HANDWRITTEN NUMERAL RECOGNITION USING NEURAL NETWORKS

MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DEGREE OF

Master of Technology

In

Information Systems

Submitted By:

SUVARNA KOTHARI

(2k13/ISY/25)

Under the Guidance

Of

Dr. O. P. Verma

(Prof. and Head, Department of CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(2013-2015)

CERTIFICATE

This is to certify that **Suvarna Kothari (2k13/ISY/25)** has carried out the major project titled "**Handwritten Numeral Recognition Using Neural Networks**" in partial fulfilment of the requirements for the award of Master of Technology degree in Information Systems by **Delhi Technological University**.

The major project is bonafide piece of work carried out and completed under my supervision and guidance during the academic session 2013-2015. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any degree or diploma.

Dr. O. P. Verma

Professor and Head

Department of Computer Science and Engineering

Delhi Technological University

Delhi-110042

ACKNOWLEDGEMENT

I take the opportunity to express my sincere gratitude to my project mentor Dr. O. P. Verma, Prof. and Head of Department, Department of Computer Science and Engineering, Delhi Technological University, Delhi, for providing valuable guidance and constant encouragement throughout the project. It is my pleasure to record my sincere thanks to him for his constructive criticism and insight without which the project would not have shaped as it has.

It humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Suvarna Kothari

Roll No. 2k13/ISY/25

M.Tech (Information Systems)

E-mail: suvarnakothari91@gmail.com

ABSTRACT

Machine learning algorithms including Support Vector Machines (SVM's), Multilayer Perceptrons and Neural Networks have been successful in vision tasks earlier, but it has been seen that there is stagnation in the error rate or accuracy of these algorithms due to reasons including poor generalization, local minima and weight change. The challenge to improve further still remains. Deep learning has posed several interesting possibilities and state of the art results have been achieved in vision tasks including image labelling, hand written digit recognition etc. Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Deep architectures attempt to learn hierarchical structures and seem promising in learning simple concepts first and then successfully building up more complex concepts by composing the simpler ones together

Many existing learning methodologies have been used in recognition of hindi numerals but none of the deep learning approaches have been tried much in this area as of now. By applying deep learning methods, we are free of hand-crafted low-level features and can automatically learn mid-level and higher-level features from a large amount of unlabelled raw samples beyond types and domains of handwriting recognition also.

Through this work we look at various deep learning methodologies and specifically methodologies which help in pre-training of the multiclass classifiers thus avoiding the need of hand crafting features for better results. The idea of deep learning has been implemented using different kinds of neural networks though the concept is not restricted to only neural networks. State-of-the-art results have been achieved on classification experiments performed on hindi numeral images involving techniques of pre-training a deep multiclass classifier basically Autoencoders.

Table of Contents

Figures and Tables

Chapter 1

INTRODUCTION

Automatic handwriting recognition and machine learning are two very strongly linked research fields as algorithms from machine learning research area can be applied for solving complex problems of handwriting recognition. Machine learning on the other hand can take advantage of the large database for handwriting recognition available thus helping to move machine learning towards reality.

Essentially handwriting recognition can be seen as a classification task in which the input data have to be classified in different classes (i.e 0-9 in case of numerals and a-z in case of alphabets).Thereby, the best classifier to solve this task can be machine learning algorithms. Deep Neural Network provide not only as a classifier but also a form to extract features in an automated manner and give better classification accuracy.

The two main topics of this Master Thesis will be introduced in this chapter: handwriting recognition and deep neural network. After that, the motivation and goals of the Master Thesis will be presented and an outline of the dissertation will be included at the end of this section.

1.1 HANDWRITING RECOGNITION

Handwriting Recognition is the technique by which symbols or characters written by hand can be recognized by a computer system. The sources of input can be photographs, paper documents, touchscreens or other devices. The image can either be sensed “offline” by optically scanning a piece of paper or can be sensed “online” through the movement of pen tip or stylus.

There can be two basic categories of handwriting recognition-

- Numeral Recognition- Is the process in which we try to recognize digits i.e. 0-9 and try to classify them correctly.
- Character Recognition- Is the process in which we try to recognize alphabets i.e. a-z and try to classify them correctly.

Since there are many languages in the world, we have limited the scope of this Master Thesis to the offline recognition of Hindi Numerals in devanagari script.

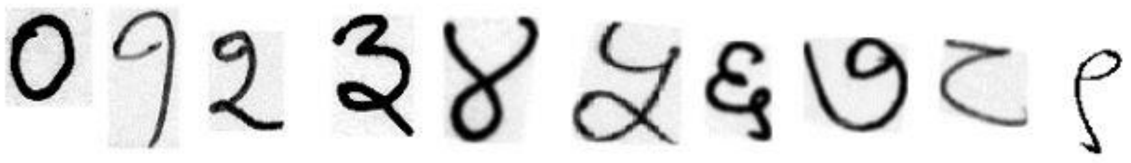


Fig 1.1 showing the different hindi numerals

The structure of a handwriting recognition system can be divided in the following main parts.

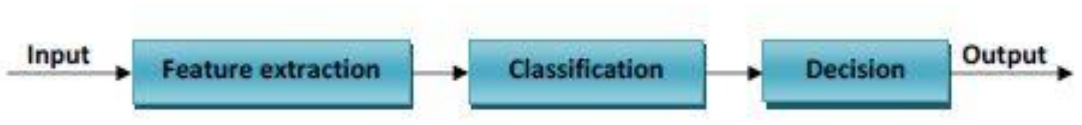


Fig 1.2 showing the structure of a handwriting recognition system

- Feature Extraction – in this part, relevant information is extracted from the input signal for classification purposes. Thus, a new depiction of input signal is obtained called feature.
- Classification – in this phase a score is given based on confidence or distance measure to fulfil the classification task.
- Decision – given the score obtained in previous part, the system decides depending upon the task at hand.

1.2 DEEP NEURAL NETWORKS

A relatively new field of machine learning that has been in news recently for gaining success is Deep Neural Network (DNN henceforth). A range of structures are included under this name which have these things in common: the starting point for these networks is the classical neural network with only one hidden layer, more hidden layers are added to it so that it can solve more complex problems which the traditional neural networks could not solve.

Hence, human learning technique is copied by DNN as the features are directly extracted from the input data without dependence on human crafted features. Features are extracted from multiple levels of abstraction and complex learning functions are absorbed directly from data.

In the ever growing world of data and increasing usage of machine learning, the automated learning of features is a much sought after thing.

Due to limitations in training of this kind of structure, experimental success was not reported till 2006. However, the limitations were mostly from hardware point of view rather than theoretical. Although many algorithms existed for training of such structures, poor results were found when they were randomly initialized. All these limitations disappeared with the upgradation of hardware devices and the famous learning algorithm given by Hinton in 2006 in which he used the advantage of unsupervised learning algorithms to initialize parameters and followed a greedy approach of training one layer at a time.

Some of the drawbacks of this kind of neural network that still exist are the huge number of free parameters, high computation cost for training of such structures and amount of data required to train such networks.

1.3 MOTIVATION

The good results obtained in the field of handwritten numeral and character recognition is the main motivation behind the idea of combining handwritten recognition with DNN. On the basis of the results encountered, we can conclude that deep architectures have the capability to learn the correct approach for handwriting recognition that can be used for other pattern recognition tasks as well.

At one hand, better performances have been found by researchers on systems using DNN than other approaches. It is seen that not only in the field of handwriting recognition but also computer vision tasks and speech learning. All these examples prove that DNN is indeed a powerful machine learning tool.

On the other hand, numeral recognition problem has almost similar issues as character recognition, so the solution through DNN can also be applied to other similar fields. Moreover, other research areas like Biometric Recognition System (iris, signature, face, fingertips) can also be benefitted by the study in this Master Thesis.

1.4 GOAL OF MASTER THESIS

The main objective of this Master Thesis is to apply learning algorithm of deep neural network to achieve better accuracy for classification of hindi numerals.

Thus, the thesis can be divided into two parts –

- Theoretical Framework – This includes the study of different frameworks of deep neural networks and their attributes for deciding which one would suit the problem best.
- Experimental Framework – The objective of this part of the thesis is to develop a numeral recognition system based on DNN on MATLAB platform.

1.5 OUTLINE OF DISSERTATION

The dissertation is structured as follows –

- Chapter 1 introduces the issues of handwriting recognition and deep neural network and gives the motivation, objective and outline of the Master Thesis.
- Chapter 2 summarizes the current happenings in handwriting recognition and deep neural network, the main issues discussed in dissertation.
- Chapter 3 defines the proposed method of handwriting recognition by using deep neural networks.
- Chapter 4 describes the experiments performed during this effort, listing and examining the results achieved.
- Chapter 5 summarizes the main inferences drawn from this work, also outlining future research areas.

Chapter 2

LITERATURE REVIEW

Handwriting Recognition and Machine Learning based on deep neural network are the two areas in which this work is mostly focussed. These both have been very active areas of research since the past 30 years. A brief overview of both fields will be provided in this chapter.

In section 2.1, the importance of handwriting recognition for researchers will be discussed and two main branches of handwriting and two main approaches: template and feature based; will be further evaluated.

In section 2.2, will show the rise of deep neural network in the machine learning field, and some machine learning algorithms as well as architectures will be discussed.

2.1 AUTOMATIC HANDWRITING RECOGNITION

Among all topics of automatic handwriting recognition, this section evaluates some of the best methods by which this can be performed and also discuss the advantages of automatic handwriting recognition.

2.1.1 Why Handwriting Recognition?

As we mentioned above in chapter 1, there are two approaches to handwriting recognition (off-line and online).We are dealing with the off-line approach in this Mater Thesis. Thus, some of the applications of offline handwriting recognition are-

- Cheque Reading- Cheque reading in banks can be one of the most prominent applications of off-line handwriting recognition. It is also one of the most commercially viable applications of Handwriting Recognition.
- Postcode Recognition- Handwritten postal code on letters can be read by Handwriting Recognition software hence making the process faster and more efficient.
- Form processing- Handwriting Recognition can also be applied to form processing. Usually public information is gathered via forms.

- Signature Verification- Signature Verification can also be performed by Handwriting Recognition and can also be used to identify a person as handwriting differs from person to person.

The two main approaches for handwriting recognition are

- Template matching- It is the oldest and simplest method available for handwriting recognition. In this method, we have a pre-fetched reference text with which we match the input text and the text with less error is the required output. For the standard fonts, this method gives quite good results but for handwriting recognition

- Feature Extraction- In this method orthogonal properties are extracted from the image and statistical distribution is used to analyse it. Most features are extracted by performing basic and logic operations on the image. The features extracted help us in gathering clear details about the character in the image. The features are usually functions of physical properties such as relative position, length to width ratio, number of end points, number of joints etc. This method gave good results for handwriting recognition but was more prone to errors such as thickness of edges and noise.

2.2 DEEP NEURAL NETWORK

This section describes the emergence of deep neural networks as the new paradigm in the field of machine learning. It is responsible for the recent breakthrough in machine learning causing revised interest in the field. In this chapter, we will discuss the differences between supervised and unsupervised learning algorithm, basic dissimilarities between shallow and deep architectures and gather an idea on training and testing of deep architectures.

2.2.1 A new paradigm for machine learning

Artificial intelligence has always aimed at trying to model the real world [1]. It essentially implies processing huge amounts of data, generalizing new contexts and being able to answer questions put up to it. Although there have been many cases of success some drawbacks still remain as computers cannot understand real scenes well enough and are unable to express them in natural languages [1].

The basic difference between learning algorithms and human brain is the way in which useful information is extracted from data. Different levels of abstraction are used to gather information from data for feature extraction gradually in case on human brain. Hence we can say that a bigger problem is decomposed into a smaller less complex problem, gathering different levels of representation in the process.

Imitation of this behaviour is what machine learning algorithms are aimed at. Low level features are captured first by the system that are invariant to small variations. The invariance of these features is then increased by transforming them and useful information is extracted from these. This means generalization of data can be done by the occurrence of frequent patterns.

For doing all this, a structure is required that has the ability to transform the input in a non-linear way. This is required by the learning algorithm for the application of mathematical functions and transformations that are varying and non-linear.

Through the years machine learning algorithms did not have capable enough structures to apply complex functions that were necessary to solve problems. Classical neural networks were built with only one hidden layer. This means that there can be only one non-linear transformation of data. This was overcome by deep architectures as they had more hidden layers stacked on top of each other hence more transformations of data were allowed in this architecture. But this has its own drawbacks too, i.e. local minima can be found more easily as the function becomes complex.

Earlier learning algorithms were ill-equipped with handling of such “deep” architectures as they had more than one hidden layer in them. Convergence of weights and other parameters into very small value, almost close to zero while performing backpropagation and gradient descent on these architectures were the main reasons for bad results.

Convolution neural network was an exception in this case. Yann Le Cunn’s LeNet yielded the first successful results for such a deep architecture [8]. Due to the sharabilty of parameters between different parts of the network, the amount of free parameters were reduced greatly in this structure. But the main drawback for this network was that it performed supervised learning, i.e. a huge amount of labelled data was required for training, which is difficult to get.

In 2006, Hinton was the first person to achieve success in the unsupervised learning of deep neural network and called it the Deep Belief Network. This network took advantage of the unsupervised structure and trained one hidden layer at a time which are called Restricted Boltzmann Machines. This is suitable for problems having unlabelled data, which is also the advantage of unsupervised learning.

The drawback of increasing the complexity of the network is that the number of free parameters rises. This means more time is taken for its estimation or learning. Also more computation capacity and more data storage is also required. Since there was a limitation of hardware resources before 2006, the algorithms though theoretically correct could not be tested. Although there are no hardware limitations now but the computation cost of DNN still presents some hindrances.

Some open ended challenges that still remain for DNN include how to choose the structure and parameters of a network to solve a particular problem or how much of a knowledge about a problem must the network have. As theoretical results have proven, there is no specified depth of a network, i.e. it is problem related [4]. These issues being open ended challenges can be considered as drawbacks of DNN but can also be seen as future areas of research.

In conclusion, we can say that given all advantages of DNN and the huge scope of application it has along with their unresolved issues, DNN can be considered as a new paradigm of machine learning which requires further explorations.

2.2.2 Supervised and Unsupervised Learning

A learning algorithm can be seen as a method to predict the output from input. Thus prediction can be seen as a function that is used to match the inputs with the outputs. Learning is essentially a concept closely related to generalization. For a classification problem, it can be seen as the ability to correctly classify new data that differ from the data supplied during training. Therefore, datasets used for machine learning are split in generally two parts:

- Training Data – This is the set that covers samples used for estimating the parameters of the objective function which should be selected such as the error should be minimalized for misclassification.
- Testing Data- This is the data by which we prove whether the parameters selected from among the training data can actually classify correctly. Since these data were not a part of training data, parameter selection is not hampered.

Learning algorithms as we said above can be classified in two major categories depending on the availability of labelled or unlabelled data. A third type of learning called reinforcement learning also exists which is a form of semi-supervised learning but we will be dealing with the above two only.

- Supervised Learning-

  One of the type of machine learning problem is of classification, i.e. given an input, we have to classify it into a category or class of input. If labelled data is used for training the network, it is called supervised learning. In this type of learning, the difference between predicted output and actual output is fond out called the error function. This is minimized so that the error is reduced thereby minimizing the cost function.

  Deep convolutional network, Logistic regression and Multi-layer perceptrons can be highlighted as falling under this kind of learning algorithms.
- Unsupervised Learning-

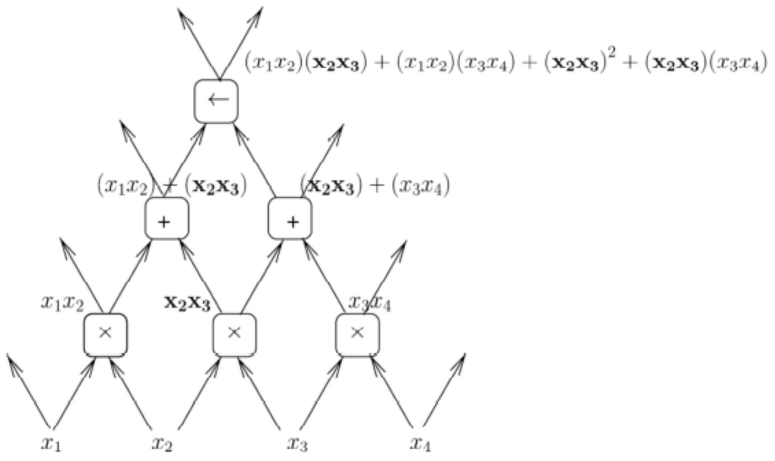
  The main approach followed by this type of algorithms is clustering. In clustering, similar data is combined to form a group based on some similarity measure among the data. Since no labelled data is available for training of the data, no error can be found out between the target data and the actual output. Generally, the cost function is minimized based on reduction of a reinforcement error which can be considered as a parameter for training the algorithm.

  Autoencoders and Restricted Boltzmann Machine are a part of this set.

Aside from this, Hinton et. al in 2006[8] gave a semi-supervised learning algorithm in which building blocks of RBM were used for initial training but later a supervised learning algorithm was used to fine tune the network.

2.3 FROM SHALLOW TO DEEP ARCHITECTURES

It was proven theoretically beforehand that the limitations of shallow architectures would be overcome by the deep ones but due to hardware limitations, implementation of deep structures was not feasible.



$(x_1x_2)(\mathbf{x_2x_3}) + (x_1x_2)(x_3x_4) + (\mathbf{x_2x_3})^2 + (\mathbf{x_2x_3})(x_3x_4)$
$(x_1x_2) + (\mathbf{x_2x_3})$
$(\mathbf{x_2x_3}) + (x_3x_4)$
x_1x_2
$\mathbf{x_2x_3}$
x_3x_4
x_1
x_2
x_3
x_4

Fig 2.1 Expression computed by a deep architecture with sums and products [1]

The inefficient representation of some functions is the major motivation for the shift towards deep learning. This simply means that the architectures are too shallow to be trained by such a few number of parameters. This mostly happens with compact functions. [3]. If the compact function were to be implemented by an architecture with a single hidden layer, there would be an exponential rise in the number of parameters used.

All this implies that if an architecture with single hidden layer is used, the training samples should be large in number with respect to the number of parameters, otherwise training would not be as efficient as would be in deep architectures.

As seen in figure 2.1, we can see the multiple occurrence of x2x3 and how a deep network simplifies this calculation. Since it is a deep network, only 12 connections are visible, but in the similar case of fig 2.2 more than 20 connections would have been used for this.

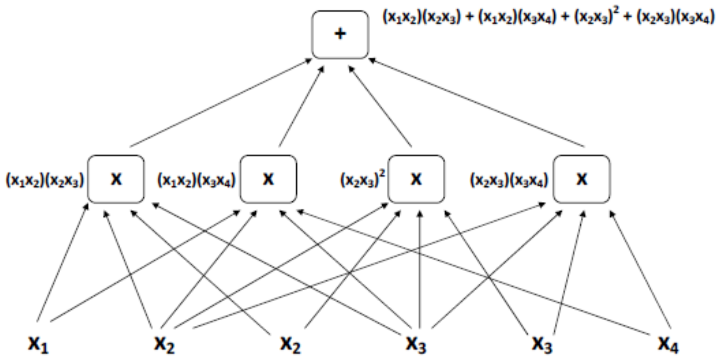


Fig 2.2 Same expression from Fig 2.1 computed by a shallow architecture with sums and products.

Inspite of all the advantages being presented by the deep architecture, if the problem is not complex enough, i.e. a simpler structure can be fitted on it, deep architectures are not always required for solving a problem, they can make the solution more complex.

Thus, we can say that deep structures are formed using shallow structures a building blocks. For eg if a supervised learning based classification task is to be performed, the structure similar to fig 2.3 can be used which is composed of the following –

- Input layer – It defines the inputs used to feed the network.
- Hidden layer – sigmoid and tanh functions are used to apply non-linear transformation on the input data.

$$sigmoid(a) = \frac{1}{1+e^{-a}}$$

$$tanh(a) = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}}$$

  The number of hidden layers depends on the problem where the output of one layer will serve as the input for the next layer.
- Logistic Regression Classifier – classification is performed by this generally by maximizing the cost function and generally forms the output layer.

$$P(Y=i|x,W,b) = softmax_i(Wx+b) = \frac{e^{W_i x+b_i}}{\sum_j e^{W_j x+b_j}} \tag{2.1}$$

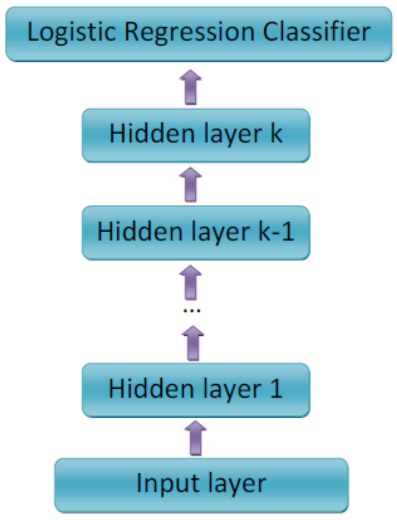


Fig 2.3 Example of deep architecture for classification.

Where given an input, the selected class would be :

$$y = argmax_i P(Y = i|x, W, b) \tag{2.2}$$

By using a gradient descent algorithm [Bishop,2007] , the parameters that need to be tuned are the bias vector and the weight matrix.

2.4 LEARNING METHODS FOR DEEP LEARNING

2.4.1 Convolutional Neural Network

CNNs are a family of multi-layer neural networks particularly designed for use on two-dimensional data, such as images and videos. The idea of weight sharing in CNNs leads to less computations and sparse encodings resulting in better training of lower layers as well. CNNs are the first truly successful deep learning approach where many layers of a hierarchy are successfully trained in a robust manner, here robust means state of the art error rate on vision tasks and better generalization. CNNs were proposed as a deep learning framework that is motivated by minimal data preprocessing requirements.

Figure 1.5 illustrates the convolution process consisting of convolving an input (image in this case) with filter fx then adding a trainable bias bx to produce the convolution layer Cx. The filters are trainable. The subsampling (max pooling) consists of summing a neighborhood (four pixels), weighting by scalar wx+1, adding trainable bias bx+1, and passing through a sigmoid function to produce a roughly 2x smaller feature map Sx+1 which is then fed to a multilayer neural network for gradient based optimization. Convolutional training acts as a feature extractor. Features extracted can then be fed to a supervised learning algorithm such as a multilayer neural network, softmax classifier etc.

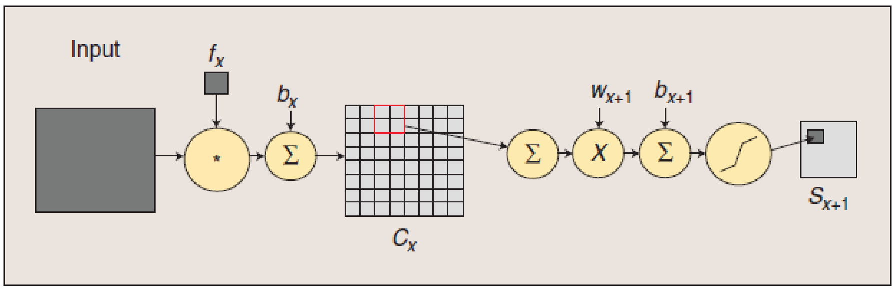


Fig 2.4 The convolutional and sub sampling process [7]

2.4.2 Deep Belief Network

DBNs are probabilistic generative models that stand in contrast to the discriminative nature of traditional neural nets which are mainly used for classification tasks. Generative models provide a joint probability distribution over observable data and labels, facilitating the estimation of both $P(Observation\,|\,Label)$ and $P(Label\,|\,Observation)$, whereas discriminative models like neural networks only deal with $P(Label\,|\,Observation)$. Below are the problems which DBNs address which are faced when traditional neural networks are trained with back propagation:

- Availability of substantial labelled data set for training
- Large convergence times due to poor training of lower layers.
- Inadequate parameter selection techniques leading to poor local optima.

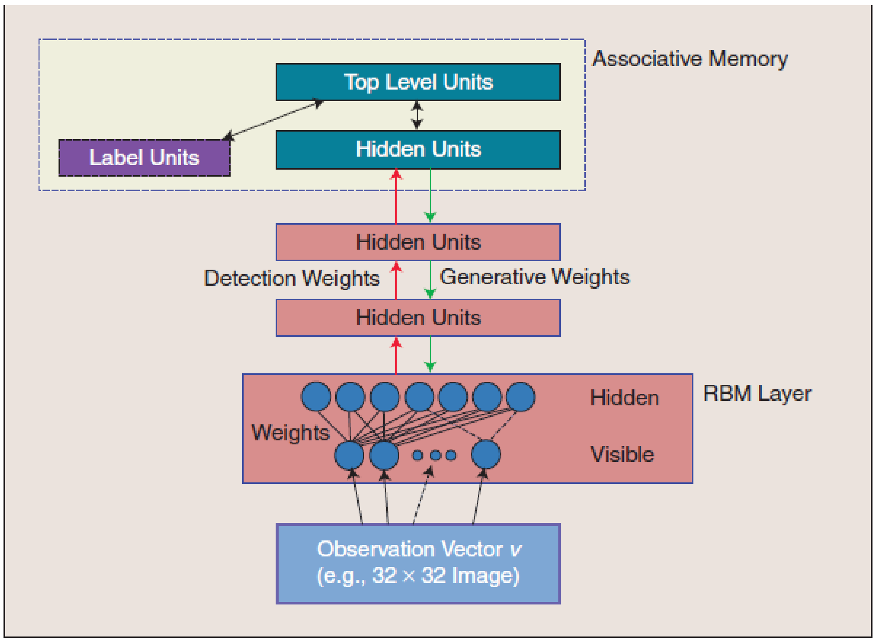


Fig 2.5 Illustration of layers in a Deep Belief Network [7]

The hidden units are trained to capture higher-order data correlations that are observed at the visible units. Initially, aside from the top two layers, which form an associative memory, the layers of a DBN are connected only by directed top-down generative weights. A DBN may be fine-tuned after initial training for better discriminative performance by utilizing labelled data through back-propagation. It has been proved by [9] that DBNs perform better than traditional neural networks since the only task at hand is the local search for better weights as compared to convergence times, and speedy training.

2.4.3 Autoencoder

Autoencoders or Auto-Associators or Diabolo networks are neural networks which employ unsupervised learning to gather interesting features inherent in the data. They have a single hidden layer with input and output layers being the same i.e., they aim at reconstructing the input when the reconstruction error is minimized using a suitable optimization technique such as gradient descent with back-propagation. The identity function seems a particularly trivial function to be trying to learn; but by placing constraints on the network, such as by limiting the number of hidden units, interesting structure about the data can be discovered. Figure 1.7 depicts an autoencoder framework with input, hidden and output layers. The type depicted is an under-complete autoencoder where the number of nodes in the hidden layer is less than the length of input feature vector.

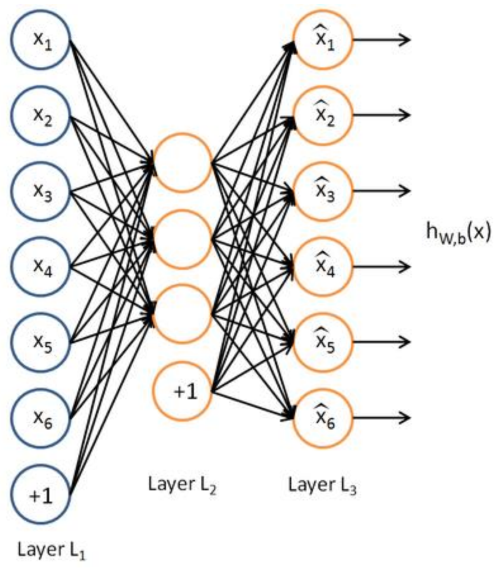


Fig 2.6 Autoencoder Framework [10]

Every autoencoder type has two parts:

Encoder: The deterministic mapping $f_\theta(x)$ (from layer L_1 to L_2) that transforms an input vector x into hidden representation y is called the encoder. Its typical form is an affine mapping followed by a sigmoidal nonlinearity.

Decoder: The resulting hidden representation y is then mapped back to a reconstructed d dimensional vector z in input space, $z = g_{\theta'}(x)$. This mapping $g_{\theta'}(x)$ (from layer L_2 to L_3) is called the decoder. Its typical form is again an affine mapping optionally followed by a squashing non-linearity.

In general z is not to be interpreted as an exact reconstruction of x, but rather in probabilistic terms as the parameters (typically the mean) of a distribution $p(X|Z = z)$ that may generate x with high probability. There are types of autoencoders depending upon on factors such as number of nodes in hidden layer, the reconstruction error which is used, sparsity parameters and noisy inputs.

Autoencoders can be stacked together in a greedy layer-wise fashion for pre-training (initializing) the weights of a deep network. Stacked Autoencoders consist of multiple layers of autoencoders in which the outputs of each layer are wired to the inputs of the successive layer and the activations of the last hidden unit is fed directly as input to a supervised criterion for training.

2.5 APPLICATIONS OF DEEP LEARNING

There have been several studies demonstrating the effectiveness of deep learning methods in a variety of application domains. In addition to the Mixed National Institute of Standards and Technology Database (MNIST) handwriting challenge, there are applications in face detection, speech recognition and detection, general object recognition, natural language processing, and robotics. Interest in deep machine learning has not been limited to academic research, the Defense Advanced Research Projects Agency (DARPA) has announced a research program exclusively focused on deep learning.

Approach	**Un-Supervised Pre-training Required?**	**Generative v/s Discriminative**	**Comments**
Convolutional Neural Networks	No	Discriminative	Introduces a notion of spatial invariance and reduces the cost of computations
Deep Belief Nets	Helpful	Generative	Multi-layered recurrent neural network trained with energy minimizing methods

Stacked Auto Encoders	Helpful	Discriminative (denoising encoder maps to generative model)	Stacked neural networks that learn compressed encodings through reconstruction error
Hierarchical Temporal Memory	No	Generative	Hierarchy of alternating spatial recognition and temporal inference layers with supervised learning method at top layer

Table 2.1 Summary of mainstream deep machine learning approaches [7]

Chapter 3

DEEP NEURAL NETWORK APPLIED TO HINDI NUMERAL RECOGNITION

There can be many applications of deep neural networks as a machine learning tool. The two major applications are in handwritten recognition and object recognition. In this chapter we discuss the methodology that has been used to counter the problem statement. There is no work that has been done on hindi numerals using the methodologies of deep and self-taught learning. In this chapter we describe the steps that we use to successfully classify a hindi numeral image.

3.1 STACKED AUTOENCODERS

We have come across autoencoders in chapter 2. In this chapter we look at another way they can be used for the same purpose. Autoencoders we now know can act as dimensionality reduction agents similar to PCA, keeping in mind certain constraints, but besides dimensionality reduction they can learn interesting features from data for e.g. in case of images then can learn edges and gabor like features.

But the question that should be considered is, what if we need to learn more complex features than just edge strokes? This is where stacked autoencoders come into picture! Stacked autoencoders can build up on features which have been learnt by a previous autoencoder and can learn more complex features such as edges which are part of same object. The idea is simple and abides by the ideology of deep learning i.e. if we have a more complex problem to solve we need more layers of non-linearities that can efficiently factorize the problem. Below are steps explaining how stacked setting of autoencoders works taken from [10].

First, you would train a sparse autoencoder on the raw inputs $x^{(k)}$ to learn primary features $h^{(1)(k)}$ on the raw input as given in the figure 3.1

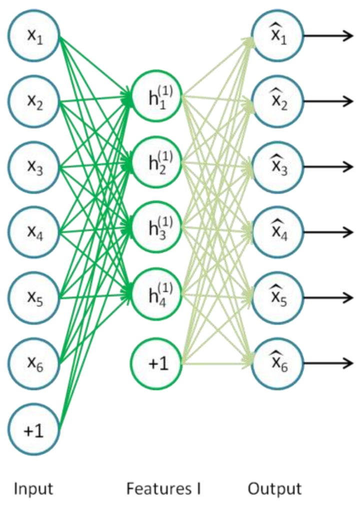


Fig 3.1: First autoencoder from the stacked series [10]

Next, we will feed the raw input into the above trained sparse autoencoder (figure 3.1), obtaining the primary feature activations $h^{(1)(k)}$ for each of the inputs $x^{(k)}$. These primary features will be used as the "raw input" to another sparse autoencoder to learn secondary features $h^{(2)(k)}$ on these primary features as show in figure 3.2

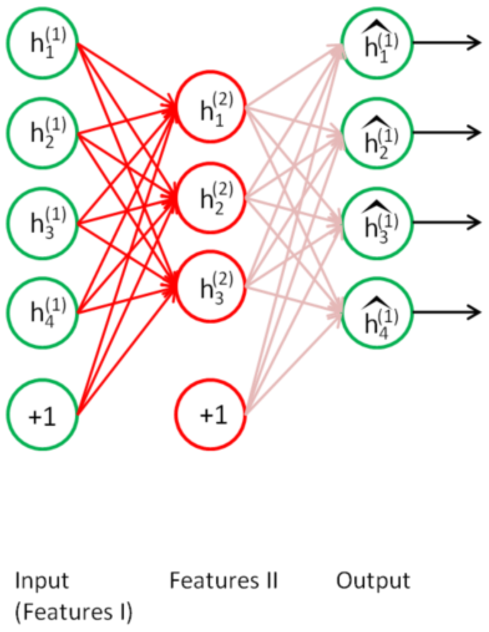


Fig 3.2: Second autoencoder of the stacked series which takes as input the features learnt by the first autoencoder [10]

The secondary feature activations $h^{(2)(k)}$ for each of the primary features $h^{(1)(k)}$ (which correspond to the primary features of the corresponding inputs $x^{(k)}$). These secondary features are fed as "raw input" to a classifier. The below figure 3.3 displays all the layers in action where the encoded input is fed to the classifier.

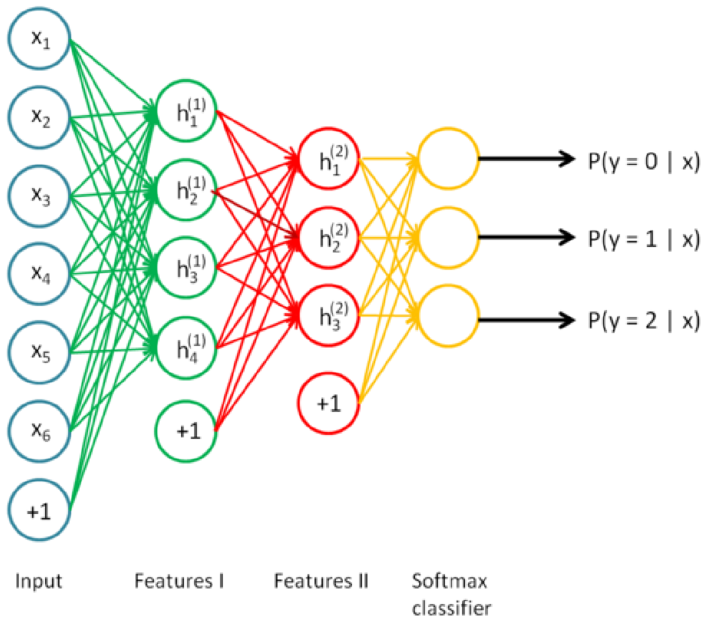


Figure 3.3: Deep learning setting with stacked autoencoders [10]

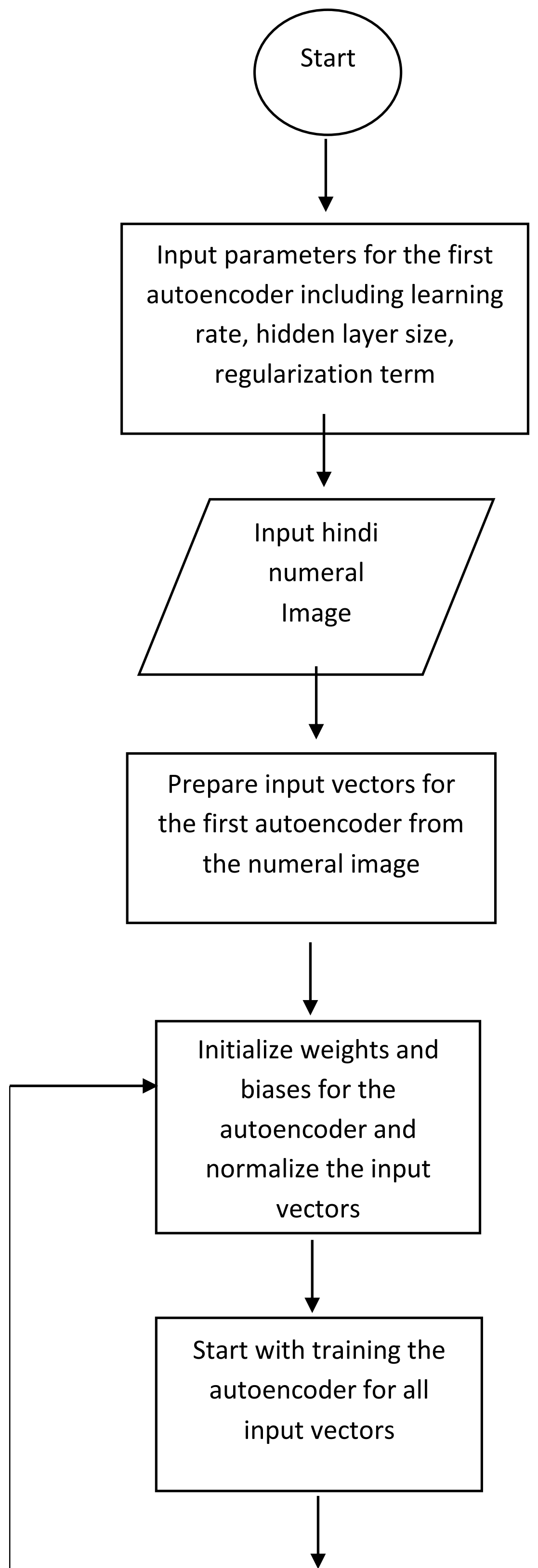
3.2 GREEDY LAYER-WISE TRAINING

In section 3.1 we saw how autoencoders can be stacked together to learn features which are complex. That being said, the questions to be answered are - how will the whole stack be trained, and what happens when we attach a supervise criterion to the last hidden layer? In the process, we have to train that classifier. In this section we discuss an approach which has been used in this research for training a deep network.

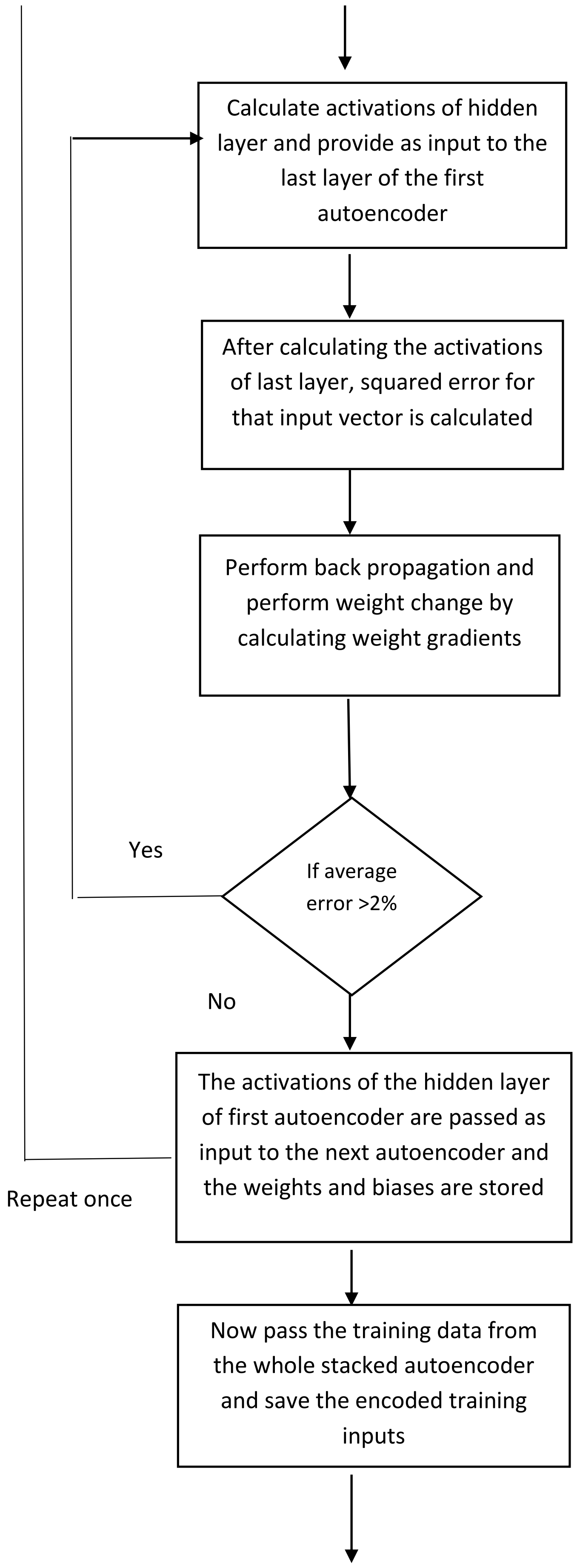
In this approach the parameters which include the weights and biases specific to an autoencoder are trained separately from other autoencoders which are part of the same stack. For example mentioned in section 3.1 the first autoencoder will be trained using gradient descent and backpropagation and after the training is finished after a specific criteria has been achieved we move on to the next autoencoder and train it the same way. This way parameters after being trained greedily layer wise of the entire stack are saved so that the training inputs can be encoded and fed as input to a supervise criterion for classification.

3.3 FLOW CHART OF THE MODEL

This flowchart represents the basic workflow of how autoencoders are being used to classify images containing handwritten hindi numerals.



Start
Input parameters for the first autoencoder including learning rate, hidden layer size, regularization term
Input hindi numeral Image
Prepare input vectors for the first autoencoder from the numeral image
Initialize weights and biases for the autoencoder and normalize the input vectors
Start with training the autoencoder for all input vectors



Calculate activations of hidden layer and provide as input to the last layer of the first autoencoder
After calculating the activations of last layer, squared error for that input vector is calculated
Perform back propagation and perform weight change by calculating weight gradients
If average error >2%
Yes
No
The activations of the hidden layer of first autoencoder are passed as input to the next autoencoder and the weights and biases are stored
Repeat once
Now pass the training data from the whole stacked autoencoder and save the encoded training inputs

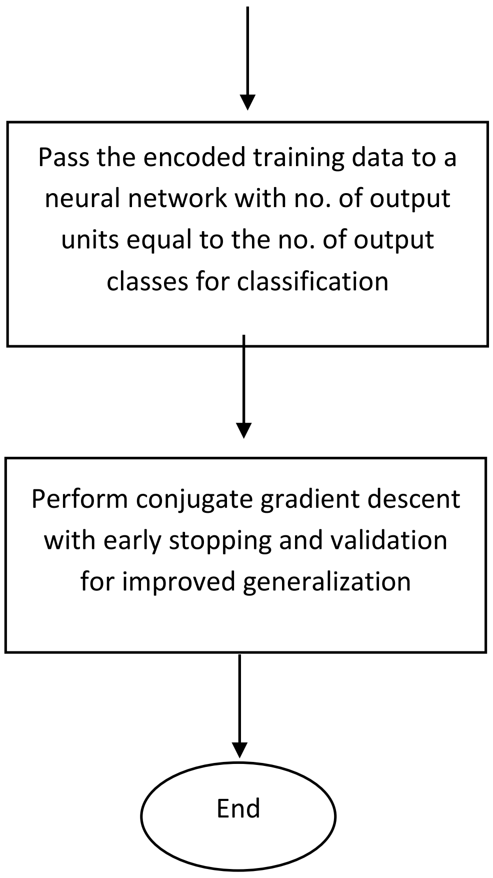

Pass the encoded training data to a neural network with no. of output units equal to the no. of output classes for classification
Perform conjugate gradient descent with early stopping and validation for improved generalization
End

Fig 3.4: Flow Chart of Model Used

Chapter 4

EXPERIMENTS AND RESULTS

Classification problems for complex data such as images can be solved using neural networks with multiple hidden layers. Different level of abstraction are helpful in learning features at each layer. But, it is still difficult to train neural networks with multiple hidden layers.

By training one layer at a time, we can train a neural network effectively. A special type of network known as an autoencoder can be used for training each anticipated hidden layer.

Classification of digits which are in the form of images has been done in this chapter by training a neural network with two hidden layers. Hidden layers are trained individually in using autoencoders in an unsupervised manner. At last, we train a final softmax layer, and all the layers are joined together to form a deep network, which is trained in a supervised manner just one time.

4.1 SYSTEM CONFIGURATION

The following system configuration has been used while conducting the experiments:

- Processor: Intel Core i5
- Clock Speed: 2.30 GHz
- Main Memory: 6 GB
- Hard Disk Capacity: 750 GB
- Software Used: MATLAB R2015a

4.2 DATABASE

Since there are no standard databases available for hindi numerals, database built by two students, Akarshan Sarkar and Kritika Singh who were guided by Prof. Amitabha Mukherjee was taken as it has nearly 41,000 samples for training with good inter as well as intra-class variability.

Digits	No. of samples
0	25740
1	43060
2	24950
3	24660
4	15840
5	28160
6	13950
7	6690
8	7230
9	29410

Table 4.1 showing number of samples of each numeral in the database

They have used the following two methods to increase the variation between training samples-

- Rotation- Each sample in the original database has been rotated by two arbitrarily selected angles one in the range +5◦ to +10◦ and other in the range −5 ◦ to −10◦ [1].

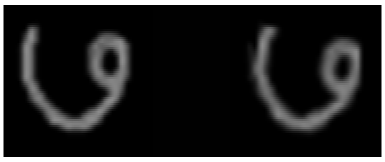


Fig 4.1 Before and After performing Rotation

- Elastic Distortion[4] - Uniform distribution is generated for first random displacement fields(between -1 and +1). Normalization(by a norm of 1) is applied after convolution with a Gaussian of standard deviation σ on the displacement fields. When values of σ are neither too high nor too low, where σ stands for the elasticity coefficient, the displacement fields look like elastic deformation. Moreover, α, which is the scaling factor is multiplied to displacement field to controls the elasticity of deformation. New values are generated by applying the displacement field to the original image pixels.

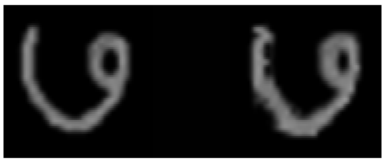


Fig 4.2 Before and After performing Elastic Distortion

4.3 EXPERIMENTAL PROCEDURE

For training the network, we use a matrix where columns are used to represent a single image. We can build this, by assembling the columns of an image to form a vector, and then creating a matrix from these vectors. Hence, the input matrix has the dimensions 784-by-41,068 as every image has size 28*28 and there are 41,068 samples in the database. The labels for the images are kept in a 10-by-41,068 matrix, where all other elements in the column will be 0 except a single element which will be 1 that will specify which class that digit belong to.

4.3.1 Training the First Autoencoder

We begin by training a sparse autoencoder on the training data in an unsupervised manner without the help of labels. An autoencoder is a neural network whose basic functionality is duplication of its input at its output. Thus, the input and output will be of the same size. A compacted representation of the input is leaned by the autoencoder as the size of the input is more than the size of the hidden layer.

A feed-forward network was used to create the autoencoder, and then some of the settings were modified. The subsequent significant thing is to establish the size of the hidden layer for the autoencoder. For the autoencoder that we are going to train, hiddenSize1 = 100 is chosen which is lesser than the input layer size of 784.

The training algorithm used is Scaled Conjugate Gradient Backpropagation and the network is trained to a maximum of 400 epochs. The transfer function used at both layers is logistic sigmoid.

hiddenSize1	100
Training function	trainscg
Epochs	400
Transfer Function	logsig
Dividing Function	dividetrain

Table 4.2 Parameter values for training the first layer autoencoder

Regularizers were added to boost the autoencoder to absorb a sparse representation in the first layer. By using the performance function |msesparse| we can regulate the effect of these regularizers by setting numerous parameters:

- |L2WeightRegularization| should be typically quite small as it regulates the allowance of an L2 regularizer for the weights of the network (and not the biases).
- |sparsityRegularization| regulates the allowance of a sparsity regularizer, which inhibits large fractions of the neurons in the hidden layer from activating in response to an input.
- |sparsity| generally has values between 0 and 1 and regulates the fraction of neurons that should be activated in the first layer in response to an input.

The values used by us for training autoencoders at first layer is-

L2WeightRegularization	0.004
sparsityRegularization	4
Sparsity	0.5

Table 4.3 Parameter values for the performance function msesparse

Hence, after training the first autoencoder, the network would look like

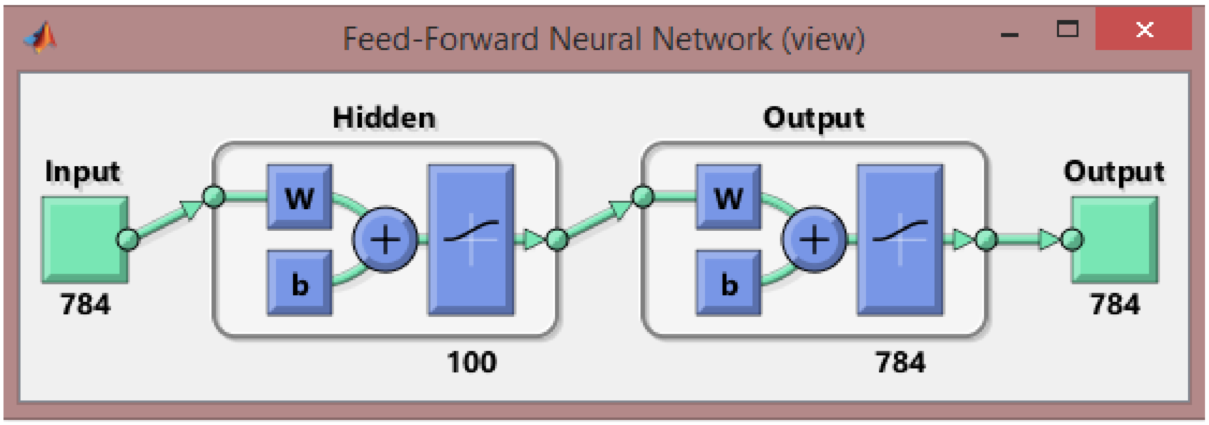


Fig 4.3 Network structure of the first autoencoder

After removing the topmost layer, the network now looks like -

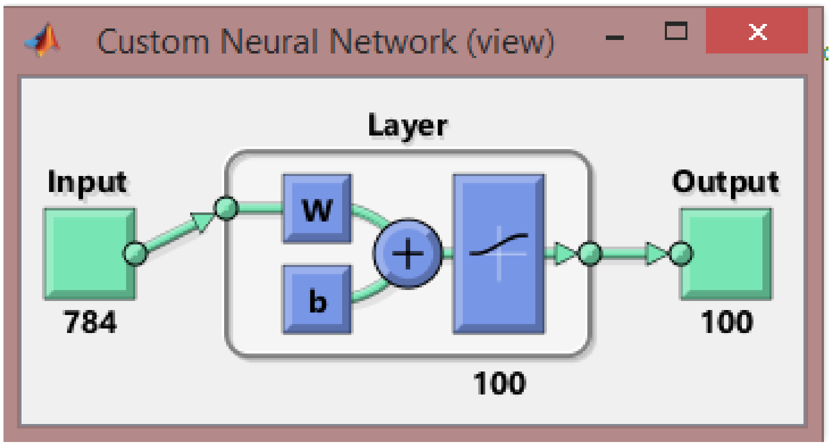


Fig 4.4 Modified structure of first layer autoencoder after removing the top layer

4.3.2 Training the Second Autoencoder

The top layer of the modified network having 100 nodes is actually is the hidden later of the first autoencoder, which actually contains summarized input actually fed to the autoencoder. Training the next autoencoder on this summarized input is done by again using a feed-forward network with the input layer n output layer having the same number of nodes, which are greater than the number of nodes in the hidden layer. The main difference between the first layer autoencoder and the second layer autoencoder is that the whole of the input vector was fed to the first layer autoencoder as it is whereas the inputs fed to the second layer autoencoder have been greatly reduced in dimensionality.

hiddenSize1	50
Training function	Trainscg
Epochs	100
Transfer Function	Logsig
Dividing Function	Dividetrain

Table 4.4 Parameter values for training the first layer autoencoder

|msesparse| is used as the performance function after the creation of network. Mean squared error with L2 weight and sparsity regularizers are used for the performance function.

L2WeightRegularization	0.002
sparsityRegularization	4
sparsity	0.1

Table 4.5 Parameter values for the performance function msesparse

Thus, the autoencoder created would be

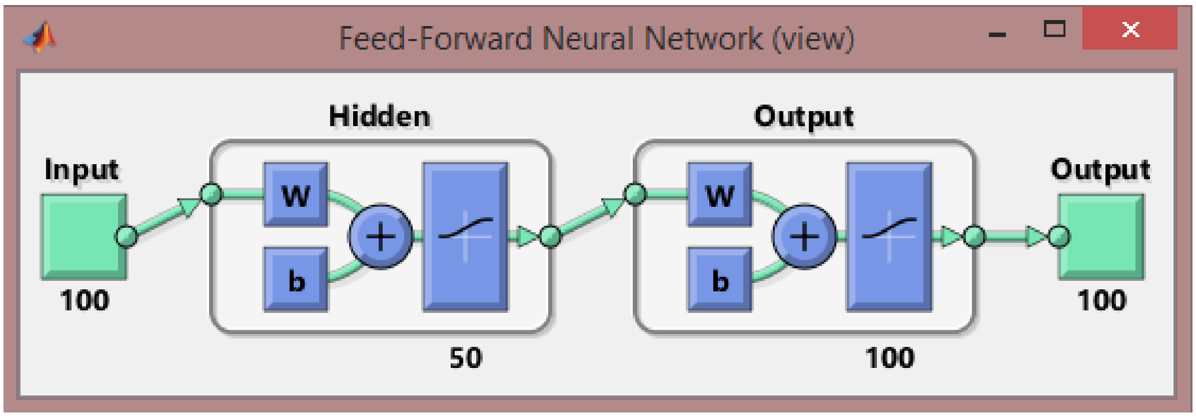


Fig 4.5 Network structure of the second layer autoencoder

And after removing the top layer, the autoencoder now looks like -

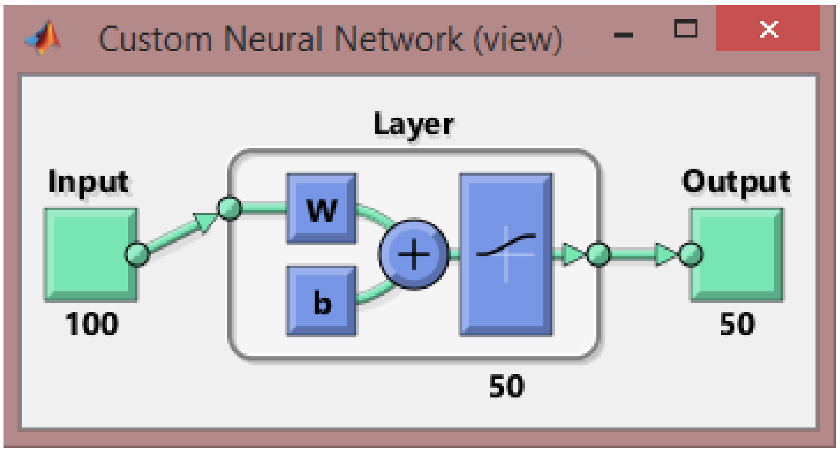


Fig 4.6 Modified structure of second layer autoencoder after removing the top layer

4.3.3 Training the final Softmax Layer

The input vector of the training data had 784 dimensions. It was condensed to 100 dimensions after passing through the first layer autoencoder. This was further condensed to 50 dimensions after it had passed through the second layer autoencoder. A final layer needs to be trained to classify these 50 dimensional vectors into 10 digit classes.

We create a softmax layer, and train it on the output from the hidden layer of the second autoencoder. As the softmax layer comprises of only one layer, hence it is created manually.

hiddenLayerSize	None
Training function	Trainscg
Epochs	400
Transfer Function	softmax
Dividing Function	dividetrain
Performance Function	Crossentropy

Table 4.6 Parameter values for training the final softmax layer

Unlike previous layers, this layer is trained in a supervised manner and hence the network looks like -

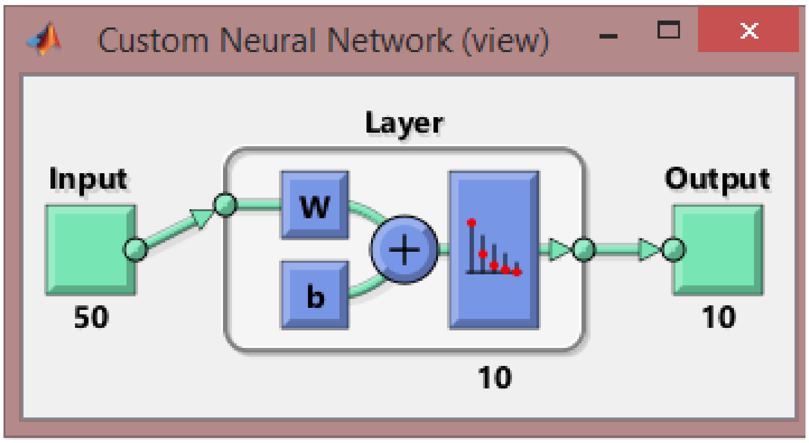


Fig 4.7 Network structure of the final softmax layer

4.3.4 Forming a Multilayer Neural Network

Three separate constituents of a deep neural network have been trained in segregation. They are -

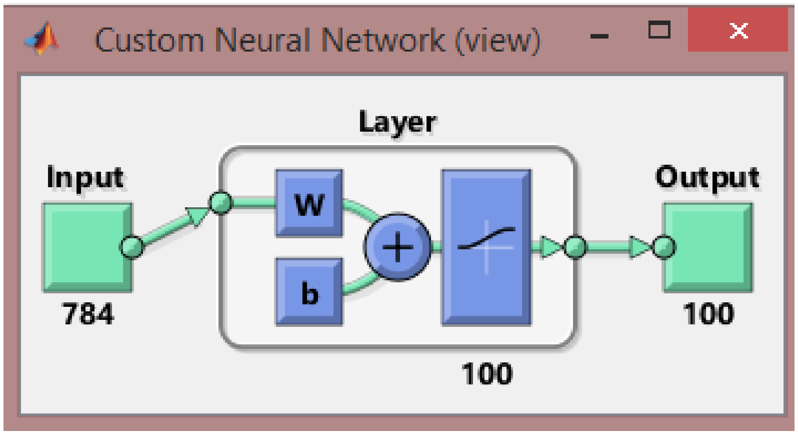


Fig 4.8 First Layer Autoencoder

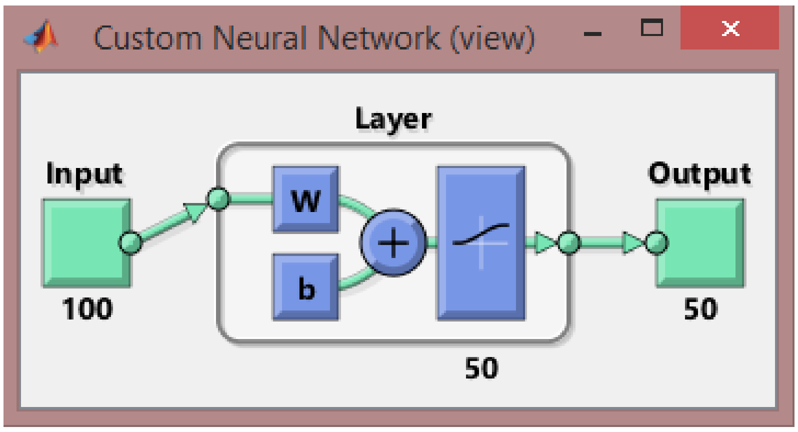


Fig 4.9 Second Layer Autoencoder

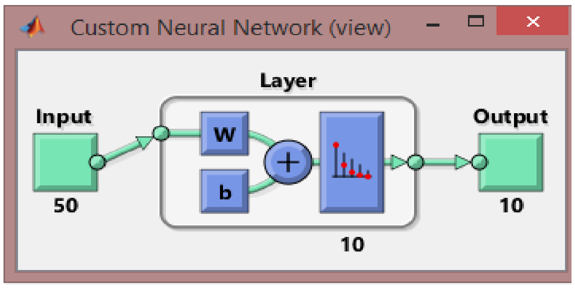


Fig 4.10 Final Softmax Layer

Now these three are joined together to form one network that looks like -

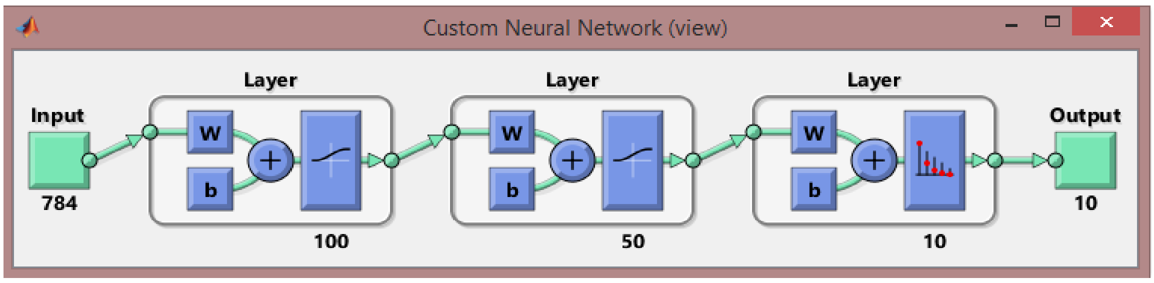


Fig 4.11 Final Network formed by joining the above three networks

For training the final network

hiddenSize1	100
hiddenSize2	50
Final Layer Size	10
Training Function	Trainscg
Epochs	100
Transfer Function(1)	Logsig
Transfer Function(2)	Logsig
Transfer Function(3)	Softmax
Dividing Function	Dividetrain
Performance Function	Crossentropy

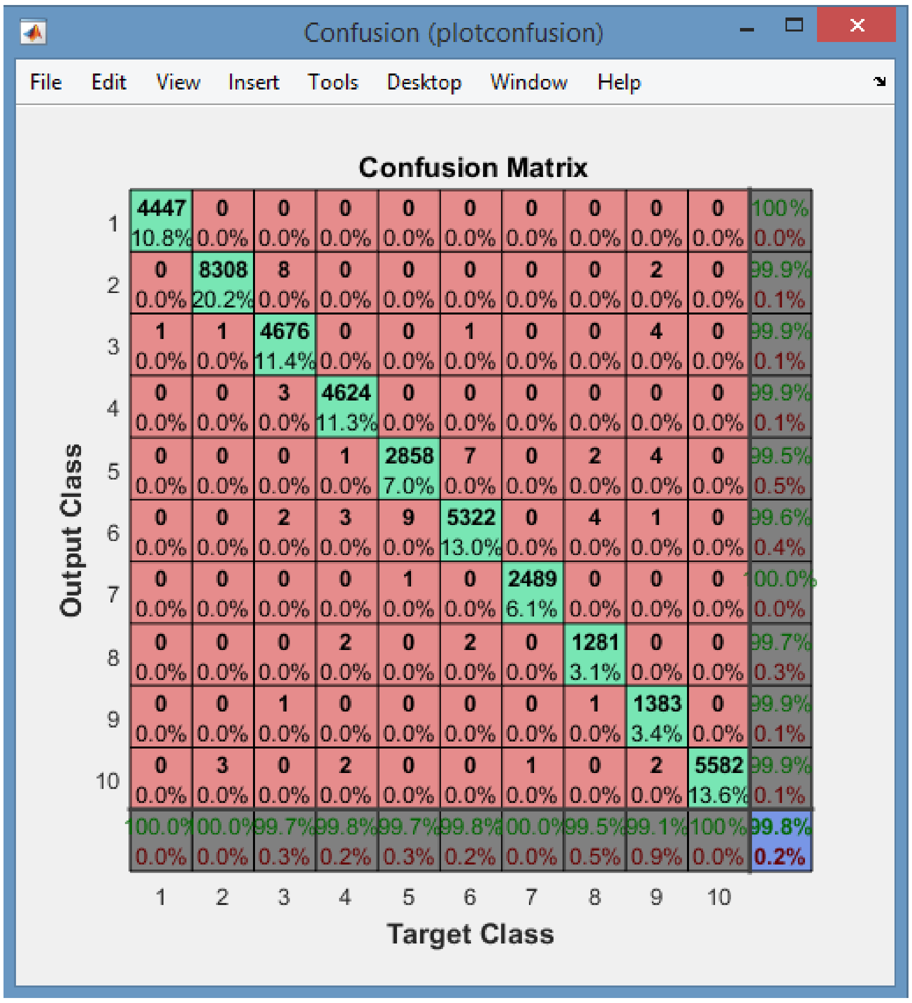
Table 4.7 Parameter values for training the final network

Hence the complete structure of the proposed network is complete.

4.4 RESULTS OBTAINED

The results for the deep neural network can be improved by performing backpropagation on the whole multilayer network. This process is often referred to as fine tuning. We fine-tuned the network by retraining it on the training data in a supervised fashion. We then viewed the results again using a confusion matrix.

In confusion matrix, we can see how many samples were classified correctly as well as incorrectly thus knowing the accuracy of each numeral. Thus, the training accuracy came out to be 99.8% as shown by this confusion plot.



Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	
1	4447 10.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	8308 20.2%	8 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	99.9% 0.1%
3	1 0.0%	1 0.0%	4676 11.4%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	4 0.0%	0 0.0%	99.9% 0.1%
4	0 0.0%	0 0.0%	3 0.0%	4624 11.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.9% 0.1%
5	0 0.0%	0 0.0%	0 0.0%	1 0.0%	2858 7.0%	7 0.0%	0 0.0%	2 0.0%	4 0.0%	0 0.0%	99.5% 0.5%
6	0 0.0%	0 0.0%	2 0.0%	3 0.0%	9 0.0%	5322 13.0%	0 0.0%	4 0.0%	1 0.0%	0 0.0%	99.6% 0.4%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	2489 6.1%	0 0.0%	0 0.0%	0 0.0%	100.0% 0.0%
8	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	2 0.0%	0 0.0%	1281 3.1%	0 0.0%	0 0.0%	99.7% 0.3%
9	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	1383 3.4%	0 0.0%	99.9% 0.1%
10	0 0.0%	3 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	2 0.0%	5582 13.6%	99.9% 0.1%
	100.0% 0.0%	100.0% 0.0%	99.7% 0.3%	99.8% 0.2%	99.7% 0.3%	99.8% 0.2%	100.0% 0.0%	99.5% 0.5%	99.1% 0.9%	100% 0.0%	99.8% 0.2%

Fig 4.12 Confusion Plot showing the Training Accuracy for Proposed Method

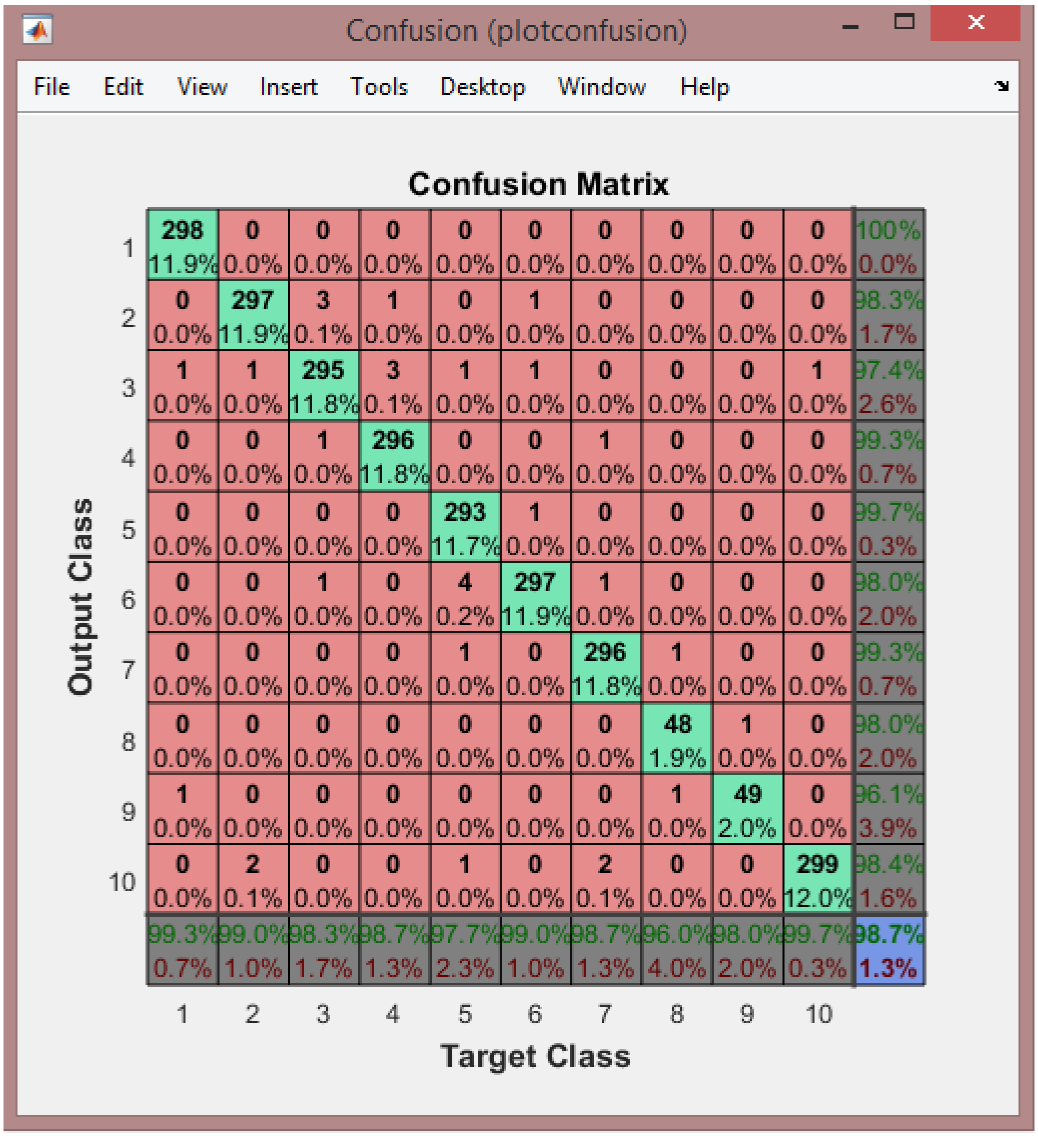
Explanation of the above confusion plot is further elaborated in the table below.

Digit	No. of samples correctly classified	No. of samples incorrectly classified	Accuracy
0	4447/4447	0/4447	100%
1	8308/8320	10/8320	99.9%
2	4676/4683	7/4683	99.9%
3	4624/4627	3/4627	99.9%
4	2858/2872	14/2872	99.5%
5	5322/5341	19/5341	99.6%
6	2489/2490	1/2490	100%
7	1281/1285	4/1285	99.7%
8	1383/1385	2/1385	99.9%
9	5582/5590	8/5590	99.9%
Overall Accuracy			99.8%

Table 4.8 Explanation of Training Confusion Plot for Proposed Method

For testing the network, 2500 samples were used which were previously not seen by the network.

Thus, the testing accuracy came out to be 98.7% as shown by this confusion plot.



Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	
1	298 11.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	297 11.9%	3 0.1%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.3% 1.7%
3	1 0.0%	1 0.0%	295 11.8%	3 0.1%	1 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	97.4% 2.6%
4	0 0.0%	0 0.0%	1 0.0%	296 11.8%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	99.3% 0.7%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	293 11.7%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.7% 0.3%
6	0 0.0%	0 0.0%	1 0.0%	0 0.0%	4 0.2%	297 11.9%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	98.0% 2.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	296 11.8%	1 0.0%	0 0.0%	0 0.0%	99.3% 0.7%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	48 1.9%	1 0.0%	0 0.0%	98.0% 2.0%
9	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	49 2.0%	0 0.0%	96.1% 3.9%
10	0 0.0%	2 0.1%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	2 0.1%	0 0.0%	0 0.0%	299 12.0%	98.4% 1.6%
	99.3% 0.7%	99.0% 1.0%	98.3% 1.7%	98.7% 1.3%	97.7% 2.3%	99.0% 1.0%	98.7% 1.3%	96.0% 4.0%	98.0% 2.0%	99.7% 0.3%	98.7% 1.3%

Fig 4.13 Confusion Plot showing the Testing Accuracy for Proposed Method

The explanation of the confusion plot can be better understood from the table below.

Digit	No. of samples correctly classified	No. of samples incorrectly classified	Accuracy
0	298/298	0/298	100%
1	297/302	5/302	98.3%
2	295/303	8/303	97.4%
3	296/298	2/298	99.3%

4	293/294	1/294	99.7%
5	297/303	6/303	98%
6	296/298	2/298	99.3%
7	48/49	1/49	98%
8	49/51	2/51	96.1%
9	299/304	5/304	98.4%
Overall Accuracy			98.7%

Table 4.9 Explanation of the Testing Confusion Plot for the Proposed Method

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

In this dissertation we saw the opportunities which deep learning brings forward in the domain of handwriting recognition through classification of hindi numeral images. Deep learning methods in vision have proven quite successful and in handwriting recognition and we made an attempt to realize its potential. Handwritten text images contain a lot of variance which is a lot of information to be exploited and deep learning implementation with unsupervised pre-training followed by supervised learning proved a viable candidate. In literature review we saw theoretically how better deep learning can prove at applications which require multiple layers of non–linear operations i.e. if we need to learn complex features that cannot happen with a shallow network.

Deep architectures are good but how do you extract features and then train the whole architecture? The answer to the first part of the question we came across in chapter 4 where we tried unsupervised pre-training on database and saw how without labels the neurons were able to learn the strokes of the handwritten digits. The idea of unsupervised pre-training has been adopted in literature successfully where a deep network has a lot of parameters and to estimate them it requires a lot of training data so pre-training tunes the parameters in a reduced space from where the supervise criteria can proceed.

Greedy layer-wise training helps training a deep network where parts of the deep network can be trained separately and can then be stacked together as a single system to form a deep learning model. We implemented greedy layer-wise training in all our experiments with the study area as stated in chapter 4.

5.2 FUTURE SCOPE

There are other intriguing possibilities also posed by Deep Learning which can be used in the handwritten recognition domain. We looked at how a deep network can be pre-trained and trained greedily to classify a hindi numeral image. Techniques such as convolutional neural networks which are excellent at object recognition due to the advantage of their translational invariance recognition can be used for detecting objects. Deep convolutional networks have been used for object recognition tasks and state of the art accuracy has been achieved. Restricted Boltzmann Machines which are energy based models have outperformed normal autoencoders in terms of dimensionality reduction using neural networks. These flavours of deep learning can be explored for tasks in handwriting recognition.

REFERENCES

[1] Bengio, Yoshua, "Learning deep architectures for AI." Foundations and trends in Machine Learning vol 2, no.1 (2009): 1-127. Now Publishers

[2] T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio, “A quantitative theory of immediate visual recognition,” Progress in Brain Research, Computational Neuroscience: Theoretical Insights into Brain Function, vol. 165 (2007): 33–56

[3] Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." The Journal of Machine Learning Research 11 (2010): 3371-3408.

[4] Yaser Abu-Mostafa, “Lecture notes from Caltech's Machine Learning course -CS 156”, [online] 2014 https://www.youtube.com/playlist?list=PLjUC8HjyxGTQ5F390csOAAHaQp8sguSx1,

[5] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar, Foundations of Machine Learning (2012), The MIT Press

[6] Freund, Yoav, and David Haussler. "Unsupervised learning of distributions on binary vectors using two layer networks." In Advances in Neural Information Processing Systems (1992): 912-919

[7] Arel, Itamar, Derek C. Rose, and Thomas P. Karnowski. "Deep machine learning-a new frontier in artificial intelligence research [research frontier]."Computational Intelligence Magazine, IEEE 5.4 (2010): 13-18

[8] LeCun, Yann A., Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. "Efficient backprop." In Neural networks: Tricks of the trade (2012): 9-48. Springer Berlin Heidelberg

[9] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313, no. 5786 (2006): 504-507.

[10] Andrew Ng, “Autoencoders and Sparsity”, [online] 2013, http://ufldl.stanford.edu/wiki/index.php/Autoencoders_and_Sparsity.

[11] Bengio, Yoshua, and Yann LeCun. "Scaling learning algorithms towards AI." Large-scale kernel machines 34, no.5 (2007). The MIT Press

[12] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. Lecun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pages 1{8. IEEE, 2007. 19.

[13] LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." Neural computation 1.4 (1989): 541-551

[14] Ciresan, Dan Claudiu, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. "Deep, big, simple neural nets for handwritten digit recognition." Neural computation 22, no. 12 (2010): 3207-3220

[15] Williams, DE Rumelhart GE Hinton RJ, and G. E. Hinton. "Learning representations by back-propagating errors." Nature (1986): 323-533

[16] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. pages 153{160, Apr. 2009. 3, 17.

[17] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. Neural Comput., 18(7):1527{1554, July 2006. ISSN 0899-7667. URL http://dx.doi.org/10.1162/neco.2006.18.7.1527. 3, 14,17, 18.

[18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. CoRR, abs/1207.0580, 2012b. 45.

[19] M. Hanmandlu and O. V. Ramana Murthy, “Fuzzy Model Based Recognition of Handwritten Hindi Numerals”, Pattern Recognition, Vol. 40, Issue 6, pp. 1840-1854, 2006.

[20] R.J. Ramteke, S.C. Mehrotra “Feature extraction based on Invariants Moment for handwritten Recognition”, Proc. Of 2nd IEEE Int. Conf. On Cybernetics Intelligent System (CIS2006), Bangkok, pp. 1-6, June 2006.

[21] U. Bhattacharya, S. K. Parui , B. Shaw, K. Bhattacharya, “ Neural Combination of ANN and HMM for Handwritten Devnagari Numeral Recognition”.

[22] A. Elnagar, S. Harous, Recognition of handwritten Hindu numerals using structural descriptors, J. Exp. Theor. Artif. Intell. 15 (3) (2003) 299–314.