



TAMPERE UNIVERSITY OF TECHNOLOGY

OGUZHAN GENCOGLU
ACOUSTIC EVENT CLASSIFICATION USING DEEP NEURAL
NETWORKS

Master's Thesis

Examiners: Adj. Prof. Tuomas Virtanen

Dr. Eng. Heikki Huttunen

Examiners and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 4 September 2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Degree Programme in Information Technology

GENCOGLU, OGUZHAN: Acoustic Event Classification Using Deep Neural Networks

Master of Science Thesis, 62 pages

January 2014

Major subject: Signal processing

Examiners: Adj. Prof. Tuomas Virtanen, Dr. Eng. Heikki Huttunen

Keywords: acoustic event classification, artificial neural networks, audio information retrieval, deep neural networks, deep belief networks, pattern recognition

Audio information retrieval has been a popular research subject over the last decades and being a subfield of this area, acoustic event classification has a considerable amount of share in the research. In this thesis, acoustic event classification using deep neural networks is investigated. Neural networks have been used in several pattern recognition (both function approximation and classification) tasks. Due to their stacked, layer-wise structure they have been proved to model highly nonlinear relations between inputs and outputs of a system with high performance. Even though several works imply an advantage of deeper networks over shallow ones in terms of recognition performance, advancements in training deep architectures were encountered only recently. These methods excel conventional methods such as HMMs and GMMs in terms of acoustic event classification performance.

In this thesis, effects of several NN classifier parameters such as number of hidden layers, number of units in hidden layers, batch size, learning rate etc. on classification accuracy are examined. Effects of implementation parameters such as types of features, number of adjacent frames, number of most energetic frames etc. are also investigated. A classification accuracy of 61.1% has been achieved with certain parameter values. In the case of DBNs, An application of greedy, layer-wise, unsupervised training before standard supervised training in order to initialize network weights in a better way, provided a 2-4% improvement in classification performance. A NN that had randomly initialized weights before supervised training was shown to be considerably powerful in terms of acoustic event classification tasks compared to conventional methods. DBNs have provided even better classification accuracies and justified its significant potential for further research on the topic.

PREFACE

This thesis work has been conducted at the Department of Signal Processing, in Tampere University of Technology, Finland.

In the first place, I would like to express my gratitude to my supervisors Tuomas Virtanen and Heikki Huttunen. Their invaluable guidance and generous interest not only enabled this work possible, but also made the whole process attractive and remarkably fun.

Moreover, I wish to express my appreciation to the members of Audio Research Team. Their supportive attitude inspired me scientifically and in every other aspect of life. It has been a pleasure to work with you.

This work would have required twice as much coffee without my friends. Thank you for setting my mind in ease by making music with me. Thank you for keeping me alive by rowing with me in the mist of the morning.

Finally, I owe my thankfulness to my family. Without their sheer support, undoubtedly, I would not be who I am now.

Oguzhan Gencoglu

Tampere, January 2014

CONTENTS

1. Introduction	1
1.1 Acoustic Pattern Recognition.	1
1.2 Neural Networks	2
1.3 Deep Architectures	2
1.4 Objectives of the Thesis	3
1.5 Results of the Thesis	3
1.6 Structure of the Thesis	3
2. Theoretical Background	4
2.1 Pattern Recognition	4
2.1.1 Learning Paradigms.	4
2.1.2 Structure of a Pattern Classification System	6
2.1.3 Methods of Evaluation	7
2.1.4 Sources of Error.	9
2.2 Acoustic Event Classification	10
2.2.1 Features Used in AEC	10
2.2.2 Classifiers Used in AEC	13
2.3 Neural Networks	14
2.3.1 The Single Neuron	14
2.3.2 Network Structures	16
2.3.3 Reasons for Using Neural Networks	18
2.3.4 Training an ANN	18
2.4 Deep Belief Networks	19
2.4.1 Need for DBNs	19
2.4.2 DBN Learning	19
3. Methodology	21
3.1 Pre-processing	21
3.2 Feature Extraction	22
3.3 Division of Training, Validation and Test Data	22
3.4 Training Algorithms	25
3.4.1 Backpropagation Algorithm	25
3.4.2 DBN Training	29
3.5 Classifier	31
4. Evaluation.	33
4.1 Data and Platform.	33
4.2 Evaluation Setup	35

4.3 Results for NNs with randomly initialized weights	36
4.3.1 Effect of network topology.	36
4.3.2 Effect of batch size.	42
4.3.3 Effect of number of adjacent frames	44
4.3.4 Effect of feature extraction.	46
4.4 Results for NNs with DBN pretraining	47
5. Discussion and Conclusions	54
6. References.	56

LIST OF SYMBOLS AND ABBREVIATIONS

α	constant to determine the slope of a sigmoid function, page 15
δ_k	sensitivity for unit k , page 27
η	learning rate of the BP algorithm, page 27
\mathcal{E}	mean square error at the output of an ANN, page 27
φ	NN activation function, page 14
τ	bias term in neural activation, page 14
\mathbf{b}_k	visible units offset vector for RBM, page 30
c	total number of distinct classes, page 5
\mathcal{C}_j	class label associated with an index j , page 5
c_k	k^{th} MFCC, page 11
\mathbf{c}_k	hidden units offset vector for RBM, page 30
d	dimension of feature vector, page 4
\mathcal{D}	set of data, page 8
E_i	log energy within each mel band, page 11
f	frequency in standard scale, page 11
f	feature, page 4
f_{mel}	frequency in mel scale, page 11
\mathbf{h}^k	k^{th} hidden layer of a DBN, page 30
l	total number of hidden layers of a DBN, page 30
m	total number of misclassified observations, page 8
n_j	total number of observations belonging to the class \mathcal{C}_j , page 8
N	frame length, page 21
N	number of mel band filters, page 11
o_i	observation associated with an index i , page 5
o_k	output value at the k^{th} node of the NN output layer, page 27
\mathbf{o}	observation represented as vector of features, page 4
$P(. ..)$	joint probability, page 30
$P(. ..)$	conditional probability, page 30
\hat{P}	test error estimate, page 9
\hat{P}	input training distribution for RBM, page 30
$Q(\mathbf{h}^k \mathbf{h}^{k-1})$	k^{th} RBM trained, page 30
r	epoch number of the BP algorithm, page 29
s_{ij}	element of S representing the number of observations that has been classified as class index j while having a true class index i , page 8
S	confusion matrix, page 8
$s[k]$	value of a sampled audio signal at temporal index k , page 13

t_k	target value at the k^{th} node of the NN output, page 27
T	total number of training observations, page 5
x	observation vector, page 30
x_i	input value at the i^{th} node of the NN input layer, page 27
y_j	output value the j^{th} node of the NN hidden layer, page 27
w	Hamming window, page 21
w	weights of an ANN, page 27
W_k	weight matrix of an RBM, page 30
AEC	acoustic event classification, page 1
ANN	artificial neural network, page 2
BP	backpropagation, 16
CD	contrastive divergence, page 19
DBN	deep belief network, page 2
DNN	deep neural network, page 2
GD	gradient descent, page 16
GMM	Gaussian mixture model, page 3
HMM	hidden Markov model, page 3
MFCC	mel-frequency cepstral coefficient, page 10
NN	neural network, page 2
RBF	radial basis function, page 16
RBM	restricted Boltzmann machine, page 19
RNN	recurrent neural network, page 16

1. INTRODUCTION

Multimedia is a huge aspect of everyday life and nowadays, one is constantly exposed to digital data in the form of image, audio, video etc. As the amount of data is constantly increasing, the need for retrieval of certain information and recognition of certain patterns out of it also increases. Multimedia information retrieval is concerned about execution of such tasks for multimedia signals. Audio information retrieval is a subfield of multimedia information retrieval in which audio signals such as speech, music, acoustic events etc. are of interest.

Audio information retrieval has numerous application areas, both in academia and industry, such as music information retrieval, speech recognition, speaker identification, acoustic event detection etc. These applications all involve various pattern recognition schemes to give a desired performance. Thus, pattern recognition principles that are tailored for audio data exhibit a high potential for research and should be put under further investigation.

1.1 Acoustic Pattern Recognition

One important area of audio information retrieval is acoustic pattern recognition which has been studied widely over the years by signal processing and machine learning scientists. It involves all kinds of pattern recognition tasks for audio signals, such as speech recognition [53], speaker identification [30], acoustic event classification (AEC) [62, 64, 65], musical genre classification [22] etc. Due to the variety of acoustic pattern recognition problems, different machine learning and signal processing schemes have been developed.

Acoustic pattern recognition applications can easily be introduced to the industry or everyday life. A mobile phone with a speech recognizer, a security system with speaker identification or a website that recommends songs by analyzing the user's taste of music are examples of already present applications and they all involve acoustic pattern recognition.

As acoustic signals can contain significant amount of information, their processing applications reach diverse fields in an increasing manner. However, the existing approaches to acoustic pattern recognition tasks still need improvement in two aspects, namely classification performance and usage of resources (time, memory etc.). The former one does not reach high accuracies when there are high number of classes and/or limited number of data. And when it comes to the latter one, the algorithms still need to be improved in many senses to be more efficient. Thus, there is an obvious need to conduct further research on the topic.

1.2 Neural Networks

Neural networks (NN) which were proposed to mimic the human brain structure, are nonlinear mathematical models used for function approximation (regression) and classification for numerous applications. They are also known as artificial neural networks (ANNs). NNs are composed of several layers each containing several neural units. They are strong classifiers due to their expression power to analyze multidimensional, nonlinear data. They are quite useful when the system is complicated and when it is difficult to express it in compact mathematical formulas. In addition, once trained, NNs have fast and reliable prediction properties.

Neural networks have shown to be noteworthy for several machine learning tasks such as stock market prediction [5, 72], optical character recognition [2], handwriting recognition [18], image compression [4, 28] etc. They are also used in acoustic pattern recognition tasks such as phoneme recognition [42], speech recognition [70], audio feature extraction [20] etc. With the help of recent developments in training algorithms and advancements in the hardware technologies as well as parallel computing (graphic processing units), the once-burdensome NN training methods are becoming popular again; this time unlikely to fade away.

1.3 Deep Architectures

As the number of layers in a neural network increases the network is said to be deeper. In general, NNs are trained in a supervised manner so that the network learns the system properties from examples which are simply the labeled data. Even though the evaluation (classification or regression) of unlabeled test data is fast, training a NN is not always a trivial task. NN training involves certain complications and the difficulty of training deep networks is one of them. The algorithm (backpropagation algorithm) used to train shallow NNs fails to learn the training data properties for deep neural networks (DNNs) if used as it is. However, an additional unsupervised pre-training stage has been proposed to overcome this problem [44] and shown to be successful. NNs that are trained in this manner are called Deep Belief Networks (DBNs). Discovery of means for training deep networks is considered a breakthrough in machine learning as they excel other approaches with a clear margin in performance.

DBNs have been recently used in several applications such as image classification [8, 9, 11], natural language processing [39], feature learning [19], dimensionality reduction [47] etc. and gave promising results. The complexity of tasks increase everyday and deeper networks can be beneficial to represent certain relations between inputs and outputs in these tasks. As recent scientific developments revealed efficient methods for training deeper networks, it would be wise to apply these findings to several fields such as acoustic event classification.

1.4 Objectives of the Thesis

The objectives of this thesis include studying artificial neural networks along with deep belief networks, understanding of working principles (effect of network parameters on classification performance, optimization etc.) of these concepts, and applying them to an acoustic event classification problem in which audio files of everyday sounds are automatically categorized into certain labels.

In addition, the comparison of the neural network classifier performance with that of conventional classifiers such as Hidden Markov Models (HMM) used with Gaussian Mixture Models (GMM) is part of the objectives.

1.5 Results of the Thesis

The primary result of this thesis work is a software implementation that includes neural and deep belief network algorithms for acoustic event classification purposes. The main result is that DBN performs slightly better than the standard NN for the given problem and the performances of both highly depend on several network and implementation parameters. The effect of these parameters on classification performance is also analyzed. Discussions and conclusions are made regarding the results.

1.6 Structure of the Thesis

The thesis is organized as follows. Chapter 2 describes the literature review on pattern recognition, acoustic event classification, neural networks and deep belief networks. Chapter 3 presents the used methodology including preprocessing, feature extraction, data division and network training algorithm descriptions. Chapter 4 reveals the evaluation details and results of several simulations. These consists of description of the used data and the classification performance results for neural and deep belief networks as well as effect of certain implementation parameters on the network performances. Finally, discussions on the results and suggestions for future research areas are pointed out in Chapter 5.

2. THEORETICAL BACKGROUND

This chapter starts with literature review on pattern recognition concepts including different learning paradigms, general structure and some characteristic properties. Then, acoustic event classification, common features used in the field and a short review of methods used in similar works will be discussed.

Further on, a brief description of a NN, types of NNs and significant aspects of them will be presented. Finally, the chapter is closed by a literature review on deep belief networks.

2.1 Pattern Classification

Pattern recognition is known as the act of processing raw data and taking an action based on the category of the pattern [29]. It is, simply, retrieving information relevant to application from the data and executing an action accordingly. Pattern classification is a subfield of pattern recognition, in which the input data is categorized into a given set of labels. It has numerous application areas varying from speech recognition to stock market prediction.

In a pattern classification system, each observation, \mathbf{o} , is represented as a *feature vector* of d dimensions, i.e., $\mathbf{o} = (f_1, f_2, \dots, f_d)$ where f represents a feature. Apparently, feature selection is a crucial part of a pattern classification system as it is domain dependent. Certain set of features for one application will probably not be useful for another. Feature extraction problem for acoustic event classification will be discussed in detail in Chapter 2.

2.1.1 Learning Paradigms

There are two main learning paradigms in pattern classification, namely, *supervised learning* and *unsupervised learning*. In unsupervised learning, the label which is known as *class*, \mathcal{C} , of any data is not available to the system. The system tries to learn the data properties and find similarities between observations which are represented as feature vectors.

Unsupervised learning can be used for diverse applications. Clustering is one of them; in which similar observations represented by feature vectors are grouped together. Examples include k-means clustering and mixture models. The former one is frequently used in computer vision [38] where the latter one can be used for speech recognition purposes [53] for instance.

For one to achieve better classification performance, the significant features that hold the most relevant information should be identified. As one can easily come up with too many features for almost any classification problem, a need for proper feature selection arises. Certain dimensionality reduction techniques overcome the problem by removing less relevant features from the data and thus; reducing the dimensions of it [37]. It does not only establish a better and more compact representation of the observations; but also avoids the problem of data becoming sparser as the volume increases with a power law. This phenomenon is known as *curse of dimensionality*. Dimensionality reduction methods such as principal component analysis, singular value decomposition, nonnegative matrix factorization have unsupervised learning principals.

There are a few reasons for usage of unsupervised learning principles. First of all, annotation and labeling of data is a burdensome process which is eliminated by unsupervised learning. Thinking of a speech recognition system, it is quite time-consuming to label each phoneme uttered by a speaker. Secondly, patterns to be classified may be time dependent. This type of time-varying cases cause serious difficulties for supervised systems. Lastly, one may need to extract an overall knowledge of the data properties before applying supervised learning. For instance, basic clustering algorithms such as k-means can be applied to find better initialization of certain supervised algorithms. Unsupervised learning has its own drawbacks too; difficulty for determining the number of classes, ambiguity in selection of distance metrics, poor performance for small datasets, to name a few.

Unlike unsupervised learning, in supervised learning the system is given a set of annotated (labeled) examples, i.e., the *training data*. Each training data is a vector of features representing an observation and the label information is available to the system. The aim is to categorize each observation, o_i , into a class, c_j , from a given set of classes where $i = 1, 2, \dots, T$ and $j = 1, 2, \dots, c$. Here, T is the total number of training observations and c is the total number of distinct classes. So, essentially, the system learns the properties of the data belonging to a certain class from examples.

One can list many examples for supervised learning algorithms and their applications. For instance, a k-nearest neighbor algorithm can be used for optical character recognition. Or a decision tree can be trained for data mining purposes. ANNs employ backpropagation algorithm which is also executed in a supervised manner. Further discussions on NN training can be found at the end of this chapter.

In general, these two learning paradigms are not the alternatives of each other. Instead, they are useful for distinct machine learning tasks. For example, certain problems are too complex to be solved without any supervision. Therefore, if annotated data is already available or one can afford a manual process of labeling, supervised learning can be utilized. There is also a third learning paradigm called *semi-supervised learning* in which the data to be used consists of both labeled observations and unlabeled ones.

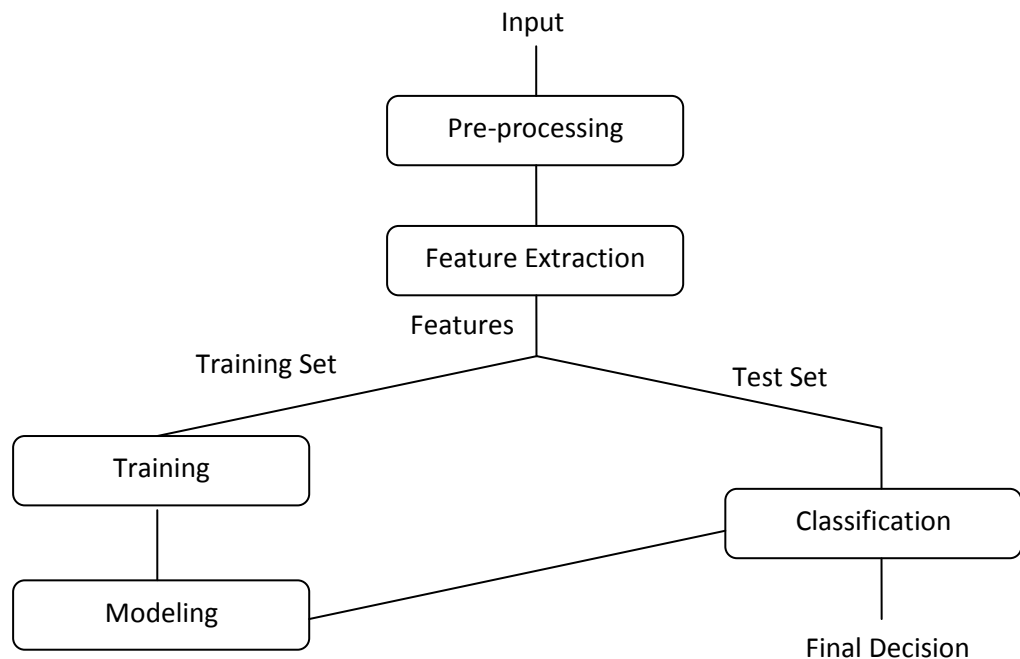


Figure 2.1. Block diagram for a typical supervised classification system.

2.1.2 Structure of a Pattern Classification System

A typical supervised pattern classification system whose schematic is given in Figure 2.1, is composed of the following blocks:

(i) *Preprocessing*: Input data is usually preprocessed before being fed into the next phase, i.e., feature extraction. Preprocessing techniques are signal processing operations such as filtering, normalization, transformation, trimming, alignment, windowing, offset correction, smoothing etc. and depend on the application. For instance, brightness and color intensity normalization for a face-recognition system or end-point detection for a speech recognizer are commonly used preprocessing techniques for corresponding systems.

(ii) *Feature Extraction*: Features are higher level representations compared to raw data representations, for example, corners instead of pixels, frequencies instead of raw temporal samples. After preprocessing, important attributes of the data should be selected in such a way that, those would contain enough information to properly represent the similarities between the inter-class observations and variations between the intra-class observations. Obviously, feature extraction is a highly problem-dependent phase.

(iii) *Training*: As a supervised system needs to learn the properties of the problem, it requires analysis of examples. Training phase correspond to the process of learning

from labeled data, i.e., training data. It can also be considered as detection of decision boundaries which distinguish different classes in the feature space. For the unsupervised case, there is no learning from labeled data but the decision boundary detection phase can be thought together with the classification phase.

(iv) *Modeling*: There are two types of modeling paradigms in pattern classification; one being *generative model* and the other *discriminative model*. Assuming an input x , represented by feature vectors and an output y which is simply the class information, the former one tries to learn the joint probability distribution of the input and the output, i.e., $p(x, y)$. So a generative algorithm models how the data is actually generated. The motivation for classification is to find an answer to the question, “Which class is more likely to generate this specific data?”. Thus, for classification, $p(x, y)$ is turned into $p(y|x)$ with the help of Bayes’ rule. The discriminative model, on the other hand, directly learns the conditional probability distribution $p(y|x)$. It can be interpreted as modeling the decision boundaries between the classes. Some examples of generative models are hidden Markov models (HMMs), Gaussian mixture models (GMMs) and naive Bayes classifiers. ANNs and support vector machines are examples of discriminative models.

(v) *Classification*: After modeling, classification has to be performed on the test data, i.e., the data which has not been available to the training phase. The test data represents the observations unseen to the system and the system’s performance of generalization is based on the evaluation of the classification phase.

2.1.3 Methods of Evaluation

Estimation of the performance of a pattern classifier is essential, as one wants to check how good a system generalizes for possible unseen data. It is a need to compare performances of different classifiers as well. There are three main evaluation methods for performance, namely, *resubstitution method*, *hold-out method* and *leave-one-out method*.

Before explaining the three evaluation methods, the concepts of *training error* and *test error* should be clarified. Training error and test error are the evaluation metrics (mean square error with respect to a desired value, distance to the decision boundary, percentage of misclassifications etc.) of the pattern classification system when the training data and test data are given as input to it, respectively. Training error is a measure of how well a system has learned the training data. However, as a system is judged according to its ability to generalize over an unseen data, test error is the significant one for evaluating a system. Due to its nature, error for the training data is less than that of the test data.

One has to be aware of that, low training error does not always imply low test error. For instance, the training error for a nearest neighbor classifier is zero, which clearly does not mean a test error of zero. For many pattern recognition systems, it is possible to encounter the problem of high test error while having a small training error. This

unwelcome phenomenon is known as *overfitting* or *overlearning*. It simply means that the system learns the properties of the training data too much and fails to generalize.

Assume a dataset \mathcal{D} with c different classes where \mathcal{D}_j corresponding to a subset including all observations belonging to the class \mathbb{C}_j and n_j corresponding to the total number of observations belonging to the class \mathbb{C}_j , that is:

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \dots \cup \mathcal{D}_c \quad (2.1)$$

Resubstitution method simply uses the training data as the test data, thus comes up with a conclusion by looking at the training error. Due to the reason explained above, it is most likely to be an overoptimistic estimate of the classifier performance.

A better evaluation method would be hold-out method where the dataset is divided into training and test sets, $\mathcal{D}_{training}$ and \mathcal{D}_{test} , respectively. Apparently, a division as $\mathcal{D}_{test} = \mathcal{D}_j$ for any j , is not desired. The division can be performed by *random sampling*, in which the dataset is simply divided randomly over all observations. If the number of observations belonging to each class differs a lot from each other, *stratified sampling* can also be used. In stratified sampling, observations belonging to each class are divided by preserving the division ratio of training over test. Hold-out method can be used for large datasets, considering the idea that the presence of sufficiently many training data will be enough to train the classifier even after partitioning.

In leave-one-out method, a single randomly chosen observation from the dataset \mathcal{D} is left out to be the test set and the classifier is trained with the rest of the data. Then the classifier is tested with the left-out observation. This process is repeated by sweeping all of the observations and leaving out one of them for testing one by one. Then the performance (test error) estimate, $\hat{\mathcal{P}}$, is

$$\hat{\mathcal{P}} = \frac{m}{n} \quad (2.2)$$

where m is the total number of misclassified observations and $n = \sum_{j=1}^c n_j$ is the total number of observations in the dataset. Note that leave-one-out method is computationally expensive as the training has to be done for n times. A more general approach is known as *cross-validation*, in which the dataset is randomly divided into k subsets of equal sizes and each subset is used as the test set once, while the rest $k - 1$ subsets are altogether used as the training set. Then the average of classification errors for each fold is calculated for an estimate of test error. It is straightforward to see that leave-one-out method is a special case of cross-validation in which $k = n$.

For many applications, the information of the classification rate for each class separately may be valuable. By knowing this, one may lead to conclusions about whether the observations belonging to a certain class are easy to classify or not. A frequently used visualization tool for this purpose is the *confusion matrix* (CM), S . It is a $c \times c$ matrix in which each row represents instances (observations) of an actual class, while each column represents instances of the predicted class. Thus, the element s_{ij} in the matrix represents the number of observations that has been classified as class index j while having a true class index i .

$$S = \begin{matrix} & \begin{matrix} \text{predicted class } j \\ \hline S_{11} & S_{12} & \cdots & S_{1c} \\ S_{21} & S_{22} & \cdots & S_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ S_{c1} & S_{c2} & \cdots & S_{cc} \end{matrix} \\ \begin{matrix} \hline \\ \\ \\ \end{matrix} & \text{true class } i \end{matrix}$$

Confusion matrix can be formed using the same methods described above. The performance prediction from the confusion matrix can be calculated easily with the following formula:

$$\hat{\mathcal{P}} = \frac{\sum_{i=1}^c \sum_{j=1}^c S_{ij} - \sum_{i=1}^c S_{ii}}{\sum_{i=1}^c \sum_{j=1}^c S_{ij}} \quad (2.3)$$

where $\hat{\mathcal{P}}$ is simply the test error.

2.1.4 Sources of Error

When designing a system, one has to be aware of the possible sources of error. This awareness enables one to both keep these errors under a certain limit that can be tolerated for the application, and to avoid the unwanted consequences of minimizing those errors as much as possible.

For a pattern recognition system there are three different sources of error. First one is the *Bayes error* that comes from the pattern recognition problem itself. This type of error may only be reduced by changing the problem, for example the features and the overlap of classes in the feature space. The second source of error is the *model error*. Model error comes from the inappropriate assumptions made on the class conditional densities for the parametric classifiers such as support vector machines. For the nonparametric case, it comes from the poor choice of certain parameters for example, k for a k -nearest neighbor classifier. Lastly, there is the *estimation error* which is inevitable for practical cases as it is due to the finite number of training observations. Estimation error can simply be reduced by increasing the number of training data.

Even though one desires to minimize the abovementioned errors, it is usually not a simple task to do so. In many cases, an attempt to decrease one of these errors results in certain other undesirable consequences such as increase in model complexity, increase of computations etc. For example, adding more features may decrease the Bayes error but will result in an increase of dimension which leads to an increased computational burden. Similarly, adding more data will surely effect the computation time for an algorithm. A designed pattern recognition system has to establish a proper balance between these trade-offs for high performance and low cost.

2.2 Acoustic Event Classification

As scientists want to learn more and more about human behavior, many aspects of human daily life has been under inspection. The investigation of sounds around humans' environment, which are generated by nature, by objects handled by humans or by humans themselves, is one of the research topics. Classification and detection of these sounds, namely *acoustic events*, has been studied over the years as it would be fruitful to describe human activity or improve other pattern recognition areas such as speech recognition.

Research on acoustic event classification has been conducted in different ways. One is classification of acoustic events into event classes for a specific context; meaning recognition of events for a given environment. Such environments can be meeting rooms, office, sports games, parties, work sites, hospitals, restaurants, parks etc. In [16] sounds of drill during spine surgery has been classified to give feedback to the doctors on density of the bones. In [51] detection and classification of sounds from a bathroom environment has been established. Human activity detection and classification in public places have been under investigation in [57]. In [49] a system for bird species' sound recognition was proposed.

Another case of AEC research is classification of acoustic events into contextual classes. In [34] authors have clustered events into 16 different environment classes (campus, library, street etc.). A classification system for a similar everyday audio context, such as nature, market, road, have been proposed in [35]. For hearing-aid purposes, research has been conducted on classification of events into classes like speech in traffic or speech in quiet [64].

Apart from these, classification of sounds which are not strictly related to an environment has also been examined. Alarm sound detection and classification was proposed in [32]. For autonomous surveillance systems, non-speech environment sound events have been classified in [23]. A wide variety of sounds such as motorcycle, sneezing, dishes etc. has been classified in [36].

Throughout these works, varying classification rates have been achieved depending on the complexity of the problem (number of different classes, available number of data, quality of the data, distribution of the data etc.) The features used to represent the audio data and the classifiers used for the classification task also differ from work to work. Those two aspects will be discussed in this chapter as well.

2.2.1 Features Used in AEC

For acoustic pattern recognition, one can extract numerous number of features and the number of possible features do not really decrease when it comes to its subfield, i.e., acoustic event classification. As feature extraction is extremely crucial for a system, many features have been tried out for AEC purposes.

Automatic speech recognition (ASR) features such as mel-frequency cepstral coefficients (MFCCs) have been widely used as well as perceptual features. Some of the main

features used in AEC are explained below. Note that preprocessing techniques such as preemphasis, frame blocking and windowing are quite commonly encountered before feature extraction phase. Most of the following features are assumed to be applied on a particular frame of the signal (frame-blocking is explained in Chapter 3) instead of on the whole signal.

Mel-frequency Cepstral Coefficients

MFCCs have been proposed first as a set of features for ASR [25]. These coefficients are derived from the mel-frequency cepstrum which is a representation of short time power spectrum of a sound. As the vocal tract shapes the envelope of this spectrum, MFCCs tend to represent the filtering of the sounds by vocal tract. A mel-frequency cepstrum differs from a regular one as it is linearly scaled in the mel scale to mimic the human auditory system better, whose frequencies are defined as:

$$f_{mel} = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad (2.4)$$

where f_{mel} is the mel frequency mapping of a standard frequency scale value f .

MFCCs have been widely used as acoustic features [53, 56] and are shown to be effective for representing audio data. The k^{th} MFCC, c_k , is defined as

$$c_k = \sum_{i=1}^N E_i \cos \left[k \left(i - \frac{1}{2} \right) \frac{\pi}{N} \right] \quad k = 1, 2, \dots, L \quad (2.5)$$

where E_i is the log energy within each mel band, N is the number of mel bands filters and L is the number of mel-scale cepstral coefficients.

The block diagram of a MFCC extractor can be seen in Figure 2.2. The input signal is assumed to be preprocessed, i.e., scaled, frame-blocked and windowed. The DFT is an abbreviation for discrete Fourier transform. The output of this block represents the power spectrum of the signal which is then point-wise multiplied with a certain number of triangular mel-scale filter responses. This multiplication in frequency domain corresponds to filtering in time domain. Then, the logarithm of the energies for each mel-scale filter is computed to compress the dynamic range. Lastly, discrete cosine transform (DCT) is applied to decorrelate the coefficients from each other.

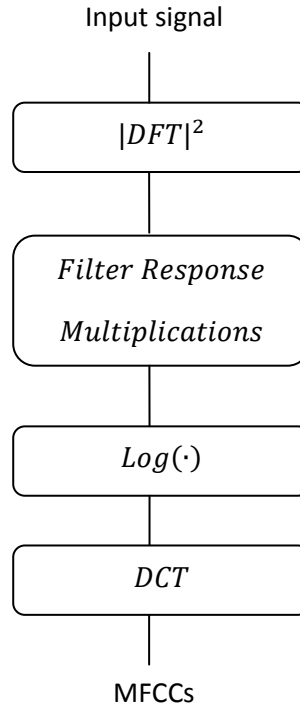


Figure 2.2. Extraction process of MFCCs from an input signal

Mel Energies

Mel energies are another set of commonly used spectral features. They are composed of coefficients representing the energy of the signal in each mel filterbank. Figure 2.3 shows the process of extracting mel energy features.

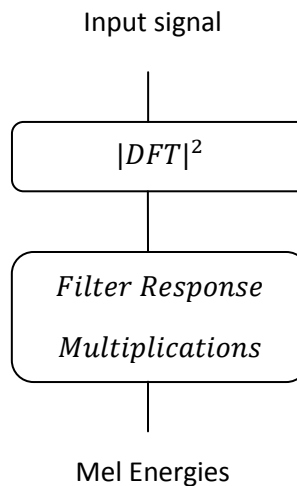


Figure 2.3. Extraction process of mel energies from an input signal

There are numerous other features that can be used in AEC such as zero-crossing rate [41, 62, 65], short-time energy [41, 62], spectral centroid [52] etc. The properties of these features will not be discussed in detail as only MFCCs and mel energies were used

in the implementation of this work. A few of these other features are shortly presented below.

Zero-Crossing Rate

Zero-crossing rate (ZCR) is simply the rate of number of zero-crossings of a signal, s , within a frame and can be calculated as

$$ZCR = \frac{1}{2N} \sum_{k=1}^N |sgn(s[k] - s[k - 1])| \quad (2.6)$$

where N is the length of the frame under investigation and

$$sgn(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \quad (2.7)$$

Short-time Energy

Short-time energy (STE) is the total signal energy in a frame:

$$STE = \sum_{k=0}^{N-1} s(k)^2 \quad (2.8)$$

Spectral Centroid

Spectral centroid (SC) is a measure of spectral brightness and can be calculated as

$$SC = \frac{\sum_{\forall i} f(i)A(i)}{\sum_{\forall i} A(i)} \quad (2.9)$$

where $f(i)$ and $A(i)$ are the frequency and amplitude values of the i^{th} discrete Fourier transform bin.

2.2.2 Classifiers Used in AEC

There are several classifiers used in acoustic event classification. One of the first works in AEC [17] have used minimum distance classifier according to a chosen metric to find the distance between two observations in the feature space. A few others coming after that have establishes the k-nearest neighbor classifier [57, 58, 59] for certain acoustic events. ASR algorithms such as GMMs [1, 3, 14, 15, 50, 57, 68, 69] and HMMs [16, 27, 33, 54, 60, 61, 64] are the most commonly used methods. Some have also used ANNs [16, 31, 32]. Other methods such as vector quantization [24], decision trees [48] and support vector machines [16, 33, 41, 63] have also been tried. For audio-visual data, a

k-means clustering algorithm was used in [26]. For a compact visualization, a list of different classifiers used in various works can be seen in Table 2.1.

Table 2.1. Various works on acoustic pattern recognition and corresponding classifier used in their pattern recognition systems

Classifier	Works
Minimum Distance	[17]
k-Nearest Neighbor	[57, 58, 59]
Gaussian Mixture Model	[1, 3, 14, 15, 50, 57, 68, 69]
Hidden Markov Model	[16, 27, 33, 54, 60, 61, 64]
Artificial Neural Networks	[16, 31, 32]
Vector Quantization	[24]
Decision Trees	[48]
k-Means Clustering	[26]
Support Vector Machines	[16, 33, 41, 63]

2.3 Neural Networks

The idea of neural network comes from the biological sciences. Scientists wanted to build up a mathematical model that resembles the structure of a brain, which in real life has extremely powerful recognition capabilities. The human brain consists of an estimated number of 10 billion neurons (nerve cells) and 60 trillion connections (known as *synapses*) between them [43]. This network processes all kinds of information in our body and gives decisions accordingly.

2.3.1 The Single Neuron

The most elementary unit of a neural system is a *neuron* in both biological and artificial networks. Synapses correspond to the connections between neurons and are responsible for transmitting information (stimulus). As a neuron can be connected to many other neurons, several stimuli can cumulate in a neuron. For an ANN, one can think of the stimuli as the incoming signal x_i and the synapses as the connections w_i . In practice, w_i are represented as weights that scale the incoming inputs according to their importance. These weighted inputs accumulate inside the neuron and some function of the sum is given as an output, y . This function, $\varphi(\cdot)$, is called the *activation function*. In general there is also a bias (threshold) term, τ , for each neuron. An example schematic of a simple NN structure can be seen in Figure 2.4.

In mathematical terms, the output is given by:

$$y = \varphi \left(\sum_{i=1}^n x_i w_i - \tau \right). \quad (2.10)$$

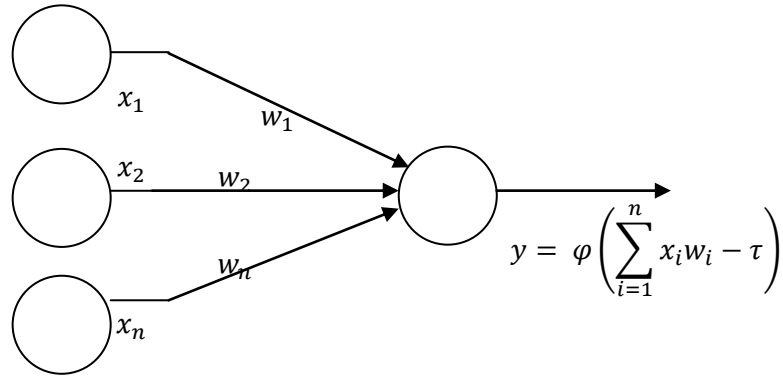


Figure 2.4. A simple NN structure

Types of Activation Functions

Activation functions for NNs are usually three kinds:

(i) *the threshold function*

$$\varphi(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases} \quad (2.11)$$

(ii) *the piecewise linear function*

$$\varphi(x) = \begin{cases} 1 & x \geq \frac{1}{2} \\ x & -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 0 & x \leq -\frac{1}{2} \end{cases} \quad (2.12)$$

(iii) *the sigmoid function* which include the functions that has an S shape. The most frequently used sigmoid function is the *logistic function* which can be described as:

$$\varphi(x) = \frac{1}{1 + e^{-\alpha x}} \quad (2.13)$$

where α determines the slope of its curve. The plots of these three functions can be seen in Figure 2.5. Other similar types of sigmoid functions are arctangent and hyperbolic tangent.

The sigmoid function is frequently used as an activation function in NNs due to two reasons. First, it is a differentiable function. Secondly, its derivative has a compact form, i.e.

$$\frac{\partial \varphi(x)}{\partial x} = \frac{\alpha e^{-\alpha x}}{(1 + e^{-\alpha x})^2} = \alpha \varphi(x)(1 - \varphi(x)) \quad (2.14)$$

which enables easier derivative computations. As ANNs are trained with the backpropagation (BP) algorithm which involves derivative computations of activation functions due to gradient descent (GD) algorithm, sigmoid activation functions are favored. Details of the backpropagation training will be given in Chapter 3.

Obviously, the output of a neuron can be both binary (having two possible values) or continuous depending on the activation function. The range for activation functions are usually either between 0 and 1 or between -1 and 1.

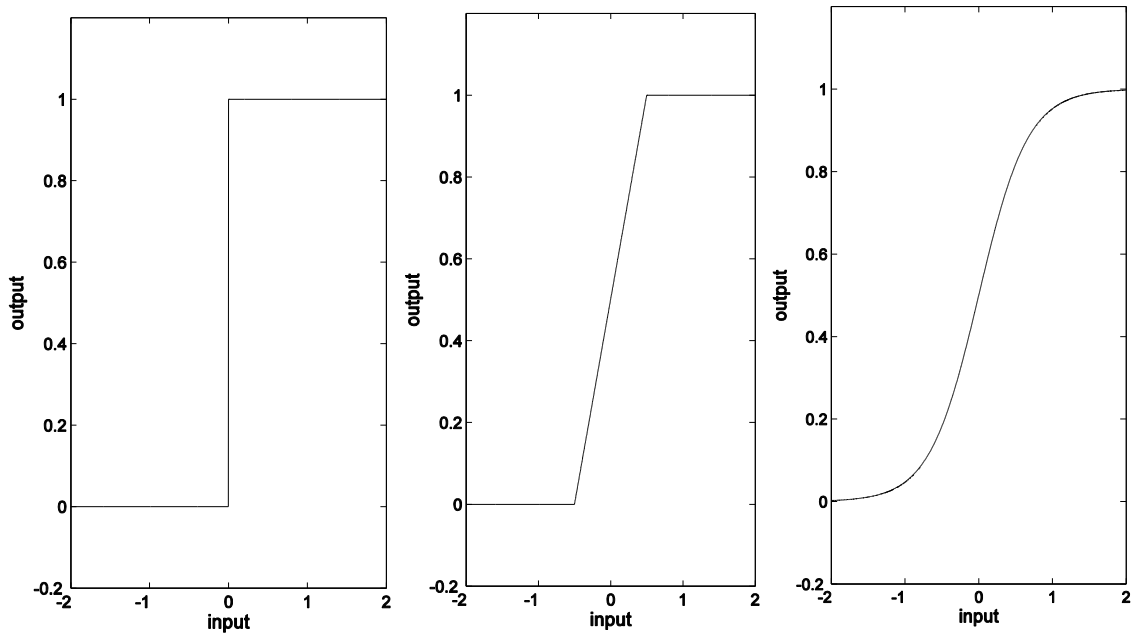


Figure 2.5. Plots of three different types of neural activation functions

2.3.2 Network Structures

The structure and topology of a NN is significant on its performance [43]. Categorization of NNs is rather ambiguous but one can assume that there are mainly four types of neural networks, i.e., *Feed-forward Neural Networks*, *Recurrent Neural Networks (RNNs)*, *Radial Basis Function (RBF) Networks* and *Modular Neural Networks*. *Kohonen Self-Organizing Networks* may also be included, however, those perform unsupervised learning and are different than the rest in that sense.

Feed-forward Neural Networks

Feed-forward neural networks can be considered as the simplest and the most typical NN type. A regular multi-layer feed-forward network consists of several layers each containing several units called neurons. The first and the last layers are called the *input layer* and the *output layer* respectively. The layers in between these two are called the *hidden layers*. The total number of layers and the number of units in each layer affects the expression power of a NN.

A NN is said to be *fully connected* if each neuron in a layer is connected to every other neuron in the following layer. The example in Figure 2.6 corresponds to this type of networks as there are no missing connections between neurons. Otherwise, the NN is said to be *partially connected*.

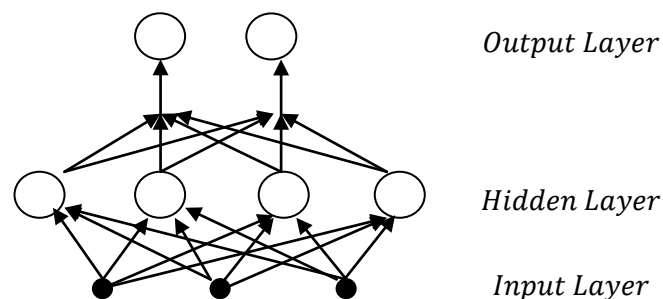


Figure 2.6. A typical feed-forward NN structure

Recurrent Neural Networks

Recurrent neural network is a type of NN which contains at least one feedback loop in its structure. Biological neural networks, e.g. brain, are RNNs. The ability to use internal memory for processing arbitrary input sequences makes them powerful on certain tasks such as handwriting recognition [13].

Radial Basis Function Networks

Radial basis function networks are ANNs which establishes radial basis functions as its activation functions in each unit. A radial basis function is such a function that its value depends only on the distance from the origin. The most common one is the Gaussian. RBF networks can be trained using the standard iterative algorithms. The application areas vary from time series prediction to function approximation.

Modular Neural Networks

Modular neural networks are networks that are composed of several neural nets which perform certain subtask of the original task. The solutions of each subtask are then combined to form the solution to the original problem.

2.3.3 Reasons for Using Neural Networks

A neural network derives its computational power from two aspects; first from its highly parallelized structure, second, from its ability to generalize [43]. NNs are used in numerous applications due to the following reasons:

(i) *Nonlinearity*: NNs are highly nonlinear classifiers not only because they have nonlinear activation units but also because of the layer-wise structure stacked one after another. This framework enables the NNs to learn the highly nonlinear input-output relationships of many classification and regression problems in a successful manner.

(ii) *Robustness*: A NN can be considered to be robust in a structural sense and it is rather intuitive to understand it. Taking a hardware implementation of a NN, e.g., VLSI, into account, one can safely claim that the NN will not totally crash down and stop functioning immediately if a single neuron or connection is damaged. Even though certain degradation of performance would be observed, the multi-layer, multi-unit framework would prevent a sudden failure.

(iii) *Ease of Use*: One can use NNs for solving a certain problem without going deep into the formal mathematical and statistical relations between inputs and outputs. In general, complex nonlinear relationships of variables can be learned implicitly. It is significant at this point to emphasize that this property can also be interpreted as a drawback. The black-box nature of the NN makes the understanding of effects of parameters on the performance (both computational and statistical) quite hard. Therefore, one may say that the ease of use property of NNs come hand in hand with the difficulty of building up intuitions for a problem.

(iv) *No Need of Assumptions*: Once the labeled data is obtained, it can be fed into the training algorithm without any statistical assumptions.

2.3.4 Training of ANNs

ANNs are trained in a supervised manner with the *backpropagation algorithm*, an abbreviation for *backward propagation of errors*. Even though the very first implementation of the algorithm did not aim NN training [71], the discovery of its benefit in the subject revived the NNs in science of machine learning [46]. There are several BP algorithms but the main aspect of all of them is the same.

As BP algorithm involves supervised learning, the principal idea behind it, is to adjust the network coefficients (weights) so that the output values for the training data are as close as possible to the desired output values. To establish that, after initializing the network weights to small random numbers, the error at the output layer, i.e., the discrepancy between the output value and the desired value, is calculated. Then, the network weights are updated after each iteration according to gradient descent rule to decrease the output error. The training continues until a certain criterion is satisfied. A detailed discussion of the BP algorithm is in Chapter 3.

2.4 Deep Belief Networks

BP algorithm performs effectively for shallow networks, i.e., those that have 1 or 2 hidden layers, but its performance declines when the number of layers increases. Numerous experiments show that the algorithm gets stuck in local optima easily and fails to generalize properly [46, 47] (with a possible exception of *convolutional neural networks*, which were found to be easier to train even for deeper architectures [40, 43, 67]). However, in general it is shown that, when NN weights are randomly initialized, DNNs perform worse than the shallow ones [8, 46]. The solution to this problem is encountered by deep belief networks.

2.4.1 Need for DBNs

It is hard to say that there exist a universal right number of layers for every recognition task but deep architectures might have theoretical advantages over the shallow ones when learning complex input-output relations. Furthermore, results suggest that a relation that can be represented by a deep architecture might need a very large architecture to be represented by a shallow one [6, Chapter 2]. Larger structures may require an exponential number of computational elements which will decrease the computational efficiency. In addition, if a concept requires abounding elements to be represented (weights to be tuned for example) by a model, the number of training examples needed to learn that concept may grow very large. Thus, research on training of deep architectures as well as understanding the effects of its parameters to generalization ability is crucial.

2.4.2 DBN Learning

As mentioned in the beginning of this chapter, serious difficulties are encountered while training a DNN with BP algorithm when its weights are randomly initialized. Yet, in 2006 it was discovered that an unsupervised pre-training, conducted layer by layer, to initialize the network weights results in much better performance [45]. DNNs which are pre-trained in such a greedy layer-wise unsupervised manner are called *deep belief networks*. Thus, DBNs are not any different than DNNs in terms of architecture or structure, but have a clever learning strategy tailored for several-layer training.

The training scheme for a deep belief network is based on *restricted Boltzmann machine* (RBM) generative model. An algorithm called *contrastive divergence* (CD) is applied to train a RBM before applying standard supervised training which serves as a fine-tuning process of the weights of a NN. CD algorithm trains the first layer in an unsupervised manner, producing an initial parameter values set for the first layer of a NN. Then, the output of the first layer is fed as an input to the next, again initializing the corresponding layer in an unsupervised way and so forth. The details of the algorithm are given in Chapter 3.

Several results underline the advantage of unsupervised pre-training on the DNN performance [7, 8, 10, 12, 21, 44]. Simply, the unsupervised pre-training prepares the NN weights for the initialization of supervised training as usual, so that the BP algorithm converges to a better solution.

3. METHODOLOGY

In this chapter, the implementation steps and the details of the used algorithms for the thesis work are described. These steps include preprocessing, feature extraction, division of data, training algorithms and the classifier.

3.1 Preprocessing

Digital audio data, if not synthesized, is collected by recording of sounds. As conditions may differ for every recording, the peak amplitude of audio signals will most probably differ. Furthermore, it is hard to ensure that every audio data borrowed from a database has not been processed digitally. Therefore, it is a wise practice to normalize the data in terms of amplitude before feeding into our pattern recognition system for better generalization.

For the preprocessing phase, firstly peak amplitude normalization has been conducted:

$$s_{nor}[k] = \frac{1}{\max_{i \in n} s_{raw}[i]} \cdot s_{raw}[k] \quad (3.1)$$

where s_{nor} is the normalized signal (output), s_{raw} is the raw signal (input), $n = 1, 2, \dots, n_{max}$ and n_{max} is the length of the audio sequence.

It is a common practice in audio signal processing to analyze the audio data by dividing it into smaller frames instead of as a whole. By using small frame lengths, it is safe to assume that the spectral characteristics of the signal in that frame are stationary. This process is called frame-blocking. Furthermore, these frames are usually smoothed by multiplying certain window functions with them. Frame-blocking and windowing was conducted to each audio data with a Hamming window of 50 ms with 50% overlap. The Hamming window of length N is defined as

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \quad (3.2)$$

where $n = 1, 2, \dots, N$.

With the help of preprocessing, the data is made more robust for feature extraction. This will lead in a better design of a pattern recognition system with improved generalization ability.

3.2 Feature Extraction

After normalization, frame-blocking and windowing of acoustic event audio files, certain features should be extracted from them. Even though each file represents an observation in the sample space, after frame-blocking, each frame will be considered to be an observation. One can imagine this situation as each observation in the original sample space divides into many smaller observations which are ready to be mapped to the feature space by feature extraction.

For the work, 13 static MFCCs are extracted from each frame. The first coefficient mainly represents the signal energy in the frame and it is discarded. Thus, in total 12 MFCCs are used as features. Details of the feature extraction process is presented Table 3.1.

One expects certain correlations between two adjacent frames of audio data due to the dynamic properties of sounds. Neighboring frames are assumed to contain information about the current frame. Current frame features can be thought as static features while features extracted from the adjacent frames are dynamic. This idea leads one to examine the adjacent frames as well, for a certain audio data representation task. As the neighboring frames of a certain frame may contain information about the corresponding frame, MFCCs from some number of adjacent frames are also used as features. The *number of adjacent frames* used as features, can be considered as a parameter for an implementation even though this number is kept the same throughout the work.

Table 3.1. The parameters and their values for MFCC extraction

Parameter	Value
Window length	50 ms
Window overlap	50%
Number of MFCCs	12
Number of mel bands	40

3.3 Division of Training, Validation and Test Data

As mentioned in the Chapter 2.1.1, when a supervised pattern recognition system is in concern, the data is divided into two parts as training data and test data. The training set is used to teach the properties of the data to the system by examples and the test set is used to evaluate the trained model.

The NN system also has the same division. However, due to the nature of NNs, there are quite many parameters to be decided. Note that these parameters are referring to the architecture of the implementation, not the NN weights. These include structural parameters such as number of hidden layers, number of hidden units in each hidden layer, the activation function, bias terms as well as training algorithm parameters such as number of epochs, learning rate etc. As there are no concrete mathematical equations

describing the effect of these parameters on the classification performance, one needs a practical way to tune them. This is where the *validation data* comes into scene. The validation data is the data used to tune the parameters of the NN for better performance. The validation data can not be considered as test data as it is not used to evaluate the performance of the system. It can not be considered as training data either as it is not available to the system during NN training.

Apart from network parameter estimation, the validation set plays a significant role in preventing overfitting. Due to their high expression power, NNs can easily overfit a pattern recognition problem. Thus, overfitting is a serious concern when dealing with ANN training. The validation data can also be used for early stopping of a NN training by backpropagation algorithm which is described in the following chapter. The BP algorithm requires a stopping criterion/criteria to cease the iterations and the validation data gives valuable information to decide these criteria. So the validation set is used to verify that a decrease in training error does not result in a decrease in the classification accuracy of a set which was not used to modify the network weights. In a typical learning curve, the validation error is expected to stop decreasing around the same epoch number that the test error to stop decreasing. In summary, it simply helps one to decide the network architecture and prevent overfitting by acting as a fake test set.

An example of learning trajectories for a NN training can be seen in Figure 3.1. Note that, as expected the training error is lower than the other two and smoother as well due to large number of observations relative to the other two (mean square averaging smoothens the curve). Obviously, training error is constantly decreasing with more iterations as the system learns the properties of the training data more and more. After each epoch, it approaches to a limit and this behavior represents the situation of the training being stuck in local optima. On the other hand, after decreasing for a while the validation error and the test error start to increase. In that sense, the validation error has successfully mimicked the test error and provided a satisfactory estimate for the point to stop training to avoid overfitting. One should mention that even though the validation error had a very strong correlation with the test error for this particular simulation, this is not always guaranteed due to the uneven distribution of number and length of files per class for the database. The database is examined in detailed in the following chapter.

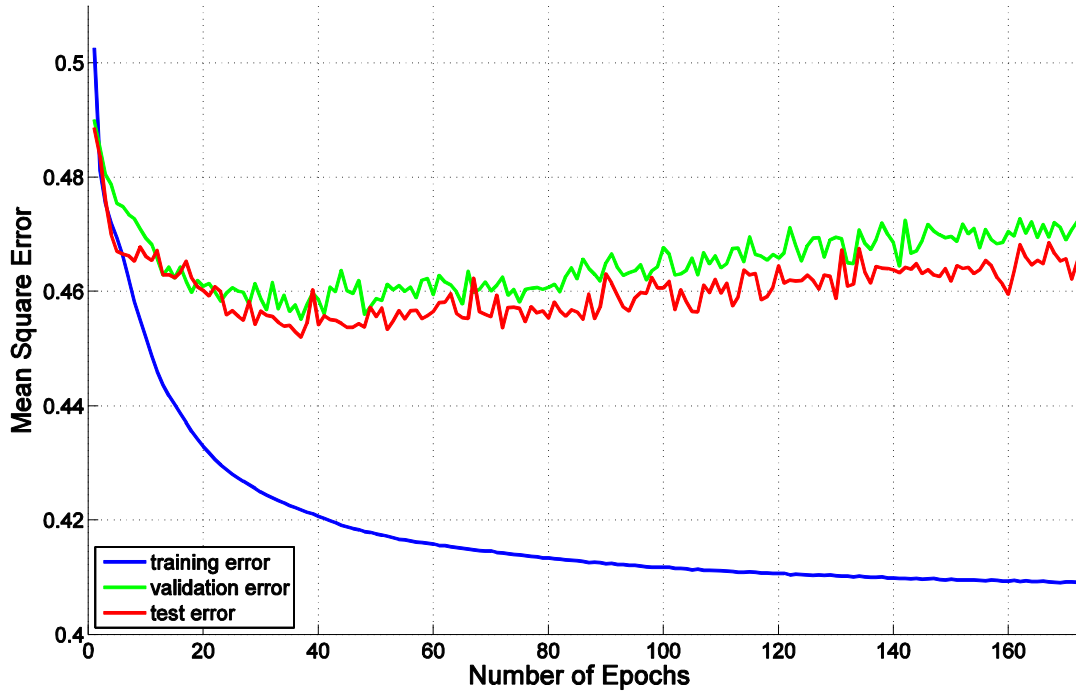


Figure 3.1. The learning curves including training, validation and test error trajectories during a NN training.

The division of training, validation and test data affects the performance of the system directly. Even though evaluating a system with a cross-validation with high number of folds gives a better estimation of the performance, due to computational complexity certain practical decisions have to be made. A 10-fold cross-validation has been established in our implementation in which the data is randomly divided into 10 equal sized subsets. This division is made on the actual audio files independent of the number of frames coming from that file (length of the file). 10% of the files (1 subset) is used as a validation set, 10% (1 subset) as a test set and the NN is trained with the remaining 80% of the files (8 subsets). Actually, the implementation enables that each file has been used at least once as a training, validation and test data as it can be seen from Figure 3.2.

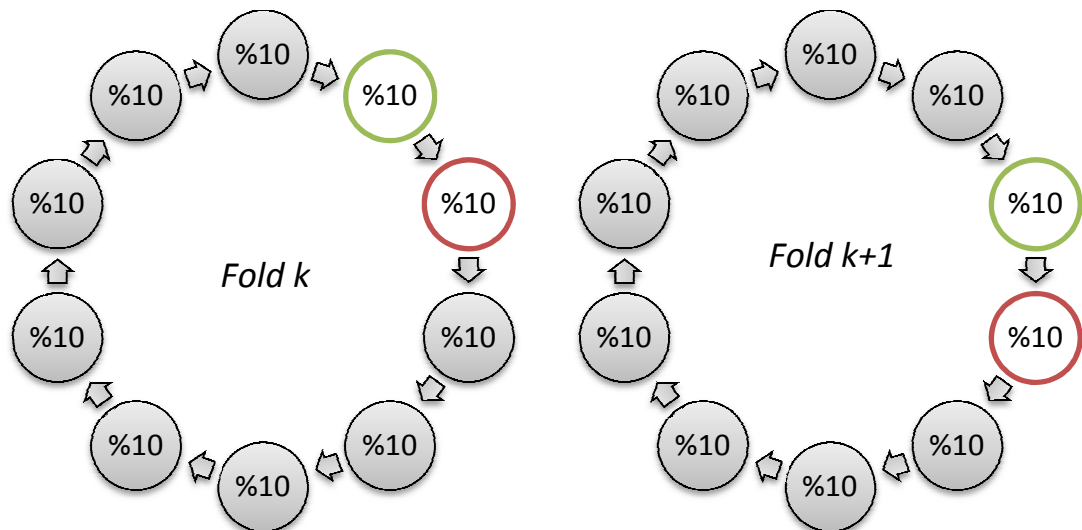


Figure 3.2. A visual schematic to demonstrate the division of training, validation and test data. Validation and test sets are represented by green and red circles respectively, while the rest corresponds to the training set.

3.4 Training Algorithms

The process of learning from examples is conducted with the help of NN training algorithms. Standard ANN training involves backpropagation algorithm.

3.4.1 Backpropagation Algorithm

As briefly mentioned in Chapter 2, the supervised training of an ANN is performed by backpropagation algorithm. The mathematical details of the algorithm are given in this chapter.

NNs have two modes of operation, namely *feed-forward mode* and *learning mode* [29]. In feedforward mode, network is fed with an input vector through its input (first) layer and an output is generated in the output (last) layer simply by a sweep of calculations through the network. These calculations are composed of multiplications of each node output by a certain weight and then computation of the activation function output for the sum of each incoming multiplications. In the learning mode, NN is again given a vector of features through its input layer but there is also a desired value (target pattern) at the output of the network.

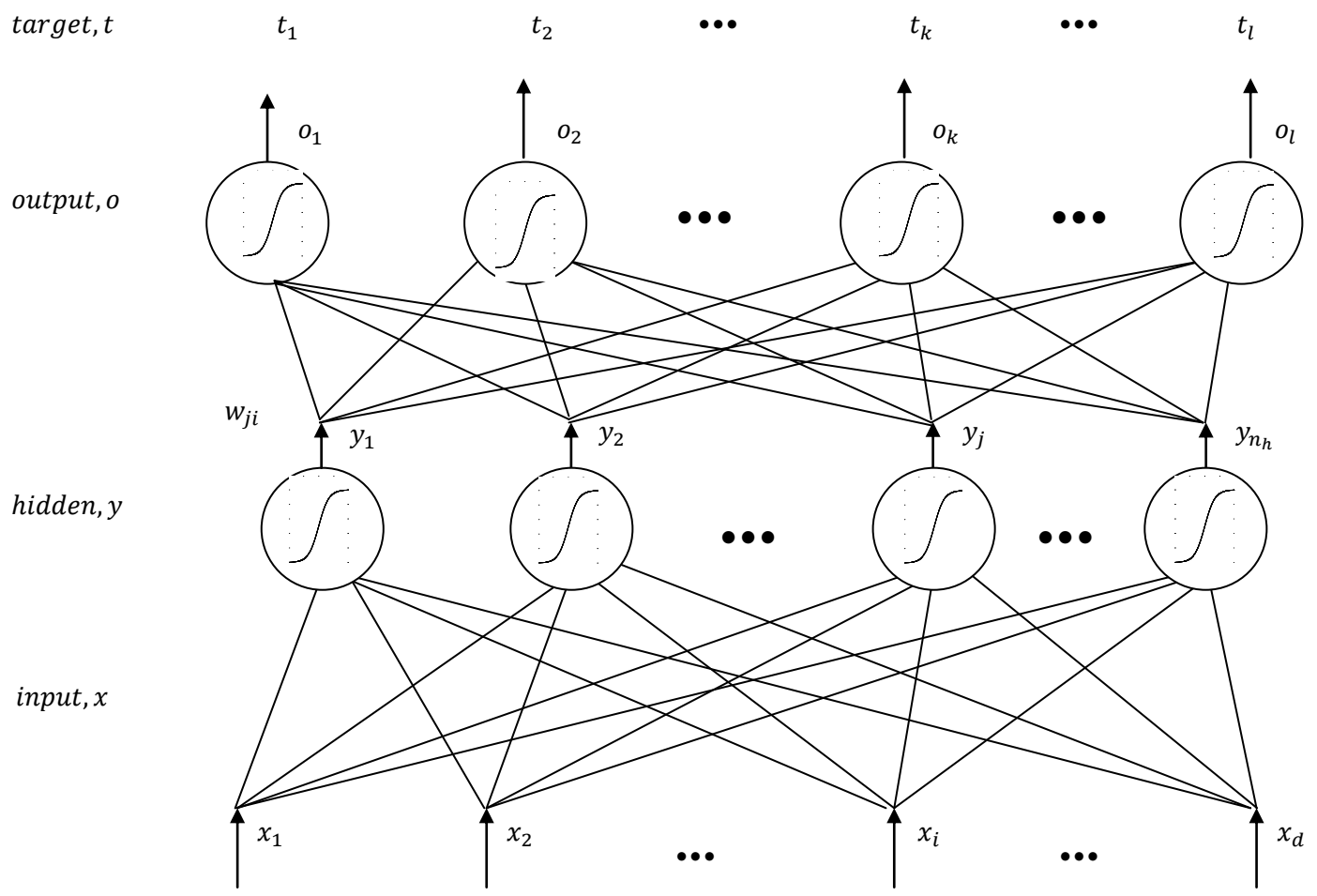


Figure 3.3. A 3-layer feed-forward NN schematic with input layer, hidden layer and the output layer

Assuming a 3-layer NN (input, hidden and output) with randomly initialized weights, the training error at the output, \mathcal{E} , is defined as:

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^l (t_k - o_k)^2 \quad (3.3)$$

where \mathbf{w} represents all of the weights in the network, l is the number of nodes at the output, t_k is the target and o_k is the output value at the k^{th} node of the output. A schematic for the network topology can be seen in Figure 3.3.

The backpropagation learning rule is based on gradient descent algorithm in which the weights are changed in the direction that would give less error:

$$\Delta \mathbf{w} = -\eta \frac{\partial \mathcal{E}(\mathbf{w})}{\partial \mathbf{w}} \quad (3.4)$$

or component-wise

$$\Delta w_{pr} = -\eta \frac{\partial \mathcal{E}}{\partial w_{pr}} \quad (3.5)$$

where η is the *learning rate* of the algorithm and w_{pr} simply represents the weight from the r^{th} unit of a layer to the p^{th} unit of the next layer. By *chain rule*:

$$\frac{\partial \mathcal{E}}{\partial w_{kj}} = \frac{\partial \mathcal{E}}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = \delta_k \frac{\partial net_k}{\partial w_{kj}} \quad (3.6)$$

where j is the index for the hidden layer units and

$$\delta_k = \frac{\partial \mathcal{E}}{\partial net_k} = \frac{\partial \mathcal{E}}{\partial o_k} \frac{\partial o_k}{\partial net_k} \quad (3.7)$$

$$net_k = \sum_{j=1}^{n_h} w_{kj} y_j \quad (3.8)$$

where δ_k is called the *sensitivity* for unit k and

$$y_j = \varphi(net_j) = \varphi\left(\sum_{i=1}^d x_i w_{ji}\right) \quad (3.9)$$

i being the index for the input layer nodes and x is the input. Note that

$$\frac{\partial \mathcal{E}}{\partial o_k} = -(t_k - o_k) \varphi'(net_k) \quad (3.10)$$

Therefore, the weight update for the hidden-to-output layer is as:

$$\Delta w_{kj} = \eta \delta_k y_j. \quad (3.11)$$

The learning rule for the input-to-hidden layer again uses the chain rule as such:

$$\frac{\partial \mathcal{E}}{\partial w_{ji}} = \frac{\partial \mathcal{E}}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}. \quad (3.12)$$

and if one examines the partial derivative of the error with respect to a hidden unit output:

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial y_j} &= \frac{\partial}{\partial y_j} \left[\frac{1}{2} \sum_{k=1}^l (t_k - o_k)^2 \right] = - \sum_{k=1}^l (t_k - o_k) \frac{\partial o_k}{\partial y_j} \\ &= - \sum_{k=1}^l (t_k - o_k) \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial y_j} = - \sum_{k=1}^l (t_k - o_k) \varphi'(net_k) w_{jk} \end{aligned} \quad (3.13)$$

This term represents how error at the output is affected by the output of a certain hidden unit. Similar to Equation 3.7, one can define:

$$\delta_j \triangleq \varphi'(net_j) \sum_{k=1}^l w_{kj} \delta_k \quad (3.14)$$

finally, leading

$$\Delta w_{ji} = \eta x_i \delta_j \quad (3.15)$$

The core idea of the algorithm comes from the propagation of errors in a backward manner as it can be seen from the derivation. Note that the derivation explains the backward propagation of errors for a single observation. The pseudocode for the algorithm can be examined in Algorithm 3.1.

The target values for each observation correspond to a vector of zeros except the class index of the row that the observation actually belongs to. That row is represented as a 1.

During training, a single sweep over all training data is called an *epoch*. The above-mentioned BP algorithm corresponds to the *stochastic backpropagation* algorithm in which the weights are updated after each observation (training data). The BP algorithm used in the thesis work is a *batch backpropagation algorithm* in which the weight updates take place after all or at least some of the training set is presented to the network. The idea of learning based on gradient descent is the same in both of the algorithms. Note that, this type of backpropagation algorithm introduces a new parameter to training, i.e., *batch size*.

```

1  begin initialize network topology(number of hidden units),  $w$ ,  $\eta$ ,  $r \leftarrow 0$ 
2  do  $r \leftarrow r + 1$  (increment epoch)
3       $m \leftarrow 0$ ;  $\Delta w_{ij} \leftarrow 0$ ;  $\Delta w_{jk} \leftarrow 0$ 
4      do  $m \leftarrow m + 1$ 
5           $x^m \leftarrow$  select pattern
6           $\Delta w_{ij} \leftarrow \Delta w_{ij} + \eta \delta_j x_i$ ;  $\Delta w_{jk} \leftarrow \Delta w_{jk} + \eta \delta_k y_j$ 
7      until  $m = n$ 
8       $\Delta w_{ij} \leftarrow \Delta w_{ij} + \Delta w_{ij}$ ;  $\Delta w_{jk} \leftarrow \Delta w_{jk} + \Delta w_{jk}$ 
9      until stopping criterion
10 return  $w$ 
11 end

```

Algorithm 3.1. Backpropagation algorithm for ANN training

The problem of finding where to stop the NN training is quite challenging as mentioned in the very previous section. Even though the training error decreases with more and more iterations, there is no guarantee to observe the same trend in the test error due to overfitting. The stopping criterion can differ for each application. For instance, a threshold level can be determined for the mean square training error and iterations can be stopped only if that error level is reached. A fixed *number of epochs* can also be used as a stopping criterion. For the implemented system, the parameter of number of epochs was commonly used to stop the learning. The decision of a proper epoch number has been made with the help of the validation data set as mentioned in the previous section.

3.4.2 DBN Training

The building blocks of deep belief networks are Restricted Boltzmann Machines. This is mainly because individual layers of a DBN share parametrication with an RBM and there are established methods to train them. *Contrastive divergence* algorithm is an unsupervised algorithm used to train RBMs. Note that RBMs are undirected graphical models in which there are no connections between units of the same layer. A simple example can be seen from Figure 3.4.

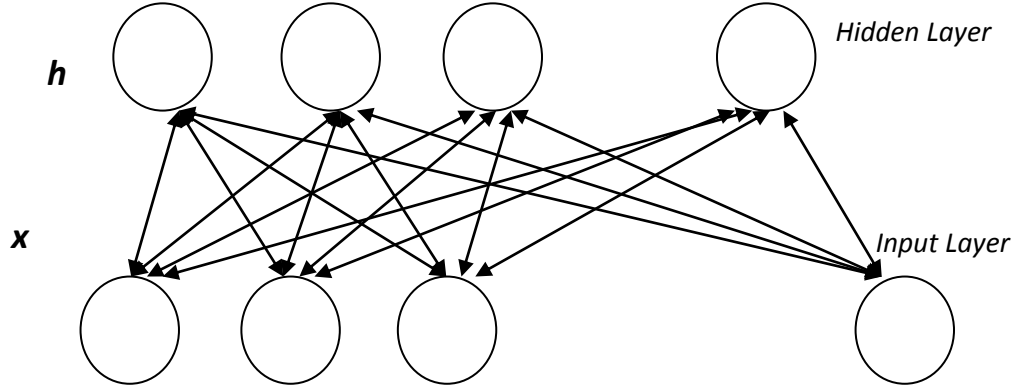


Figure 3.4. A RBM schematic with the input and hidden layer

As can be seen from Figure 3.4, each hidden unit (h) is independent of the other hidden units when conditioning on input units (x).

Contrastive divergence algorithm relies on an approximation of the gradient of the log-likelihood which is a successful update rule for training RBMs [6]. The main approximation is that the average over all possible inputs are replaced by a single sample. When an RBM is properly trained, it actually provides a representation of the distribution of the training data. Thus, it allows sampling from the learned distribution. The mathematical details of the CD algorithm, can be found in [6] and will not be presented in this work.

Assume a DBN with l hidden layers where the k^{th} layer is denoted by \mathbf{h}^k . The joint distribution between the observed vector \mathbf{x} and the hidden layers is

$$P(\mathbf{x}, \mathbf{h}^1, \dots, \mathbf{h}^l) = \left(\prod_{k=0}^{l-2} P(\mathbf{h}^k | \mathbf{h}^{k+1}) \right) P(\mathbf{h}^{l-1} | \mathbf{h}^l) \quad (3.16)$$

where $\mathbf{x} = \mathbf{h}^0$, $P(\mathbf{h}^{k-1} | \mathbf{h}^k)$ is the visible-hidden conditional distribution in an RBM at level k of a DBN and $P(\mathbf{h}^{l-1} | \mathbf{h}^l)$ is the top level RBM joint distribution. Note that the conditional distributions $P(\mathbf{h}^k | \mathbf{h}^{k+1})$ and the top level joint distribution, together define the generative model.

When a DBN is trained in a greedy layer-wise fashion, each layer is initialized as an RBM. The procedure can be described as greedy on account of the fact that it does not take into account whether there will be more layers to come. Let us denote the k^{th} RBM trained as $Q(\mathbf{h}^k | \mathbf{h}^{k-1})$. Note that $P(\cdot)$ denotes the probabilities according to DBN. The idea is that $Q(\mathbf{h}^k | \mathbf{h}^{k-1})$ is used as an approximation of $P(\mathbf{h}^k | \mathbf{h}^{k+1})$ as it is easy to compute.

Assuming \hat{P} is the input training distribution for the network and ϵ is the learning rate for the RBM training. If \mathbf{W}_k is assumed to be the weight matrix, \mathbf{b}_k to be the visible units offset vector for RBM, \mathbf{c}_k to be the hidden units offset vector for RBM at level k . Finally, assuming `mean_field_computation` is a Boolean that is true iff training data at

each additional level is obtained by a mean-field approximation instead of stochastic sampling and l is the total number of layers, the DBN training algorithm can be examined in Algorithm 3.2.

```

1   for  $k = 1$  to  $l$  do
2       initialize  $\mathbf{W}_k = 0, b_k = 0, c_k = 0$ 
3       while not stopping criterion do
4           sample  $\mathbf{h}^0 = \mathbf{x}$  from  $\hat{P}$ 
5           for  $i = 1$  to  $k - 1$  do
6               if mean_field_computation then
7                   assign  $\mathbf{h}_j^i$  to  $Q(\mathbf{h}_j^i = 1 | \mathbf{h}^{i-1})$  for all  $j$  of  $\mathbf{h}^i$ 
8               else
9                   sample  $\mathbf{h}_j^i$  from  $Q(\mathbf{h}_j^i | \mathbf{h}^{i-1})$  for all  $j$  of  $\mathbf{h}^i$ 
10              end
11          end
12          update RBM
13      end
14  end

```

Algorithm 3.2. DBN training algorithm

Here, the stopping criteria can be the number of epochs, a threshold for the reconstruction error on the validation set or the average free energy difference between the validation set and a subset of training set. *Step 12* in the algorithm corresponds to the contrastive divergence algorithm in which the important aspects of the input distribution (which is not available) is learned based on the samples from it.

After DBN training the weights are used to initialize a DNN training with standard gradient-based learning procedure (backpropagation). In this way, backpropagation algorithm starts learning the network weights near a relatively better solution compared to random initialization. This enhancement directly reflects to the classification performance.

3.5 Classifier

As frame blocking has been conducted to the test data too, each audio file is represented as multiple observations (one feature vector for each frame). The output values of the NN can be interpreted as a probabilities (not strictly speaking, as they may not sum up to 1) or likelihoods for belonging to that class index. This situation results in two possible ways of classification of a file. One is classification by majority vote and the other is classification by mean of output likelihoods.

In the classification by majority vote case, each observation frame from an audio file is assigned to the class that yields the maximum likelihood considering the NN output.

Then a majority voting is done to decide the class of the file by selecting the maximum number of votes among all classes.

In the classification by mean of outputs case, the outputs for every observation frame belonging to that file are averaged and the class index of the maximum value decides the class of the file.

Experimental results show that both of the implementations give very close results. For this work, classification by mean of output likelihoods has been used to define the classification performance of the system at all times.

4. EVALUATION

In this chapter, the setup for evaluation is described in detailed as well as the effect of certain parameters on classification rate. The results are shown in several plots and tables. The performances of ANN and DBN are compared for several cases.

4.1 Data and Platform

The database used in all of the experiments was provided by Audio Research Team under Multimedia Research Group in Tampere University of Technology. The database include real life recordings of 61 distinct classes such as sneezing, dog barking, clapping, car door, beep, yelling. The database is a collection of isolated sound events which was retrieved from the Stockmusic online sample database¹.

The data is divided into training, validation and test data as 80%, 10% and 10% respectively at all times. A 10-fold cross-validation is performed and the classification rates are the averaged rates of the all 10 folds. There are in total 1325 audio files. Both the number of files per class and the average length of files per class is very uneven which makes the classification problem harder. The minimum number of files for a class is 10 (cutting knife, human coughing, pen writing and shopping cart). The maximum number of files for a class is 94 (footsteps). The shortest file is about 0.3 ms (dishes spoon) and the longest file is about 3 min 46 s (crowd conversation) long. The histogram of number of files per class and the histogram of average number of frames per class can be seen in Figure 4.1. The second histogram was plotted assuming a frame length of 50 ms but the idea of the plot is to give a comparison of average length of files per class which is independent of frame length.

¹ <http://stockmusic.com/>

The implementation platform for the whole work was the engineering tool MATLAB. In the beginning of work, MATLAB's built in neural network toolbox was used for the purpose. Later on, an open source toolbox (DeepLearnToolbox [66]) was used as it had certain performance benefits over the former. In addition, for feature extraction, certain functions of MIRtoolbox [55] which has been developed in the University of Jyväskylä, was used.

4.2 Evaluation Setup

As NN training (as a result the performance) is dependent on numerous parameters, the effect of a parameter on classification performance is examined by keeping the other parameters constant (most of the time). One has to keep in mind that a NN is a highly non-linear structure in its nature and the effect of these parameters are not independent of each other. Therefore, the effect of a parameter on performance may change for different constant values of other parameters. For example, the effect of learning rate on classification rate may differ from topology to topology. The NN parameters include:

i) Number of hidden layers: This value determines the depth of the network. The higher the value is, the deeper the network. The theoretical expression power of a NN (assuming it does not get stuck in a local optima) is highly related to this number.

ii) Number of units in each layer: The number of units in each layer can be different for each layer. These values determine the expression power of the NN directly by determining the number of weights in total.

iii) Number of most energetic frames: The NN training is computationally expensive. Thus, instead of using all of the observation frames from a file, R most energetic frames were used. Another reason for such a design is to keep the number of observations from each file constant. Throughout the experiments R was taken 120 at all times. The files which do not contain that many frames were duplicated in order to reach to that number. The comparison of energies of frames in each file was conducted using the first MFCC which is not included as a feature.

iv) Number of features: This parameter corresponds to the number of MFCCs or to the number of mel energy bands used to extract mel-energies. The dimension of the data is automatically determined by this value. Throughout this work, audio data was represented by MFCCs unless otherwise mentioned explicitly, where in that case mel-energies from either 20 or 40 energy bands were used. When MFCCs were chosen to be the features, 12 MFCCs were calculated after discarding the first one which corresponds to the total frame energy.

v) Number of adjacent frames: As well as current frame MFCCs, the adjacent frame MFCCs are also used as features. This number, A , was decided to be 2 (meaning 2 left adjacent and 2 right adjacent); leading to 60 dimensional input vectors (12 MFCCs from each frame).

vi) Division of training, validation and test: The division of training, validation and test data affects the performance of the system. As mentioned in the previous section,

the division is done as 80% training, 10% validation and 10% test data at all times. 10-fold cross-validation is used for statistical reliability.

vii) *Neural network activation function*: The activation function for each unit can actually differ (for units on the same layer it is usually the same). A sigmoid activation function was used for all units in the implementation.

viii) *Number of epochs*: The BP algorithm runs until a certain number of epochs is reached. The iterative nature of the algorithm will decrease the training error with increasing number of epochs; however, it is an easy mistake to overtrain (overfit) the network.

ix) *Batch size*: As mentioned in Chapter 3 the BP algorithm is implemented as a batch backpropagation algorithm. Thus, the batch size should be decided.

x) *Learning rate*: Learning rate is the scalar that determines the amount of change in the gradient towards the proper direction. It is taken as $\eta = 1$ for this work unless stated otherwise.

xi) *Momentum*: Setting a too small learning rate will result in smooth trajectories but leading to longer computation times. On the other hand, a too high learning rate may result in an unstable system. For this purpose, a parameter called momentum is introduced which changes the learning rate adaptively simply by a feedback loop. It acts as a booster on the learning rate if a plateau in the error surface is encountered or vice versa. It is denoted as α and taken to be 0.5 at all times in this work.

The abovementioned parameters are relevant to ANNs whose weights are initialized randomly to small numbers. Note that if the network weights are to be initialized by DBN training, the unsupervised pretraining introduces its own parameters to the system in addition to the ones above. These are:

i) *Number of epochs in unsupervised training*: The unsupervised learning algorithm is also iterative and the number of total sweeps through the training data should be decided.

ii) *Batch size for unsupervised training*: This value was selected to be 100 for all of the experiments.

iii) *Momentum for unsupervised training*: This value was also constant for all of the experiments and set to be 0.

4.3 Results for NNs with randomly initialized weights

In this section, the performance of a NN with backpropagation training for the given acoustic event classification problem is presented. Effect of certain parameters on the classification rate is also introduced.

4.3.1 Effect of network topology

The network topology refers to the network architecture meaning the number of layers and number of hidden units in each layer. Certain experiments are conducted with

several different architectures. The conclusions are made not only by examining the classification performances but also analyzing the training, validation and test errors.

The structural aspects of a NN directly determine the number of weights in a network which can be considered as a *degree of freedom* for expression. Thus, a very rough deduction can be made to guess the upper limit of the weights: one should avoid having the number of weights more than the number of training examples.

NNs with 1 Hidden Layer

A NN with a single hidden layer is trained several times by changing the number of units in the hidden layer. There are two main scenarios. One is that the BP training is stopped at an epoch number of 50. The second one is that the BP training stops by checking the epoch where the validation error starts to increase (overfitting starts). All the other parameters were kept constant and can be seen from Table 4.1.

Table 4.1. *The parameter values that are kept constant for comparison of classification accuracy for two different early stopping conditions.*

Parameter	Value
Features	MFCCs
Number of feature coefficients	12
Number of most energetic frames used	120
Number of adjacent frames for additional features	2
Batch size	25
Number of hidden layers	1
Momentum	0.5
Learning rate	1

From Figure 4.2 one can deduce a few things. For example, for lower number of hidden units, the validation stop performance is more or less similar to the epoch-stop performance, 30-35%. One can say that 50 epochs were enough for this certain application and smaller topology, to achieve the performance of the validation stop. However, for larger number of units, 50 epochs are not enough to learn the input-output relations. Actually these topologies perform worse than smaller ones if only 50 epochs are granted them as their learning process is slower. So the decline in the blue curve is due to underfitting. But if a validation stop is applied, the algorithm takes its time to learn the data properties in a successful manner and reach around 40% classification accuracies. In general, this difference is even more significant with increasing number of hidden layers which is also discussed in this chapter.

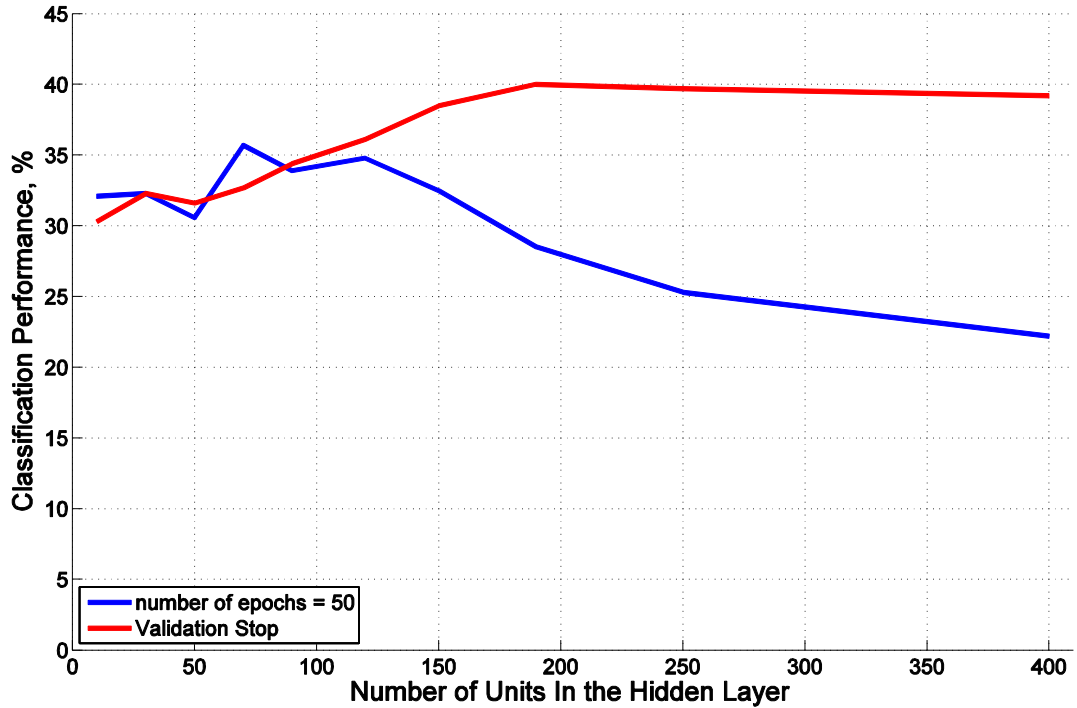


Figure 4.2: Comparison of two early stopping techniques on classification performance for several number of hidden units. Stopping criteria are validation stop and stop when epoch count reaches 50.

In order to examine the opposite cases, one 10 unit and one 250 unit single hidden layer NN are trained. The remaining parameters are kept constant with the values in Table 4.1. The learning curves including the validation and test error trajectories can be seen from Figure 4.3. The training error gets much lower for the larger network as the number of epochs increase. The minimum points of the validation curve are also marked with black markers on the plot corresponding to epoch number 10 and 36 respectively. The validation error starts to increase much sooner for the smaller network. Yet again, one can not guarantee the test error will act as the validation error.

There have been many cases in which the validation and test errors did not act correlated enough. This might be mainly due to the database. The distribution of number of files per class is extremely heterogeneous as explained in the previous chapter. The length of files also differ a lot especially between classes.

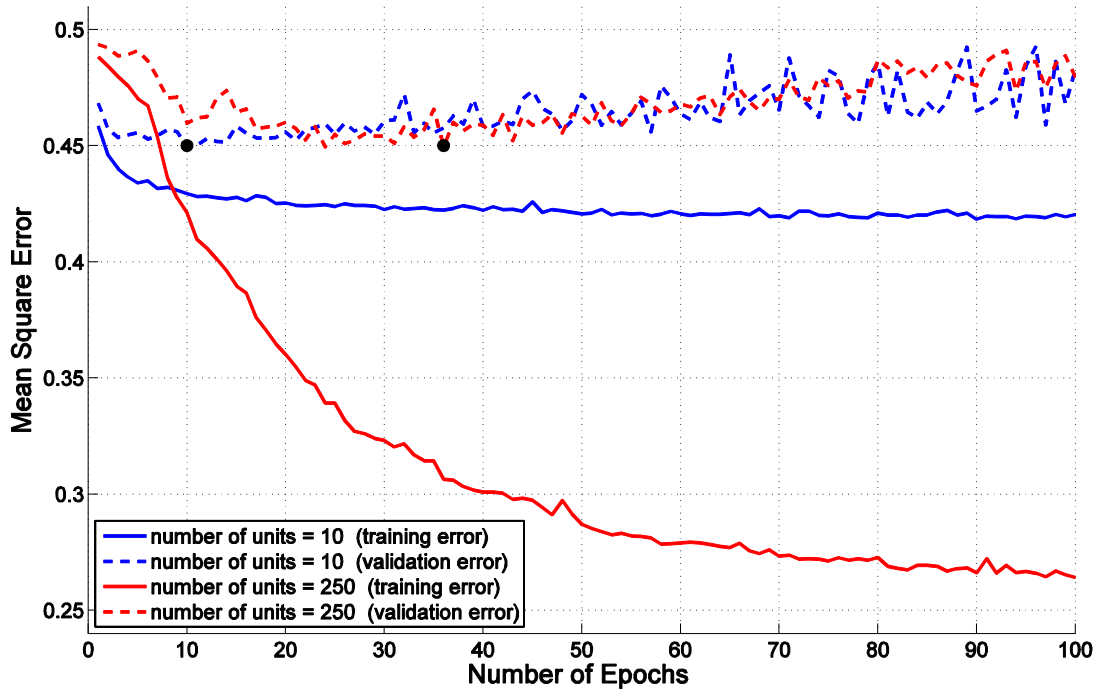


Figure 4.3. The learning curves (training error and validation error) for two single-hidden layer NNs, one having 10, other having 250 units.

NNs of 2 Hidden Layers

It is desired to get some insight about the choice of number of units for each layer in a NN for a better design, at least for this application. For this purpose, the number of units for both layers were swept between certain ranges while keeping the other network parameters constant. The increment value for the number of units is not constant. The resolution for low number of unit values is higher as the significance of adding new units is much more visible in that part.

In order to obtain the results in Figure 4.4 every single parameter including the number of epochs were kept constant (see Table 4.2). On the other hand, the Figure 4.5 is obtained by early stopping considering the validation error. Thus the number of trained epochs for each data point is different while the others are as in Table 4.2.

Table 4.2. The parameter values that are kept constant for comparison of classification accuracy for different network topologies.

Parameter	Value
Features	MFCCs
Number of feature coefficients	12
Number of most energetic frames used	120
Number of adjacent frames for additional features	2
Batch size	25
Number of hidden layers	2
Number of epochs	50
Momentum	0.5
Learning rate	1

When the results are compared from Figure 4.4 and Figure 4.5, the first significant point is that if there are many units in both layers (black and pink curves), the system becomes very powerful in learning the training data but fails to generalize even if a validation set is used for early stopping.

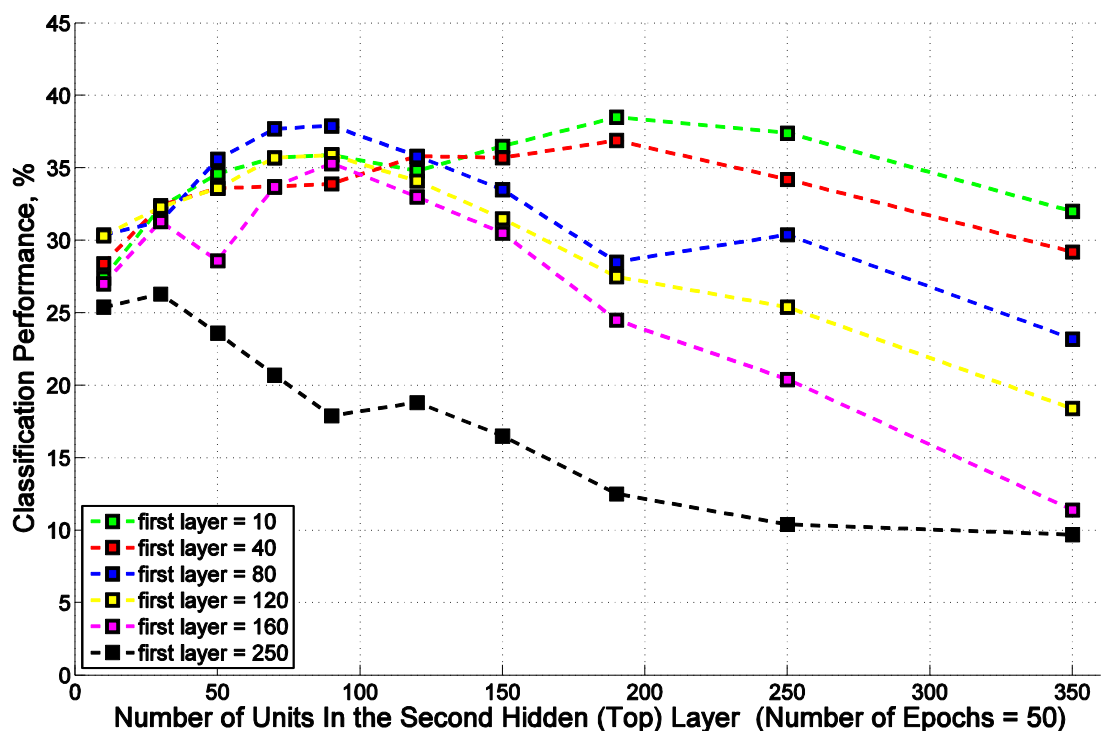


Figure 4.4. Comparison of classification performances for different combinations of number of units for a 2 hidden layer network with early stopping of 50 epochs.

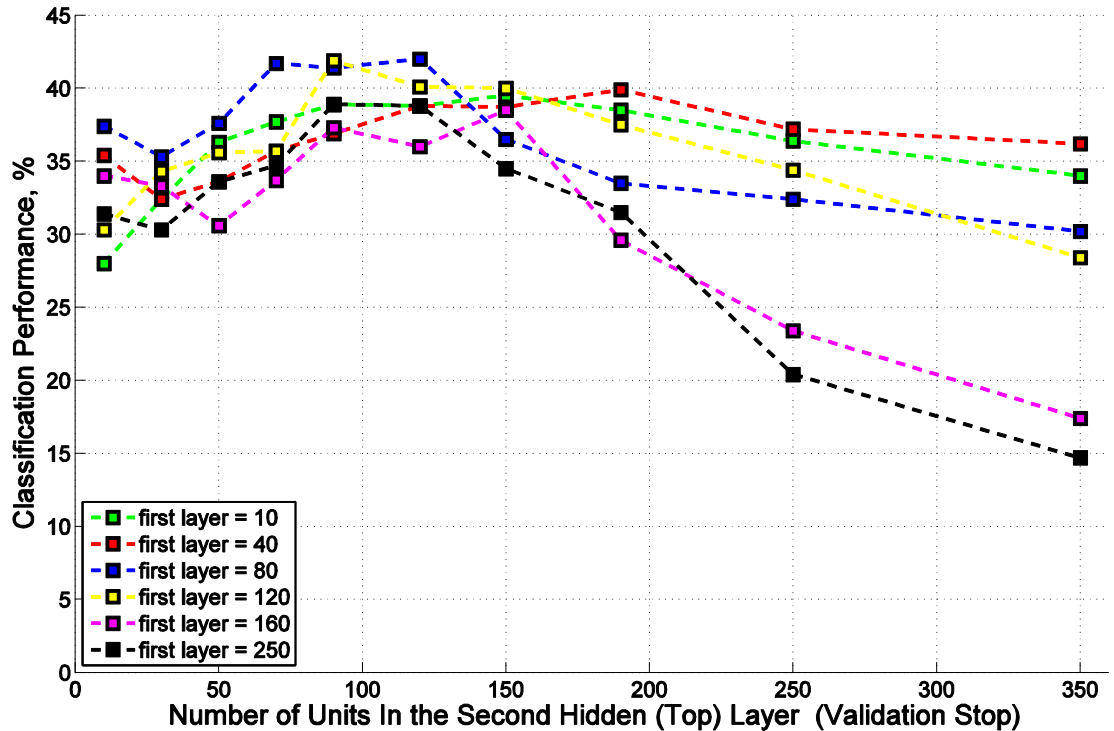


Figure 4.5. Comparison of classification performances for different combinations of number of units for a 2 hidden layer network with early stopping of validation error.

The training errors of the network at epoch 200 can be examined from Figure 4.6. This plot should not be confused with a learning curve. This plot shows the value of the training error when the training stopped at 200th epoch for different unit numbers. Obviously, the effect of the number of hidden units at the top level is huge. Due to its nature, the BP algorithm is very successful to tune the weights close to the outermost layers of the network (close to the output). Thus, increasing the number of hidden units in the outermost layer will result in better learning. In most cases this will decrease the training error independent of the number of layers in the network. The reason is that BP algorithm tends to fail in training the lower layers (closer to input) and if there is enough expression power in the top hidden layer, a very low training error can still be achieved. This, though, does not guarantee a lower test error. In fact in many cases it is shown to be overfitting. Setting the number of units for the top hidden layer somewhat comparable to the number of output units (number of classes) can be a good practice in general.

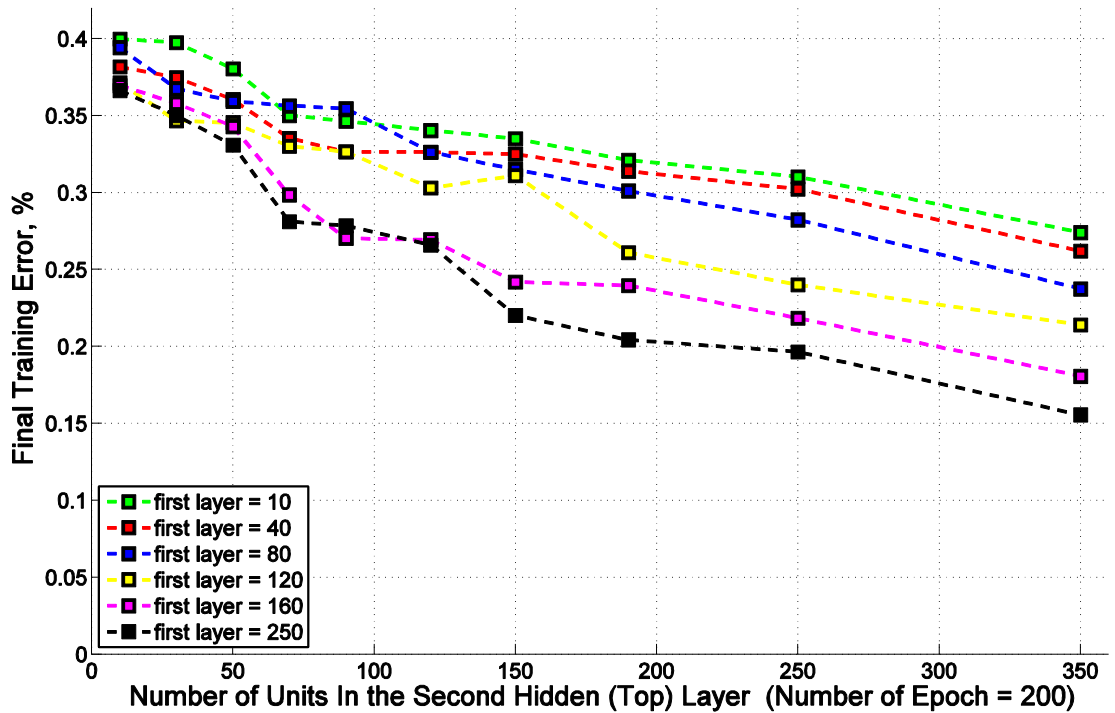


Figure 4.6. Comparison of final training error values for different combinations of number of units for a 2 hidden layer network with early stopping of 200 epochs.

NNs of 3 Hidden Layers

Due to computational restrictions, analyzing the effect of number of units for each layer on the classification rate is quite burdensome for deeper network structures. In addition, the estimated network topology to give high performance will be valid for this specific classification problem (acoustic event classification in this case) which would be no help for other applications. Thus, for networks with 3 or more hidden layers, a meaningful number of unit can be set to examine the effect of other parameters. For each layer, 70 units have been chosen for such networks. In the following section the performance of NNs with 3 hidden layers are given with respect to different batch sizes for training.

4.3.2 Effect of batch size

The effect of batch size on the performance is experimented in this part. Training was stopped by taking the validation set into account for each batch size. Even though varying batch size resulted in certain fluctuations in the classification rate, it is not a considerable amount. When it comes to computational burden, lower batch sizes require much more time for each epoch (around 8.5 seconds for batch size 25), while higher ones require less (around 2.7 seconds for batch size 400). As an epoch means one time sweep through whole training set, the smaller batch sizes require more weight updates. Thus, this is expected. On the other hand, results show that the validation stop epoch numbers are much lower for small batch sizes. This can also be guessed. If in one

epoch, small batch sized network do so many weight updates, then in terms on per-epoch, the convergence to a better solution is much faster for smaller batch sized networks. The results can be seen from Figure 4.7 and Figure 4.8 and the remaining parameter values can be examined from Table 4.3.

Table 4.3. The parameters that are kept constant for classification accuracy comparison for different BP algorithm back sizes.

Parameter	Value
Features	MFCCs
Number of feature coefficients	12
Number of most energetic frames used	120
Number of adjacent frames for additional features	2
Number of hidden layers	3
Number of units in hidden layers	(70,70,70)
Momentum	0.5
Learning rate	1

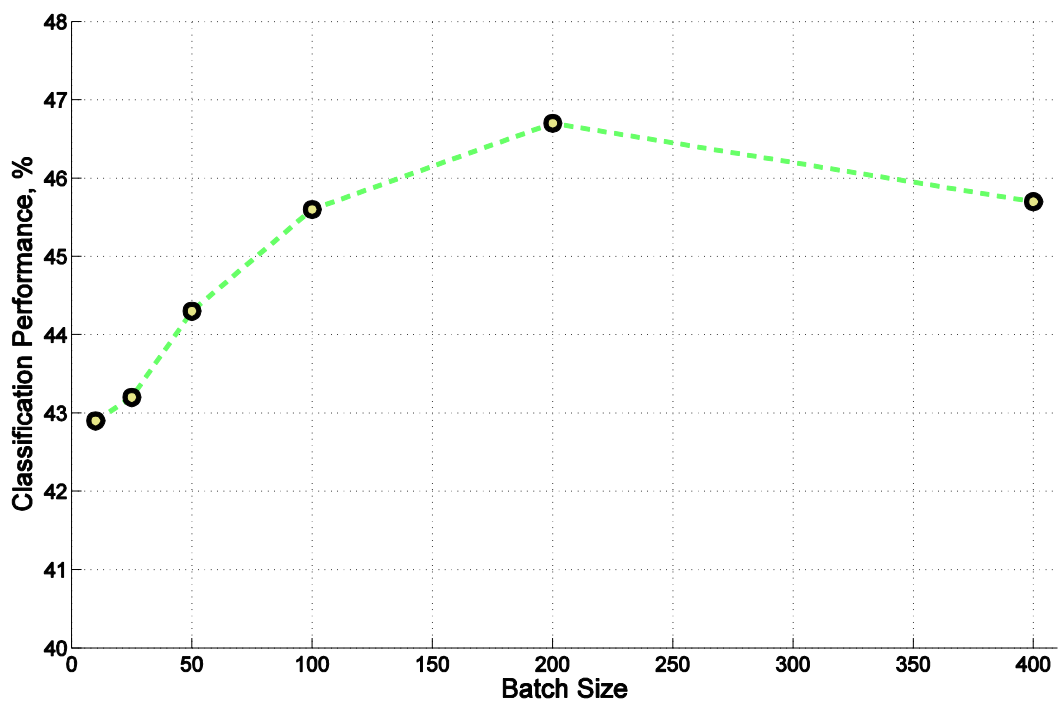


Figure 4.7. Comparison of different batch sizes in terms of classification performance while the other parameters being constant.

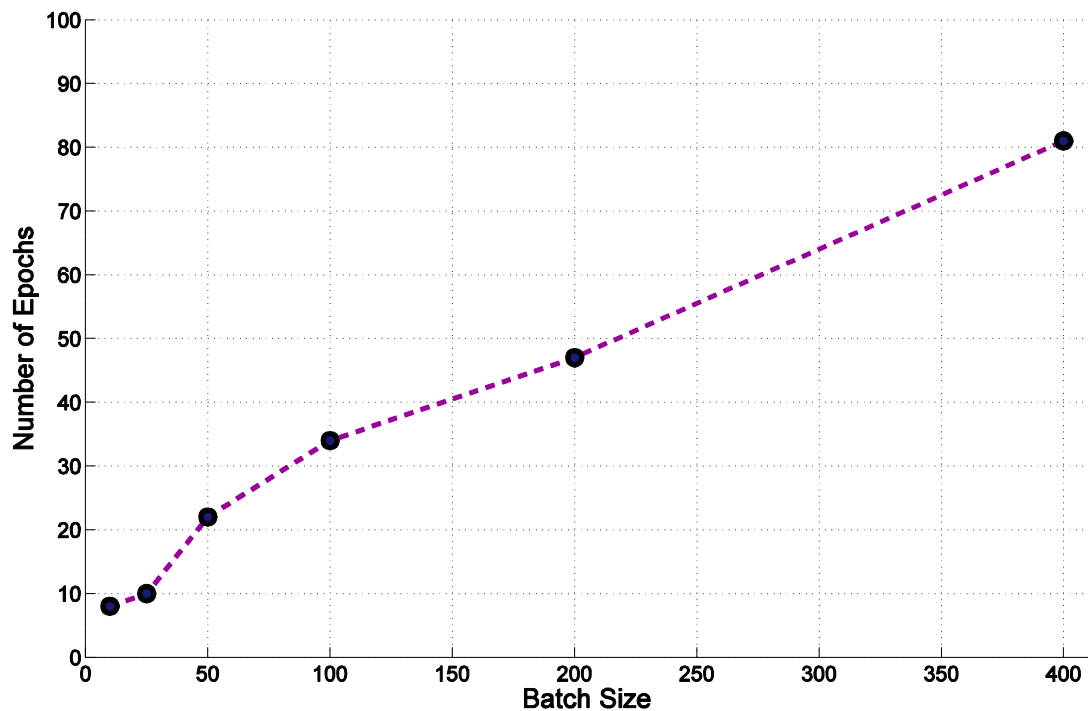


Figure 4.8. The epoch number in which the validation error stopped decreasing for different batch sizes.

4.3.3 Effect of number of adjacent frames

As mentioned before, the information contribution of the adjacent frames on the classification of the frame under investigation was also considered. The feature vectors from a certain number of left and right adjacent frames were concatenated with the current frame's feature vectors to form a higher dimension feature vector to represent the data. The effect of the number of adjacent frames to take into consideration on the classification rate can be seen from Figure 4.9. The remaining parameter values are listed in Table 4.4. For the experiment, the network topology consisted of 2 hidden layers each containing 50 units. The batch size was chosen to be 100 and the training algorithm was run for 50 epochs.

Table 4.4. The parameters that are kept constant for classification performance comparison for different number of adjacent frames used for additional features.

Parameter	Value
Features	MFCCs
Number of feature coefficients	12
Number of most energetic frames used	120
Batch size	100
Number of hidden layers	2
Number of units in hidden layers	(50,50)
Number of epochs	50
Momentum	0.5
Learning rate	1

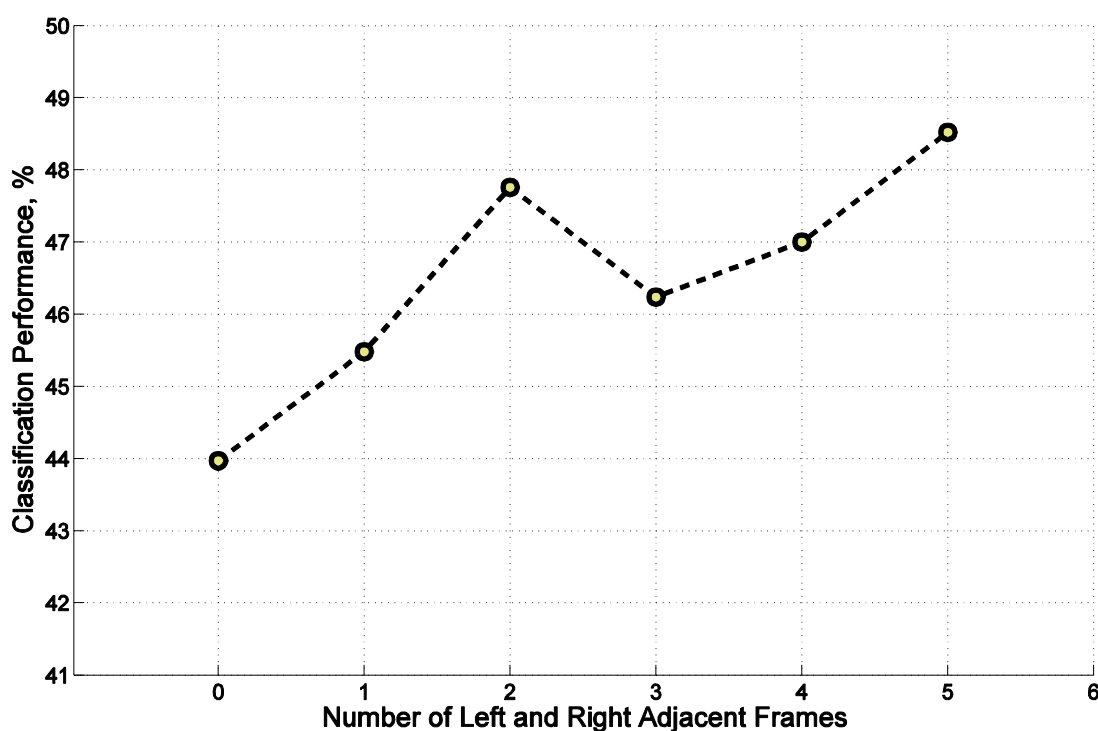


Figure 4.9. Classification accuracies for different number of left and right adjacent frames that are taken into consideration for additional features, while the other parameters are kept constant.

The classification rates varied between 43.97% and 48.52% for different trials of 0 to 5 adjacent frames to be used. The results imply that it is not easy to come up with a precise conclusion for the effect of this parameter on the performance as the classification rates are rather close. Hence, it would be appropriate to say that it is not obvious whether the differences in classification performance come from the parameter under investigation or from the randomness in the implementation (random initialization of network weights, random division of training and test data).

Under the light of these results, choosing any of these values for the number of adjacent frames seem equally suitable, 5 being a little more favorable. On the other hand, as the dimension of data is directly affected by this parameter, the computational cost should also be considered. Assuming a frame represented by 40 mel-energies, a value of 5 (5 left and 5 right adjacent) would result in 440 dimensional data and this would be reflected to the topology of the input layer of the network. Such a situation will result in much higher computational cost especially in the training phase. Consequently, to avoid that, a value of 2 was decided to be used at all times assuming it would provide a good enough representation of dynamic properties of the data but at the same time prevent too many calculations.

4.3.4 Effect of feature extraction

Intuitively, one would assume that the way of representing a data would influence the performance of classifying it. Under such a guidance, the effect of representing the audio data with two possible set of features, namely MFCCs or mel-energies, was decided to be understood. An ANN with 2 hidden layers with 70 unit in each layer was trained for 100 epochs. The batch size, the number of adjacent frames and the learning rate were chosen to be 100, 2 and 1, respectively. A 10-fold cross-validation was performed to obtain the classification performance results.

When the data was represented by 12 MFCCs, the classification rate was 49,27%. When 20 and 40 mel-energy coefficients were chosen to represent the data, the classification rates were 53,82% and 59,39%, respectively.

The improvement on the performance when mel-energies were used instead of MFCCs for feature parameters may be due to several reasons. One of the reasons could be that NNs are powerful tools to represent high dimensional, highly nonlinear relations and MFCCs acted too compact. A moderate (in terms of compactness) representation with more coefficients resulted in better performance. In addition, BP algorithm is known to be efficient when learning from low-level representations (for example directly from pixel values in image processing [18]) and mel-energies can be considered as such representations of audio data.

Highest Classification Rate

The ANN parameters and their values when the highest classification rate was achieved are presented in Table 4.5. After 10 fold cross-validation a classification accuracy of 61.14% was achieved. This classification accuracy may increase for other possible combinations of network parameters that have not been tried. One should be aware of that the mentioned result excels the classification accuracies that were obtained by conventional HMMs with GMMs used on the very same database [36]. In that case the classification rate was around 54% which underlines the success of NN classifier in AEC tasks.

Table 4.5. *The highest classification rate achieved by ANN and the corresponding design parameters.*

Classification Rate	61.14%
Features	Mel energies
Number of feature coefficients	40
Number of most energetic frames used	120
Number of adjacent frames for additional features	2
Batch size	100
Number of hidden layers	2
Number of units in hidden layers	(70,70)
Momentum	0.5
Learning rate	1
Number of supervised epochs	140

The confusion matrix for this task is huge, being a 61 by 61 matrix. Therefore, proper visualization of the whole matrix as it is, is not feasible. Table 4.6 shows a smaller CM which was obtained by taking only a few classes into consideration. It was observed that some classes were easier to classify while the others were not. However, there was no significant correlation between the number of files belonging to a class and the classification rate of that class.

Table 4.6. *Confusion matrix for 11 randomly chosen classes*

	Bi	Ca	Ch	Do	Fo	GSh	HSn	PCu	Ra	St	Th
Bicycle	6	-	-	-	-	-	-	-	2	-	-
Car	-	6	-	-	-	-	-	-	-	-	4
Chainsaw	-	-	6	-	-	-	-	-	2	-	-
Door	-	-	-	15	-	2	-	-	-	-	-
Footsteps	-	-	-	-	88	-	-	-	-	-	-
Gun Shot	-	-	-	1	-	16	-	-	-	-	-
Human Sneeze	-	-	-	-	-	-	1	-	-	-	-
Paper Cutter	-	-	-	1	-	-	-	10	-	-	-
Rain	-	-	1	-	-	-	-	-	41	-	-
Stapler	-	-	-	-	-	2	-	-	-	4	-
Thunder	-	-	-	-	-	-	-	-	-	-	35

4.4 Results for NNs with DBN pretraining

As mentioned before, BP algorithm encounter problems in training deeper networks properly due to the difficulty of training the lower layers. The backward error propagation fails to make significant changes on the weights of the lower layers. DBNs

overcome this phenomenon by initializing the network weights to better values before supervised training. The effect of unsupervised layer-wise training is thus experimented in both shallow and deep architectures.

Table 4.7. *The parameters that are kept constant for ANN and DBN classification performance comparison for different number of hidden layers.*

Parameter	Value
Features	MFCCs
Number of feature coefficients	12
Number of most energetic frames used	120
Number of adjacent frames for additional features	2
Batch size (unsupervised)	100
Batch size (supervised)	100
Number of units in each layer	70
Momentum	0.5
Learning rate	1
Number of supervised epochs	120
Number of unsupervised epochs	1

Starting from a single hidden layer, both ANN and DBN are tested with the same training parameters up to 6 layers. The parameters for the experiment can be seen from Table 4.7. The results can be considered to be expected and visualized in Figure 4.10. The greedy, layer-wise pretraining did not apply much significant effect on shallow network performances (both around 45% for 2 hidden layers). On the other hand, DBN's performance increased when the network became deeper (50.1% for 5 hidden layers). Note that for deeper networks, ANNs perform worse than they do in shallow ones (24.7% for 5 hidden layers).

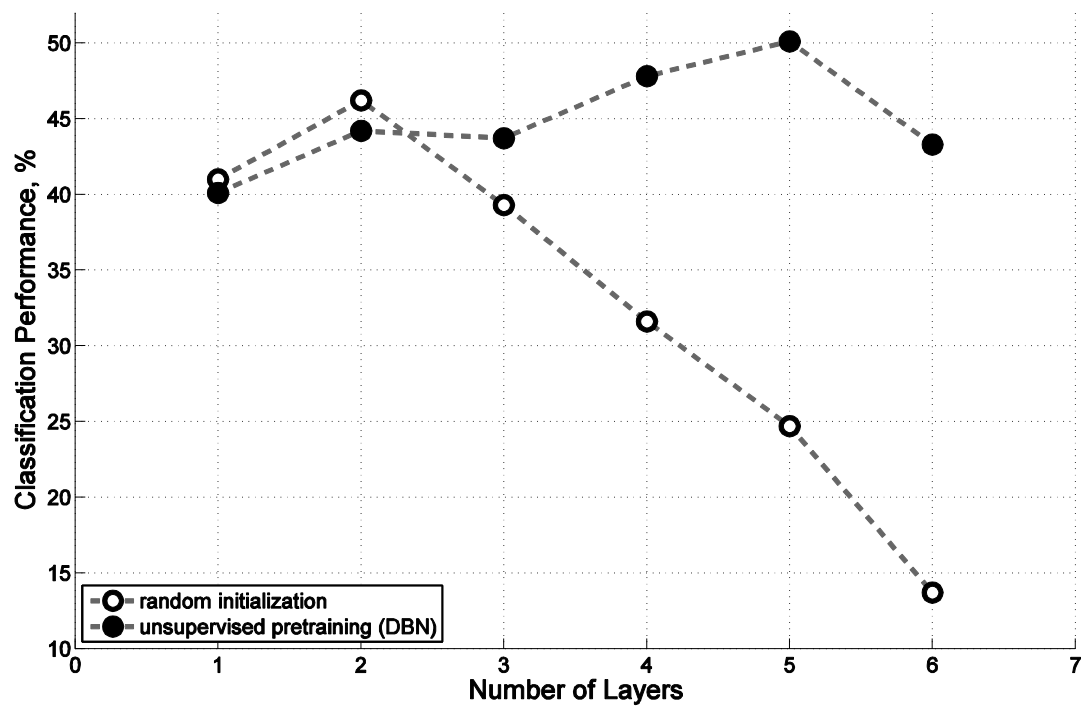


Figure 4.10. Comparison of classification performances of ANNs with random weight initializations and with DBN pretraining weight initializations for different network depths.

When one compares the learning curves for an ANN and a DBN, there are certain patterns. For example, in Figure 4.11, even though in the end the training error decreases to the same value, ANN takes the fast way to there, while DBN avoids sharp slopes. This may be due to the reason that standard ANN starts with random initializations and suddenly encounters a big gradient difference. Then, with the help of momentum, it moves fast in the error plane. Note that, it is obvious that DBN already starts from a better training and validation error value due to unsupervised pretraining.

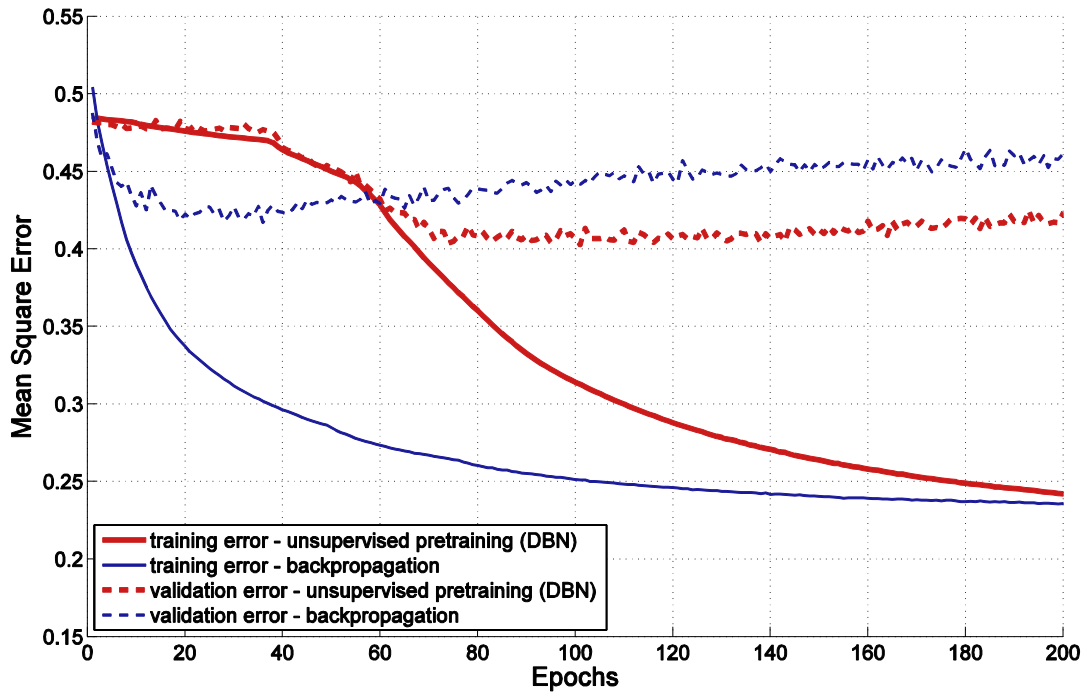


Figure 4.11. The learning trajectories of ANNs with random weight initializations and with DBN pretraining weight initializations for validation and training errors.

There is an important point that is worth mentioning in the unsupervised pretraining process. It was observed that, if the pretraining part uses a small amount of data (only certain number of most energetic frames for example) just like the supervised part, then immediately after the first unsupervised epoch, overfitting was encountered. In such a situation, the NN weights were initialized to extreme values in magnitude and therefore yielding unbelievably bad results. When all of the frames were used, it was easier to run the unsupervised training at least for a few epochs. The results are presented in Figure 4.12 while the list of the remaining parameters that are kept constant can be seen from Table 4.8. As adding more data to the pretraining part is not as much burdensome as doing so in the supervised part, one should keep this in mind when an overfitting problem is encountered.

Table 4.8. The parameters that are kept constant to experiment the effect of number of unsupervised epochs on classification accuracy in DBNs.

Parameter	Value
Features	Mel energies
Number of feature coefficients	40
Number of most energetic frames used (supervised)	120
Number of adjacent frames for additional features	2
Batch size (unsupervised)	100
Batch size (supervised)	100
Number of units in each layer	70
Momentum	0.5
Learning rate	1
Number of supervised epochs	100

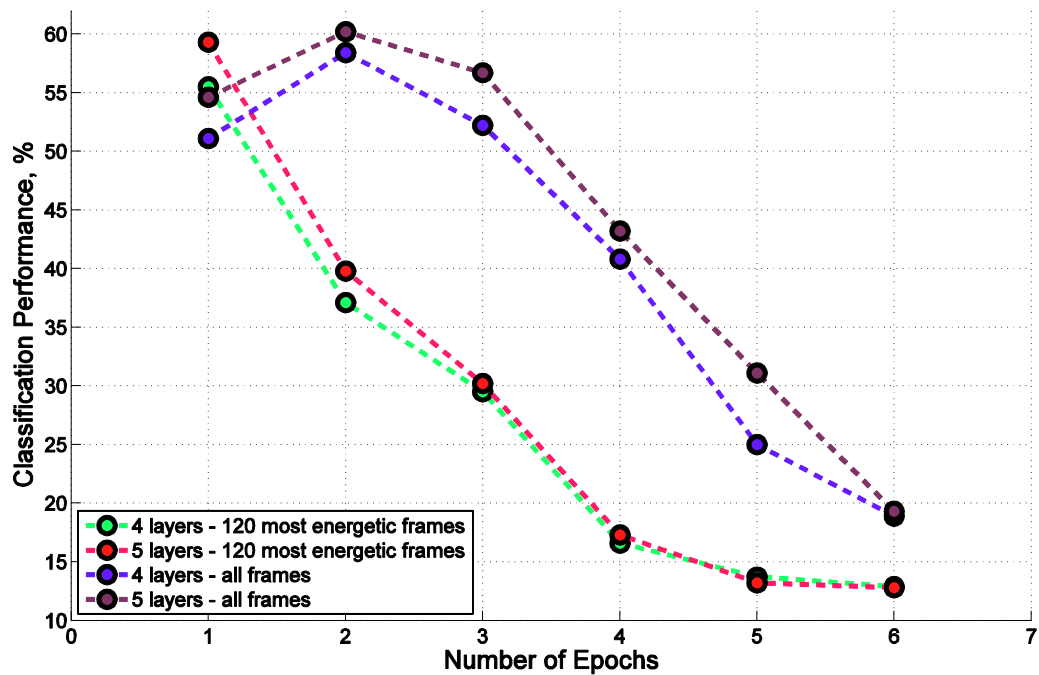


Figure 4.12. The classification accuracies when different amount of data was used for unsupervised pretraining for 4 and 5 hidden layer networks.

Highest Classification Rate

The DBN parameters and their values when the highest classification rate was achieved are presented in Table 4.9. After 10 fold cross-validation a classification accuracy of 64.7% was achieved. The increase in classification error compared to the highest classification rate achieved by randomly initialized ANNs is around 3.5%.

Table 4.9. *The highest classification rate achieved by DBN was obtained by such parameters:*

Classification Rate	64.7%
Features	Mel energies
Number of feature coefficients	40
Number of most energetic frames used	120
Number of adjacent frames for additional features	2
Batch size	100
Number of hidden layers	5
Number of units in hidden layers	(50,70,70,70,50)
Momentum	0.5
Learning rate	1
Number of supervised epochs	120
Number of unsupervised epochs	2

To sum up, networks with 4 or 5 hidden layers, layer-wise unsupervised pretraining gives further performance improvement. One also should not forget that the abovementioned statement has been proved to be true for certain values of implementation parameters. Thus, even for deep networks, for extreme values of batch size, learning rate etc. may result in the contrary (unsupervised pretraining performance falls behind regular one). However, those extreme values are most likely to result in very low classification rates for both cases.

All of the classification rates mentioned in this work are founded by averaging the classification rates of a 10-fold cross-validation. In order to eliminate the chance of getting lucky for all of the folds of a specific cross-validation, 20 10-fold cross-validations are run with the parameters mentioned in Table 4.3. The histogram of the classification rates are presented in Figure 4.10.

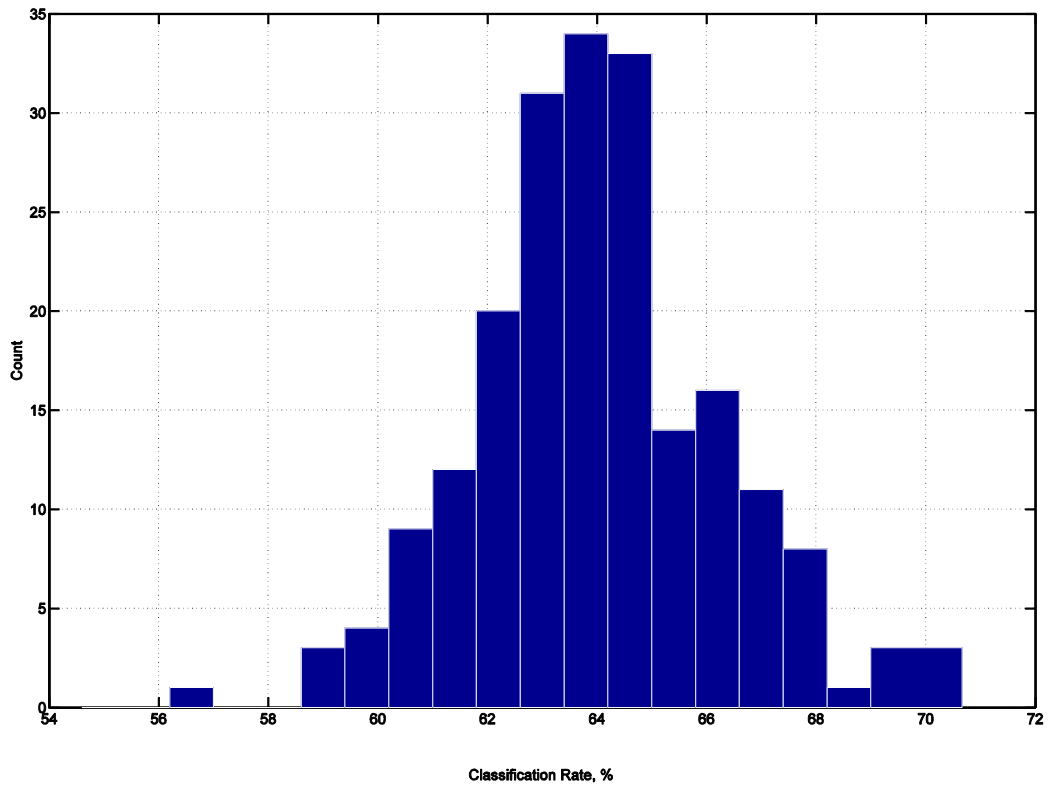


Figure 4.13. The histogram of classification accuracy for 20 different 10-fold cross-validation runs (in total 200 folds).

5. DISCUSSION AND CONCLUSIONS

The field of interest of this thesis work was acoustic event classification and the emphasis in terms of pattern recognition tools was given to two main concepts; one being artificial neural networks and the other being deep belief networks. First, several ways to efficiently represent the data under investigation, i.e., audio data, has been discussed. Understanding the working principles of the abovementioned pattern classifiers was essential to benefit from them legitimately for the given field. Therefore, afterwards, considerable amount of time has been spent on extensive task-dependent parameter optimizations for these classifiers. The whole process revealed the proper features to be extracted from the data and underlined the strengths of ANNs and DBNs to reach a state-of-the-art performance.

Two possible set of features were decided to be extracted from the audio data, namely MFCCs and mel-energies. These features were shown to be successful in representing the data with the help of NN classifiers. Mel-energies gave a better representation, possibly because of having more coefficients that sweeps over a high range of frequencies.

Parameters of the established acoustic event classifier have been examined extensively. One of them was the number of adjacent frames to consider, in order to have additional features. The change in the performance when adjacent frames were used to provide additional features was not very bold.

The effect of NN topology on the classification performance was also examined. Several combinations of number of hidden layers and number of units in each layer were tried out to understand their effect. One of the main findings of this thesis work was revealed, i.e., an ANN is successful in classifying various acoustic scenes (almost 60% classification accuracy) if it has a shallow architecture. When the architecture becomes deeper, the NN training fails to perform properly for the network to give desired performance.

The need for investigation of training algorithms for deep neural networks was explained and another important finding of the thesis was revealed. With the help of DBNs, the effect of unsupervised pretraining on enhancement of classification performance was observed. The increase in classification rate was found to be 2-4% for a pretrained DNN when compared to a shallow ANN. The unprofitable use of DBNs for shallow networks as well as the overtraining power of them with the increased number of unsupervised training epochs were also observed.

The classification performance of DBNs (around 63%) were shown to be considerably better than that of conventional audio classifiers that utilizes HMMs with GMMs on

the same database [36] (around 54%). With their highly nonlinear expression power, when trained properly, NNs were proven to be applicable to acoustic event classification tasks.

To increase the classification rate for both shallow ANNs and pretrained DNNs, certain actions could have been taken. All of these actions play a role of strong standpoints for further research on the topic. For example, for the feature extraction phase the effect of number of MFCCs and mel energy coefficients on classification performance can be examined. The effect of other spectral and temporal features can too be inspected.

The parameter of number of adjacent frames did not have a considerable impact on classification accuracy. As dynamic features usually introduce more information about the audio data and result in better performance, the established way to do so can be questioned and modified. For example, instead of using the feature vectors of the adjacent frames as additional features, the derivatives of features (e.g. delta MFCCs) can be used. One other way to highlight the most energetic frames during training can be to input those feature vectors as observations to the NN training.

As the NN classifier performances were shown to be extremely dependent on both implementation parameters and network parameters, several different combinations of these parameters can be tried. This will surely increase one's knowledge on stopping criteria for training algorithms and methods to monitor certain metrics related to overfitting. The already-gained intuition in NN and DBN trainings by constantly trying to avoid overfitting will be further improved by this way.

Furthermore, it is suggested to study the application of NNs with deep architectures to other AEC tasks as well as other audio classification ones. In this way, the potential significance of deep architectures in machine learning, especially in audio information retrieval, will be put under further investigation.

REFERENCES

- [1] M. Abe and M. Nishiguchi. Content-Based Classification of Audio Signals Using Source and Structure Modelling, in *Proc. IEEE Pacific Conference on Multimedia*, pp. 280-283, 2000.
- [2] E. Alpaydin. Optical character recognition using artificial neural networks, *Artificial Neural Networks, 1989., First IEE International Conference on (Conf. Publ. No. 313)*, vol., no., pp.191,195, 16-18 Oct 1989.
- [3] D. Anderson, S. Ravindran, and M. Slaney. Low Power Audio Classification for Ubiquitous Sensor Networks, in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2004.
- [4] M. Arozullah, , S. H. Chin and A. Namphol. Image compression with a hierarchical neural network, *Aerospace and Electronic Systems, IEEE Transactions on*, vol.32, no.1, pp.326,338, Jan. 1996.
- [5] K. Asakawa, T. Kimoto, M. Takeoka and M. Yoda. Stock market prediction system with modular neural networks, *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, vol., no., pp.1,6 vol.1, 17-21 June 1990.
- [6] Y. Bengio. Learning Deep Architectures for AI, in *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 16, 2009.
- [7] S. Bengio, Y. Bengio, A. Courville, D. Erhan, P. Manzagol, and P. Vincent. Why Does Unsupervised Pretraining Help Deep Learning?, In *Journal of Machine Learning Research*, 11: 625-660, 2010.
- [8] S. Bengio, Y. Bengio, D. Erhan, P. Manzagol, and P. Vincent. The difficulty of training deep architectures and the effect of unsupervised pretraining, In *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 153–160, 2009.

- [9] Y. Bengio, J. Bergstra, A. Courville, D. Erhan and H. Larochelle. An empirical evaluation of deep architectures on problems with many factors of variation, In *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML)*, pp. 473–480, ACM, 2007.
- [10] Y. Bengio, L. Bottou, P. Haffner and Y. LeCun. Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [11] Y. Bengio, P. Lamblin, H. Larochelle and J. Louradour. Exploring strategies for training deep neural networks, *Journal of Machine Learning Research*, vol. 10, pp. 1–40, 2009.
- [12] Y. Bengio, H. Larochelle, P. Lamblin and D. Popovici. Greedy layer-wise training of deep networks, in *Advances in Neural Information Processing Systems 19*, 153, 2007
- [13] R. Bertolami, H. Bunke, S. Fernandez, A. Graves, M. Liwicki and J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, 2009.
- [14] L. Besacier, E. Castelli, D. Istrate, J. Serignat and M. Vacher. Life Sounds Extraction and Classification in Noisy Environment, in *Proc. IASTED International Conference on Signal & Image Processing*, 2003.
- [15] L. Besacier, E. Castelli, D. Istrate, J. Serignat and M. Vacher. Smart audio sensor for telemedicine, in *Proc. Smart Object Conference*, 2003.
- [16] T. Beth, I. Boesnach, M. Hahn, J. Moldenhauer and U. Spetzger. Analysis of Drill Sound in Spine Surgery, in *Proc. International Workshop on Medical Robotics, Navigation and Visualization*, pp. 11-12, 2004.
- [17] T. Blum, D. Keislar, E. Wold, and J. Wheaton. Content-Based Classification, Search, and Retrieval of Audio, in *Proc. IEEE Multimedia*, Vol. 3, no. 3, pp. 27-36, 1996.
- [18] B. Boser, , J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel and Y. LeCun. Backpropagation applied to handwritten zip code recognition, *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.
- [19] Y. Boureau, Y. LeCun and M. A. Ranzato: Sparse feature learning for deep belief networks, In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1185-1192, 2007.

- [20] T. Chen, J. Huang, Z. Liu and Y. Wang. Audio feature extraction and analysis for scene classification, In *Multimedia Signal Processing, 1997., IEEE First Workshop on* , vol., no., pp.343,348, 23-25 June 1997.
- [21] S. Chopra, Y. LeCun, C. Poultney and M. Ranzato. Efficient learning of sparse representations with an energy-based model, in *Advances in Neural Information Processing Systems 19 (NIPS)*, pp. 1137–1144, 2007.
- [22] P. Cook and G. Tzanetakis. Musical genre classification of audio signals, In *Speech and Audio Processing, IEEE Transactions on* , vol.10, no.5, pp.293,302, July 2002.
- [23] M. Cowling. Non-Speech Environmental Sound Classification System for Autonomous Surveillance, PhD Thesis, Griffith University, Australia, 2004.
- [24] M. Cowling and R. Sitte. Analysis of speech Recognition Techniques for use in a Non-Speech Sound Recognition System, In *Proc. International Symposium on Digital Signal Processing for Communication Systems*, 2002.
- [25] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions* 28(4):357–366, 1980.
- [26] A. Divakaran, K. Goh, K. Miyahara, R. Radhakrishnan and Z. Xiong. Audio-Visual Event Detection Based on Mining of Semantic Audio-Visual Labels, In *Proc. SPIE Conference on Storage and Retrieval for Multimedia Databases*, pp. 292-299, 2004.
- [27] A. Divakaran, T. Huang, R. Radhakrishnan and Z. Xiong. Audio Events Detection Based Highlights Extraction from Baseball, Golf and Soccer Games in a Unified Framework, in *Proc. IEEE International Conference on Multimedia and Expo*, pp. 401-404,2003.
- [28] R. D. Dony and S. Haykin. Neural network approaches to image compression, In *Proceedings of the IEEE* , vol.83, no.2, pp.288,303, February 1995.
- [29] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2012.
- [30] R. B. Dunn, D. A. Reynolds and T. F. Quatieri. Speaker Verification Using Adapted Gaussian Mixture Models, *Digital Signal Processing*, Volume 10, Issues 1–3, , Pages 19-41. January 2000.

- [31] M. Dziubinski, B. Kostek and P. Zwan. Statistical Analysis of Musical Sound Features Derived from Wavelet Representation, In *Audio Engineering Society*, 2002.
- [32] D. Ellis. Detecting Alarm Sounds, in *Proc. Workshop on Consistent and Reliable Acoustic Cues*, 2001.
- [33] D. Ellis and L. Kennedy. Laughter Detection in Meetings, NIST Meeting Recognition Workshop, in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004.
- [34] D. Ellis and K. Lee. Minimal-Impact Audio-Based Personal Archives, In *Proc. ACM workshop on Continuous Archiving and Recording of Personal Experiences*, 2004.
- [35] A. Eronen, S. Fagerlund, J. Huopaniemi, A. Klapuri, G. Lorho, V. Peltonen, T. Sorsa and J. Tuomi. Audio-Based Context Recognition, In *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 14, no. 1, 2006.
- [36] A. Eronen, T. Heittola, A. Mesaros and T. Virtanen. Context-dependent sound event detection, *EURASIP Journal on Audio, Speech, and Music Processing* 2013, 2013:1.
- [37] I. Fodor. A Survey of Dimension Reduction Techniques, *US DOE Office of Scientific and Technical Information*, (UCRL-ID-148494) 2002.
- [38] B. Fulkerson and A. Vedaldi. Vlfeat: an open and portable library of computer vision algorithms, In *MM '10 Proceedings of the international conference on Multimedia* Pages 1469-1472 ACM New York, NY, USA, 2010
- [39] A. Graves, G. Hinton and A. Mohamed. Speech Recognition with Deep Recurrent Neural Networks In *IEEE International Conference on Acoustic Speech and Signal Processing (ICASSP 2013)* Vancouver, 2013.
- [40] R. Grosse, H. Lee, A. Y. Ng and R. Ranganath. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 609-616. ACM, 2009.
- [41] G. Guo, and Z. Li. Content-based Audio Classification and Retrieval using Support Vector Machines, In *IEEE Transactions on Neural Networks*, Vol. 14, pp. 209-215, January, 2003.

- [42] T. Hanazawa, G. Hinton, K. Lang, K. Shikano and A. Waibel. Phoneme Recognition Using Time-Delay Neural Networks, In *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 3, March 1989.
- [43] S. Haykin *Neural Networks. A Comprehensive Foundation*, Prentice Hall, 1994.
- [44] G. E. Hinton. Learning multiple layers of representation, In *Trends in Cognitive Sciences*, Vol. 11, Issue 10, pp 428-434, 2007.
- [45] G. E. Hinton, S. Osindero and Y. Teh. A fast learning algorithm for deep belief nets, *Neural Computation*, vol. 18, pp. 1527–1554, 2006.
- [46] G. E. Hinton, D. E. Rumelhart and R. J. Williams. (8 October 1986). Learning representations by back-propagating errors. *Nature* 323.9: 533–536, 1986.
- [47] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.
- [48] D. Hoiem, Y. Ke, and R. Sukthankar. SOLAR: Sound Object Localization and Retrieval in Complex Audio Environments, In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005.
- [49] A. Härmä. Automatic recognition of bird species based on sinusoidal modeling of syllables, In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.
- [50] D. Istrate, J. Serignat and M. Vacher. Sound Detection and Classification through Transient Models using Wavelet Coefficient Trees, in *Proc. European Signal Processing Conference*, pp. 1171-1174, 2004.
- [51] C. Jianfeng, Z. Jianmin, A. Kam and L. Shue. An automatic acoustic bathroom monitoring system, In *Proc. IEEE International Symposium on Circuits and Systems*, 2005.
- [52] H. Jiang, L. Lu and H. Zhang. Content Analysis for Audio Classification and Segmentation, *IEEE Transactions on Speech and Audio Processing*, Vol. 10, no. 7, pp. 504-516, 2002.
- [53] B. H. Juang and L. R. Rabiner. *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series. PTR Prentice Hall, 1993.

- [54] H. Kim, N. Moreau and T. Sikora. Audio Classification Based on MPEG-7 Spectral Basis Representations, *IEEE Transactions on Circuits and Systems for Video Technology*, Special Issue on Audio and Video Analysis for Multimedia Interactive services, Vol. 14, no. 5, pp. 716-725, 2004.
- [55] O. Lartillot and P. Toiviainen. A Matlab Toolbox for Musical Feature Extraction From Audio, In *International Conference on Digital Audio Effects, Bordeaux*, 2007.
- [56] B. Logan. Mel frequency Cepstral Coefficients for Music Modelling, In *ISMIR*, 2000.
- [57] B. Lukic. *Activity Detection in Public Spaces*, M.Sc. Thesis, Royal Institute of Technology, Stockholm, Sweden, 2004.
- [58] P. Lukowicz, N. Perera, T. Starner, M. Stäger, G. Tröster, and T. von Büren. Sound Button: Design of a Low Power Wearable Audio Classification System, In *Proc. IEEE International Symposium on Wearable Computers*, pp. 12-17, 2003.
- [59] P. Lukowicz, M. Stäger, and G. Tröster. Implementation and Evaluation of a Low-Power Sound-Based User Activity Recognition System, in *Proc. IEEE International Symposium on Wearable Computers*, pp. 138-141, 2004.
- [60] L. Ma, B. Milner and D. Smith. Context awareness using environmental noise classification, in *Proc. European Speech Processing Conference*, pp. 2237-2240, 2003.
- [61] L. Ma, N. Ryan and D. Smith. Acoustic environment as an indicator of social and physical context, *Personal and Ubiquitous Computing*, 2005.
- [62] D. Macho, R. Malkin, C. Nadeu, M. Omologo, A. Temko and C. Zieger. CLEAR Evaluation of Acoustic Event Detection and Classification Systems, In *Multimodal Technologies for Perception of Humans* (pp. 311-322). Springer Berlin Heidelberg.
- [63] E. Monte, C. Nadeu and A. Temko. Comparison of Sequence Discriminant Support Vector Machines for Acoustic Event Classification, In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on* , vol.5, no., pp.V,V, 14-19 May 2006.
- [64] P. Nordqvist. *Sound Classification in Hearing Instruments*, PhD Thesis, Royal Institute of Technology, Stockholm, Sweden, 2004.
- [65] M. Omologo and C. Zieger. Acoustic event classification using a distributed microphone network with a GMM/SVM combined algorithm, *INTERSPEECH*, 2008.

- [66] R. B. Palm. Prediction as a candidate for learning deep hierarchical models of data, Master's Thesis, 2012, Technical University of Denmark.
- [67] J. C. Platt, D. Simard and P. Y. Steinkraus. Best practices for convolutional neural networks, In *International Conference on Document Analysis and Recognition (ICDAR)*, p. 958, 2003.
- [68] M. Slaney. Mixtures of Probability Experts for Audio Retrieval and Indexing, In *Proc. IEEE International Conference on Multimedia and Expo*, 2002.
- [69] M. Slaney. Semantic–Audio Retrieval, In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002.
- [70] J. Tebelskis. Speech recognition using neural networks. PhD dissertation, Carnegie Mellon University, 1995.
- [71] P. J. Werbos. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, 1974.
- [72] H. White. Economic prediction using neural networks: the case of IBM daily stock returns, *Neural Networks*, 1988., *IEEE International Conference on* , vol., no., pp.451,458 vol.2, 24-27 July 1988.