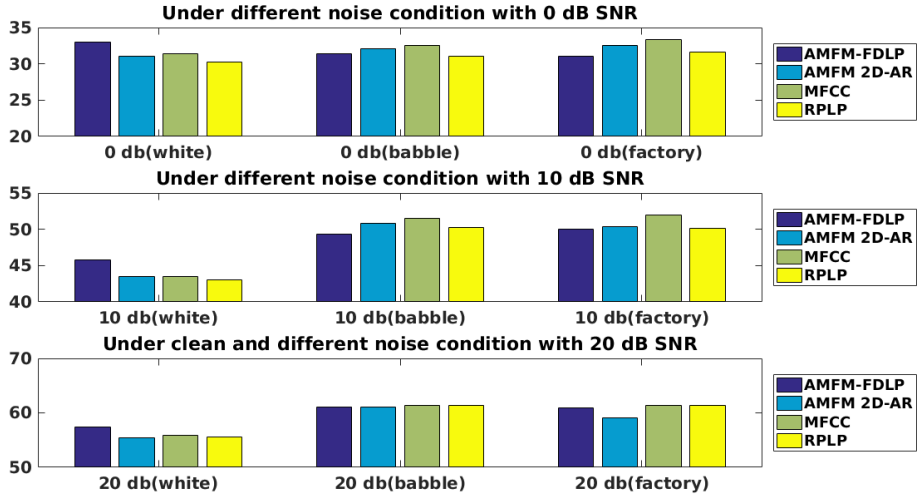
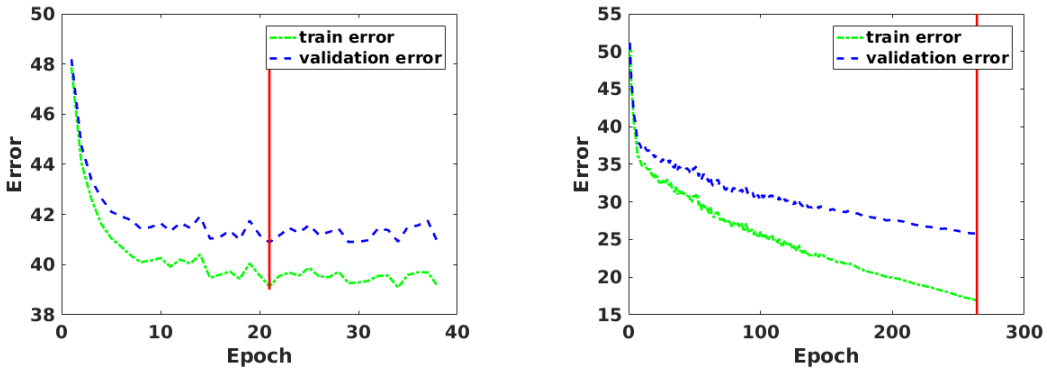


Figure 29: Accuracy in (%) with 39 phonemes under various noise conditions using Gaussian RBM

5.6.3 Experiment with the different features with RASTA feature compensation

This next section we consider a RASTA compensation for each of the features. Also we will again consider the two neural network model namely the single layer perceptron and the Gaussian RBM with dropout.



(a) Single layer perceptron (b) Gaussian RBM

Figure 30: Training and validation error in (%) of the (a) Single layer perceptron and (b) Gaussian RBM with dropout and with RASTA compensated MFCC features.The lowest point of the validation error is shown by the intersection of the red line and the trace of the validation error

As we can see from Figure.30, the training pattern for both these models is similar to the once seen when no RASTA compensation method was used. Let us now compare the accuracies of each of the features under different noise condition using these two models

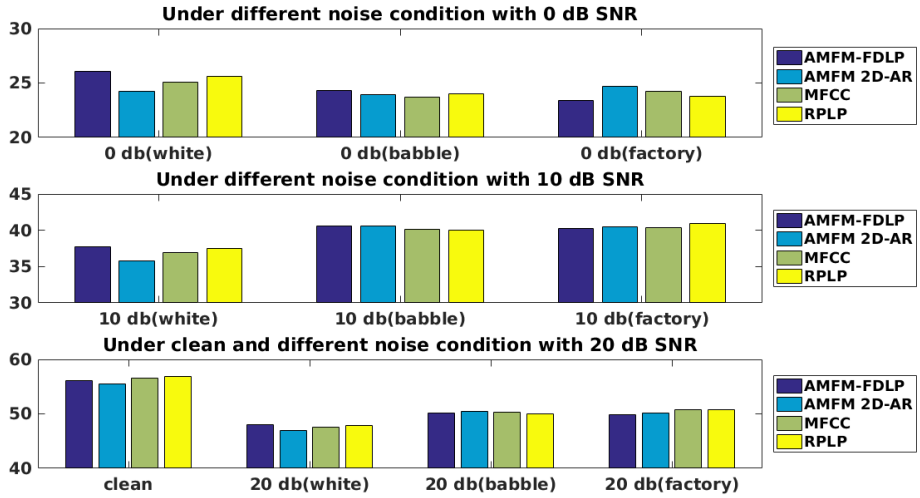


Figure 31: Accuracy in (%) with 61 phonemes under various noise conditions using single layer perceptron

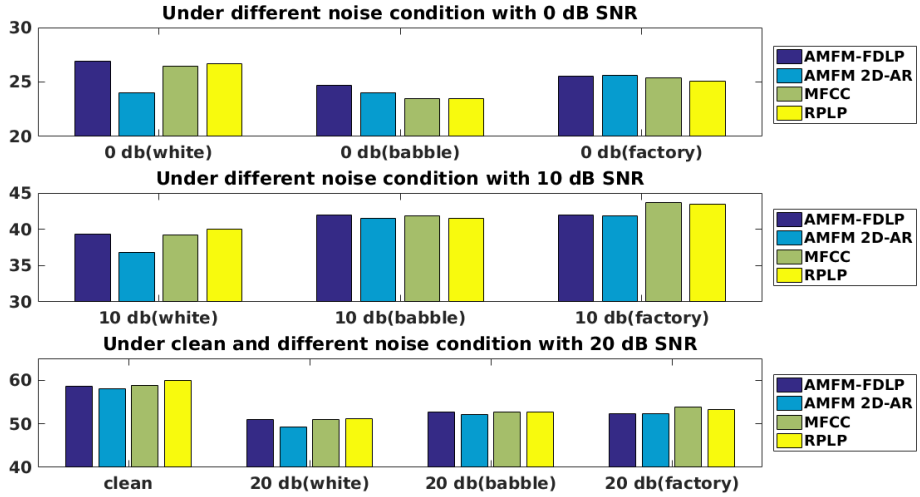


Figure 32: Accuracy in (%) with 61 phonemes under various noise conditions using Gaussian RBM with dropout

As we can see from Figure.31 and 32 the AM-FM FDLP features perform better under all noise type with 0 dB SNR. For greater SNR values of 10 dB and 20 dB the results are ambigious and different features perform better at different noise and SNR conditions.

Experiments on phoneme classification for a reduced phoneme set

Further folding the phonemes to 39 phonemes as done in the case where the features are not RASTA compensated. The results are tabulated in the Figure.33 and 34.

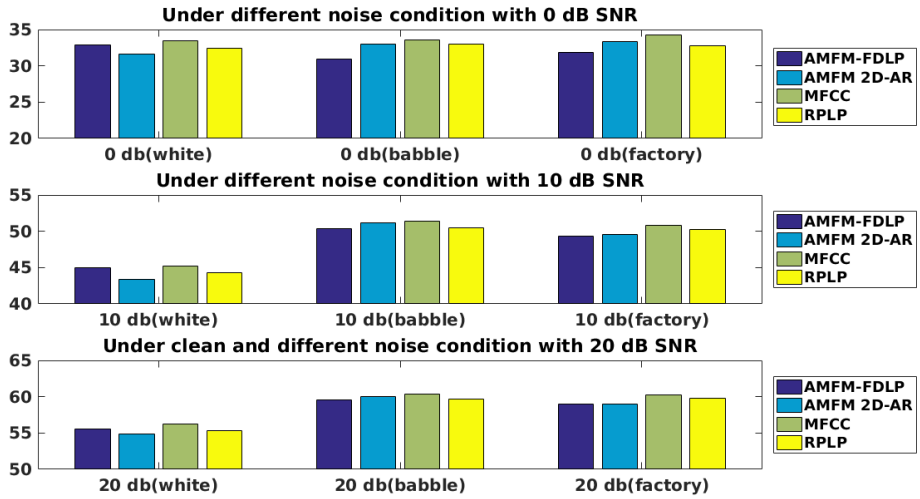


Figure 33: Accuracy in (%) with 39 phonemes under various noise conditions using single layer perceptron

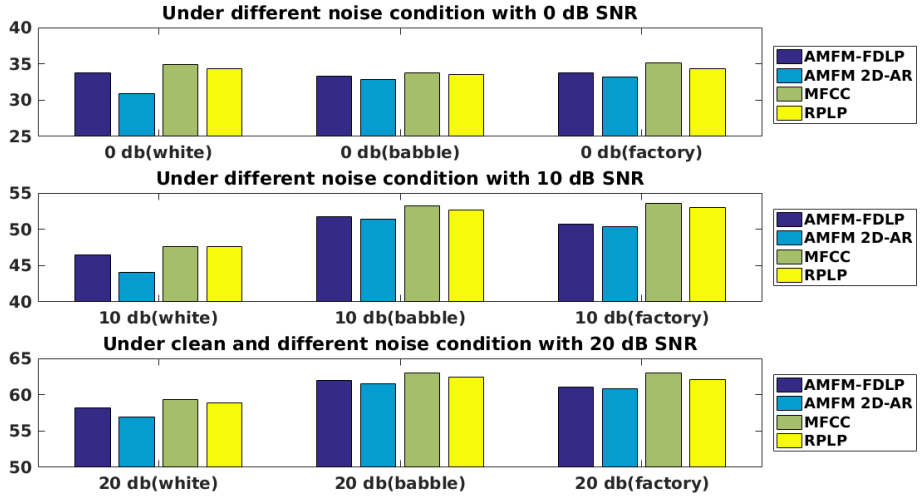


Figure 34: Accuracy in (%) with 39 phonemes under various noise conditions using Gaussian RBM with dropout

An interesting observation seen with the reduced phoneme set as seen in Figure.33 and 34 is that the MFCC features with RASTA compensation is seen to improve considerable especially under 0 dB SNR conditions. It is seen to outperform even the AM-FM FDLP features under 0 dB SNR for all types of noise conditions.

6 Summary

This thesis has focused on the functionality of many neural networks models and also on the robustness of features derived using AM-FM analysis of the speech signal, with regards to segment phoneme classification. The neural network models included are the single layer perceptron, multi-layer perceptron with and without dropout which do not involve any initial pre-training. Also considered are neural networks that are pre-trained as restricted Boltzmann machines and auto-encoder. The main objective in this thesis involves drawing a comparison between the performance of the traditional and AM-FM based features with regard to segment phoneme classification under different noise conditions. AM-FM based features involve modeling the signal as a sum of sinusoids of varying amplitude and phase. Unlike traditional features that are based on short-term spectral analysis in the form of linear prediction, cepstral analysis these AM-FM based features are resistant to interference and channel distortions. Also, considered in the are feature compensation techniques and their effect on phoneme classification results under different noise conditions.

Both the traditional features and the AM-FM based features are derived using speech samples from the TIMIT dataset. We consider the same frame length and frame shift for the extraction of all of the features. Also, all the features are represented by the same number of feature coefficients. We consider context dependency and classification of 61 phonemes. Further the number of phonemes is also collapse to 39 phonemes. Also, the experiments are performed considering different noise types such as white noise and signal-to-noise ratio(SNR) of 0, 10 and 20 dB.

We began with the set of experiments considering no RASTA filtering but only cepstal mean and variance normalization(CMVN) feature compensation on the training and the test data. Using the MFCC input feature all the various network models are made to perform segment phoneme classification. The experiment showed that there is a difference in the training pattern associated with the two groups of neural network models. With the early-stopping criterion set on all these networks, it is seen that networks such as the Gaussian restricted Boltzmann machine that undergo initial pre-training are trained for many more epochs compared to their counter-parts such as the multi-layered perceptron which does not undergo initial pre-training. Taking this difference in training patterns into perspective, two model one from each group is selected based on the phoneme error rate. Thus, the model selected include the single layer perceptron and the Gaussian restricted Boltzmann machine. These two models are further used for the phoneme classification with the other input features. The other set of experiments involves using RASTA filtering along with CMVN on all of the features for both the clean training data and the data derived from the different test conditions.

The results shows that with 61 phonemes under both condition of no RASTA and RASTA compensation the AM-FM FDLP feature is seen to perform better than the other features. This is especially accurate under white noise and 0 dB SNR conditions. The result is inconclusive for other noise types and at higher SNR. When the phonemes are folded to 39 phonemes, under both no compensation and feature compensation conditions compared to the other features there is an greater increase

in the accuracy with respect to MFCC features. The MFCC features are seen to outperform the AM-FM FDLP features when using both the single layer perceptron and the Gaussian RBM model network models.

7 Scope for future works

In this thesis we have incorporated context dependencies by stacking together adjacent frames. These stacked frames later serves as input to the neural network. An optimal choice of the number of adjacent frames is elusive and task dependent. Also, by merely stacking the frames we devoid the neural network of its ability to learn and adapt to shifted and time warped frames. Thus, as a future work we can consider using recurrent neural networks discussed in [35] as they able to process the frames in a temporal order. A further improvement would be to use a bi-directional recurrent network as discussed in [34] to incorporate context dependencies from the previous as well as the future frames. These bi-directional networks can also be further improved by including Long-short term memory(LSTM) with the bi-directional networks for larger context dependencies. Lastly, we can also scale the neural network to be become part of a hybrid neural network/ Hidden Markov model for complete speech recognition.

References

[1] Priscilla Lau. The lombard effect as a communicative phenomenon. *UC Berkeley Phonology Lab Annual Reports*, 2008.

[2] J Rajnoha. Multi-condition training for unknown environment adaptation in robust asr under real conditions. *Acta Polytechnica*, 49(2), 2009.

[3] ETSI Speech Processing. Transmission and quality aspects (stq); distributed speech recognition; advanced front-end feature extraction algorithm; compression algorithms. *ETSI ES*, 202:050, 2002.

[4] Chanwoo Kim and Richard M Stern. Feature extraction for robust speech recognition using a power-law nonlinearity and power-bias subtraction. In *INTERSPEECH*, pages 28–31, 2009.

[5] Martin Cooke, Andrew Morris, and Phil Green. Missing data techniques for robust speech recognition. In *Proc. Int. Conf. Acoustics Speech and Signal Processing*, volume 2, pages 863–866. IEEE, 1997.

[6] Jason Pelecanos and Sridha Sridharan. Feature warping for robust speaker verification. *International Speech Communication Association*, pages 213–218, 2001.

[7] Hynek Hermansky and Nelson Morgan. Rasta processing of speech. *IEEE transactions on speech and audio processing*, 2(4):578–589, 1994.

[8] Aaron E Rosenberg, Chin-Hui Lee, and Frank K Soong. Cepstral channel normalization techniques for hmm-based speaker verification. In *Proc. Int. Conf. Acoustics Speech and Signal Processing*, volume 4, pages 1835–1838, 1994.

[9] Christopher J Leggetter and Philip C Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech & Language*, 9(2):171–185, 1995.

[10] J-L Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298, 1994.

[11] Erik McDermott, Timothy J Hazen, Jonathan Le Roux, Atsushi Nakamura, and Shigeru Katagiri. Discriminative training for large-vocabulary speech recognition using minimum classification error. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):203–223, 2007.

[12] LR Bahl, Peter F Brown, Peter V De Souza, and Robert L Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proc. Int. Conf. Acoustics Speech and Signal Processing*, volume 86, pages 49–52, 1986.

[13] Ramdas Kumaresan and Ashwin Rao. Model-based approach to envelope and positive instantaneous frequency estimation of signals with speech applications. *The Journal of the Acoustical Society of America*, 105(3):1912–1924, 1999.

[14] Dhananjaya N. Gowda, Rahim Saeidi, and Paavo Alku. AM-FM based filter bank analysis for estimation of spectro-temporal envelopes and its application for speaker recognition in noisy reverberant environments. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 1166–1170, 2015.

[15] Sriram Ganapathy, Samuel Thomas, and Hynek Hermansky. Feature extraction using 2-d autoregressive models for speaker recognition. In *Odyssey*, pages 229–235, 2012.

[16] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[17] Michael L Seltzer, Dong Yu, and Yongqiang Wang. An investigation of deep neural networks for noise robust speech recognition. In *Proc. Int. Conf. Acoustics Speech and Signal Processing*, pages 7398–7402. IEEE, 2013.

[18] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *INTERSPEECH*, pages 436–440, 2013.

[19] Jared J Wolf. Efficient acoustic parameters for speaker recognition. *The Journal of the Acoustical Society of America*, 51(6B):2044–2056, 1972.

[20] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Foreword By-Reddy. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR, 2001.

[21] Xiao Xiong. Robust speech features and acoustic models for speech recognition. *Nanyang Technological University, PhD thesis*, 2009.

[22] MA Anusuya and SK Katti. Front end analysis of speech recognition: a review. *International Journal of Speech Technology*, 14(2):99–145, 2011.

[23] Joseph W Picone. Signal modeling techniques in speech recognition. *Proceedings of the IEEE*, 81(9):1215–1247, 1993.

[24] Dan Jurafsky. Speech recognition and synthesis, linguistic summer institute. http://nlp.stanford.edu/courses/lsa352/lsa352.lec6.ppt, 2007.

[25] National Instruments. Understanding ffts and windowing. http://www.ni.com/white-paper/4844/en/, 2015.

[26] Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.

[27] D Raj Reddy. Speech recognition by machine: A review. *Proceedings of the IEEE*, 64(4):501–531, 1976.

[28] Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.

[29] Lasse Holmstrom, Petri Koistinen, Jorma Laaksonen, and Erkki Oja. Neural and statistical classifiers-taxonomy and two case studies. *IEEE Transactions on Neural Networks*, 8(1):5–17, 1997.

[30] Max Welling. Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto*, 3:1–4, 2005.

[31] Šarūnas Raudys. Evolution and generalization of a single neurone: I. single-layer perceptron as seven statistical classifiers. *Neural Networks*, 11(2):283–296, 1998.

[32] Raul Rojas. *Neural Networks A Systematic Introduction.* Springer-Verlag, Berlin, New-York, 1996.

[33] Michael D Richard and Richard P Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483, 1991.

[34] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[35] Alex Graves. Neural networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 15–35. Springer, 2012.

[36] Filippo Amato, Alberto López, Eladia María Peña-Méndez, Petr Vaňhara, Aleš Hampl, and Josef Havel. Artificial neural networks in medical diagnosis. *Journal of applied biomedicine*, 11(2):47–58, 2013.

[37] Mia Louise Westerlund. Classification with kohonen self-organizing maps. *Soft Computing, Haskoli Islands*, 2005.

[38] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10(Jan):1–40, 2009.

[39] James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.

[40] Dandan Mo. A survey on deep learning: one small step toward ai. *Dept. Computer Science, Univ. of New Mexico, USA*, 2012.

[41] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[42] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[43] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[44] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[45] Andrew Ng. Deep learning and unsupervised feature learning. https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf, 2011.

[46] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.

[47] Fu-qiang Chen, Yan Wu, Guo-dong Zhao, Jun-ming Zhang, Ming Zhu, and Jing Bai. Contractive de-noising auto-encoder. In *International Conference on Intelligent Computing*, pages 776–781. Springer, 2014.

[48] Matthew D Zeiler, M Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al. On rectified linear units for speech processing. In *Proc. Int. Conf. Acoustics Speech and Signal Processing*, pages 3517–3521. IEEE, 2013.

[49] Theano Development Team. Deep learning 0.1 documentation. http://deeplearning.net/tutorial/rbm.html, 2011.

[50] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[51] R Rojas. The curse of dimensionality. http://www.inf.fuberlin.de/inst/agki/rojas_home/documents/tutorials/dimensionality.pdf, 2015.

[52] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[53] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[54] Jinyu Li, Li Deng, Yifan Gong, and Reinhold Haeb-Umbach. An overview of noise-robust automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):745–777, 2014.

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

[56] Matti Karjalainen. Auditory models for speech processing. *Proc. Int. Congr. Phon. Sciences. Tallinn*, 1987.

[57] Florian Hönig, Georg Stemmer, Christian Hacker, and Fabio Brugnara. Revising perceptual linear prediction (plp). In *INTERSPEECH*, pages 2997–3000, 2005.

[58] Chanwoo Kim. *Signal processing for robust speech recognition motivated by auditory processing.* PhD thesis, Johns Hopkins University, 2010.

[59] Chanwoo Kim and Richard M Stern. Feature extraction for robust speech recognition using a power-law nonlinearity and power-bias subtraction. In *INTERSPEECH*, pages 28–31, 2009.

[60] Hynek Hermansky, Daniel PW Ellis, and Sangita Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *Proc. Int. Conf. Acoustics Speech and Signal Processing. Proceedings*, volume 3, pages 1635–1638. IEEE, 2000.

[61] Barry Y Chen, Shuangyu Chang, and Sunil Sivadas. Learning discriminative temporal patterns in speech: development of novel traps-like classifiers. In *INTERSPEECH*, 2003.

[62] Frantisek Grézl, Martin Karafiát, Stanislav Kontár, and Jan Cernocky. Probabilistic and bottle-neck features for lvcsr of meetings. In *Proc. Int. Conf. Acoustics Speech and Signal Processing*, volume 4, pages IV–757. IEEE, 2007.

[63] Michael L Seltzer, Dong Yu, and Yongqiang Wang. An investigation of deep neural networks for noise robust speech recognition. pages 7398–7402. IEEE, 2013.

[64] John P Openshaw and JS Masan. On the limitations of cepstral features in noise. In *Proc. Int. Conf. Acoustics Speech and Signal Processing*, volume 2, pages II–49. IEEE, 1994.

[65] N Vishnu Prasad and Srinivasan Umesh. Improved cepstral mean and variance normalization using bayesian framework. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 156–161. IEEE, 2013.

[66] Luz García, Jose Carlos Segura, A de la Torre, Carmen Benítez, and A Rubio. Histogram equalization for robust speech recognition. *Speech Recognition, France Mihelic and Janez Zibert (Ed.)*, 2:174, 2008.

[67] Yifan Gong. Speech recognition in noisy environments: A survey. *Speech communication*, 16(3):261–291, 1995.

[68] Sriram Ganapathy, Samuel Thomas, and Hynek Hermansky. Comparison of modulation features for phoneme recognition. In *Proc. Int. Conf. Acoustics Speech and Signal Processing*, pages 5038–5041. IEEE, 2010.

[69] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[70] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. Phoneme recognition in timit with blstm-ctc. *arXiv preprint arXiv:0804.3269*, 2008.