Experiments on classification of Telugu characters using Counter Propagation Network

A Dissertation Submitted in the Partial Fulfillment of the Degree of

Master of Technology

in

Artificial Intelligence

By

Ramesh Naik Vankudothu

09MCMI34

Department of Computer and Information Sciences

University of Hyderabad

Hyderabad, Andhra Pradesh

India - 500046

April, 2011

CERTIFICATE

This is to certify that the project work entitled **Experiments on classification of Telugu characters using Counter Propagation Network**" being submitted to University of Hyderabad by **Ramesh Naik Vankudothu**, bearing Reg. No. 09MCMI34, in partial fulfillment for the award of the degree of Master of Technology in Computer Science, is a bonafide work carried out by him under my supervision.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Prof. Chakravarti Bhagavati
Project Supervisor,
Department of CIS,
University of Hyderabad.

Head ,
Department of CIS,
University of Hyderabad.

Dean,
School of MCIS,
University of Hyderabad.

DECLARATION

I, Ramesh Naik Vankudothu hereby declare that this Dissertation entitled **Experiments on classification of Telugu characters using Counter Propagation Network** submitted by me under the guidence and supervision of **Dr. Chakravarthy Bhagavati** is a bonafide work.

I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date: (Ramesh Naik V)

Place: Reg. No.09MCMI34

To,

My parents and all my well wishers

Acknowledgments

I would like to express my sincere gratitude to **Dr. Chakravarthy Bhagvati**, my project supervisor, for valuable suggestions and keen personal interest throughout the progress of my course of research. It is a matter of pride to have been associated with him.

I would like to express great thanks to **God** for giving me enough energy for doing hard work for the whole year.

I am grateful to **Prof. T. Amaranath**, Dean and **Prof. C R Rao**, Head of Department for providing excellent computing facilities and a congenial atmosphere for progressing with my project.

I would like thank **Mr. P Pavan kumar** for his valuable discussions and his answering for my questions during the project time.

I would like to take this opportunity to thank my friends who have been morale boosters to me and who encouraged me to take up this course and supported me throughout this course with their love and affection.

I would like to thank everyone who helped me knowingly or unknowingly during my whole research work.

With Sincere Regards,

Ramesh Naik V.

Abstract

Design of classifiers for Telugu OCR system is a complex task due to the presence of large number of classes. Its complexity is reduced to some extent if the recognition unit is taken as connected component (cc). But the number of ccs for Telugu script is around 400 which is still large. It is this complexity that throws a challenge to the classifiers. SVM and Neural Network based classifiers like Multi-layer perceptron, Convolutional Neural Networks, Counter Propagation Networks etc, are being investigated for handling these kind of large class problems. In the present work, we explore Counter Propagation Network (CPN) for the classification problem of Telugu OCR system. CPN has two layers: Instar and Outstar. In Instar, unsupervised learning is performed using only the input data and Winner-take-all approach is followed which is similar to that of Kohonen learning. In Outstar, supervised learning takes place using the target associated with each input. These two layers are trained separately. We have collected samples for each of the ccs (classes) from a corpus of Telugu documents to train the network. In the corpus, around 100 samples are obtained for more frequent classes and less number of samples (around 5) are found for less frequent classes. Using these samples, CPN is trained for 388 classes. CPN is trained using different learning rates and different number of iterations and accuracy on the training data is computed. Our experiments show that the basic CPN architecture may not be capable of handling the large number of classes required for Telugu character recognition. While high accuracy is otained for the 26-class problem of recognizing English letters, the same could not be achieved for larger problems.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Problem Definition

In this work we are exploring the capability of Counter propagation Neural Networks (CPN) for Telugu Optical Character Recognition system and examining the scalability of CNN for large multi-class problem such as Telugu OCR.

Telugu Optical Character Recognizing system (Telugu OCR) is to recognize Telugu character components by labelling them with appropriate labels. Telugu language has large number of components, approximately 400 classes to be recognized with respect to connected component strategy. These classes includes vowels, consonants, combination of consonants and vowel modifiers, votthu of consonants etc.

A network that self-organizes itself to implement an approximation to a function or mapping is called mapping neural network is the backpropagation network of Rumelhart [5]. The backpropagation network has been shown to be capable of implementing approximations to a variety of mappings from R[m] to R[n].Although most applications of backpropagation to date have been to mappings that have binary input and output vectors(i.e., each vector component is essentially either one or zero), there is ample evidence that backpropagation also works for at least some non-binary vector mappings.

In this experiment we discuss a different type of mapping neural network, called counterpropagation neural network(CPN). Under nonpathotical condition this network will self organize a near optimal lookup table approximation to the mapping used to generate its data. This method works equally well for both binary and continuous vector mapping.The counterpropagation network architecture is a combination of Kohonen learning and the

outstar structure of Grossberg. When a component is recognised as an instance of a class the classifier assigns a label for it directly that is provided at outstar unit.

1.2 Scope Of Work

In the literature, counter-propagation (CPN) architecture has been used for the recognition of handwritten English characters.Ahmed et al.,[4] made an attempt but only for digit recognition. The number of classes in both the examples is small: 26 for the former and 10 for the latter.

The main objective of this work is impementation of CPN for classifying Telugu characters which requires recognition of 388 classes and evaluate its performance. To the best of our knowledge, this is the first attempt to evaluate the performance of a CPN for such large number of classes.

1.3 Approach

Individual Telugu symbol images are extracted from documents and are given as input to the CPN in training phase. The number of training samples is proportional to the frequency of occurrence in Telugu text. The input is given as a sequence of pixels to the network and the output is a class label given in binary form. For recognizing 388 classes, the output is a 9-bit vector.

1.4 Organization of the Report

Chapter 2 explains background knowledge of different neural networks and basic architecture of Counter Propagation Neural network(CPN). Chapter 3 gives detailed description of CPN architecture that is being used to classify Telugu characters.And perform experiments on Telugu OCR. Chapter 4 gives the details of results on Telugu characters for various experiments. Chapter 5 describes experiments on English characters and its results. Chapter 6 does a performance analysis and makes observations based on our experiments. Chapter 7 contains the conclusion and future work possible in this area.

Chapter 2

Background

2.1 Neural Network

Neural networks are well known for their learning ability. Generally a neural network designed for recognition task under goes two phases. First is the training phase, the flow of operations taking place in this phase are shown in the Figure 2.1. As shown in the Figure 2.1 network has to first initialize the trainable parameters also called as weights with random values. Once the training set is provided to the network, each time network considers a sample from training set and updates the weights according to the error occurred with that sample, this method is called as stochastic method of learning. There is an other method for weights updation called as batch learning in which weights gets updated only after consideration of all the samples present in the training set.
There exists many algorithms for updating weights such as gradient descent back-propagation. If we take gradient descent back-propagation as a learning algorithm, mathematical equations for calculating output, finding the error and updating the weights are shown below.

Calculating output at each neuron :

$$X_j = f\{\Sigma_{i\epsilon M_j} X_i * K_{ij} + b_j\} \tag{2.1}$$

Here X_j is the output at present unit, X_i is output of unit in the previous layer,k_{ij} is connetion weight between connecting layer j and layer i, b_j is trainable bias for unit in layer j.

Calculating Error

$$E = \frac{1}{2}\sum_{n=1}^{N}(t^n - X^n) \tag{2.2}$$

$t^n$is the target value and X^n is achieved output.

Calculating gradient

$$\Delta W = -\eta\frac{\partial E}{\partial W} \tag{2.3}$$

Weight updation rule

$$W = W + \Delta W \tag{2.4}$$

The standard approach of neural network is fully connected neural networks which can learn complex, high dimensional, non linear mappings from large collection of examples.

2.2 Counter Propagation Netwrok

Hecht-nelson [13] Figure 2.2proposed CPN as an alternative function approximator which can be developed on the available input data. The name counterpropagation derives from the initial presentation of this network as a five-layered network with data flowing inward from both sides, through the middle layer and out the opposite sides. There is literally a counterflow of data through the network. Although this is an accurate picture of the network, it is unnecessarily complex; we can simplify it considerably with no loss of accuracy. In the simpler view of the counterpropagation network, it is a three-layered network.
The input layer is a simple fan-out layer presenting the input pattern to every neurode in the middle layer.It is a bi-directional mapping between input and output layers.In the essence, the while data is presented to the input layer to generate a classification pattern on the output layer, the output layer in turn would accept an additional input vector and generate an output classification on the network's input layer.The network got its name from this counter-posing flow of information through its structure.

The basic CPN architecture consists of Instar unit and Outstar unit i.ie., the three layered CPN is:
1.Input layer

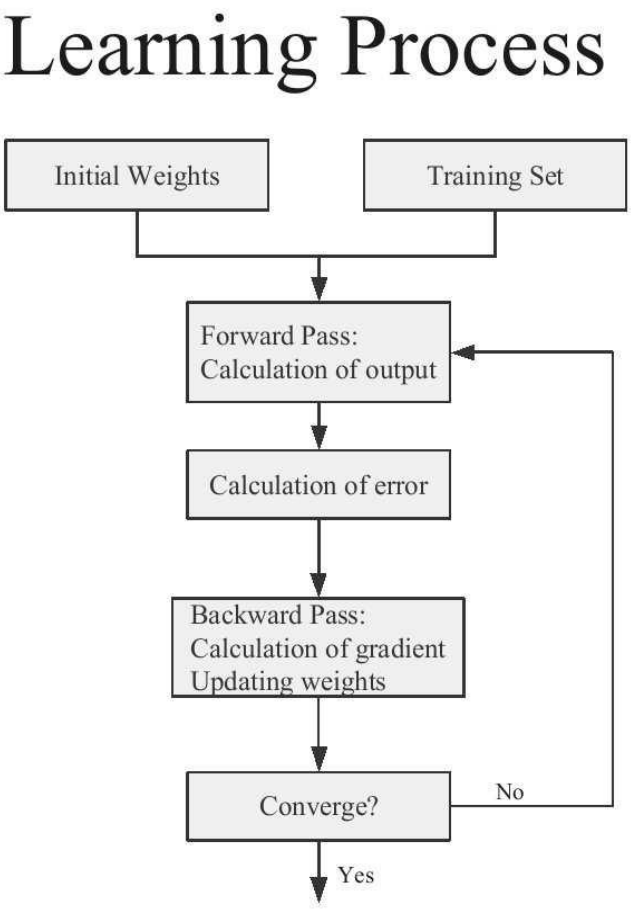

Figure 2.1: Learning Process in Neural Networks

2.Instar layer

3.Outstar layer

The underlying principle for CPN is simple: for a given independent variable vector **I** not present in the available data set, find the independent variable vector in the data set closest to **I**. The criterion of closeness in the n-dimensional Euclidean space can either be distance based(minimum Euclidean distance) or can be angle based(minimum angle between vectors of normalized lengths).If X_k is the vector found closest in the data set,then the value of$f$ (**I**) can be approximated as the dependent variable value corresponding to X_k.

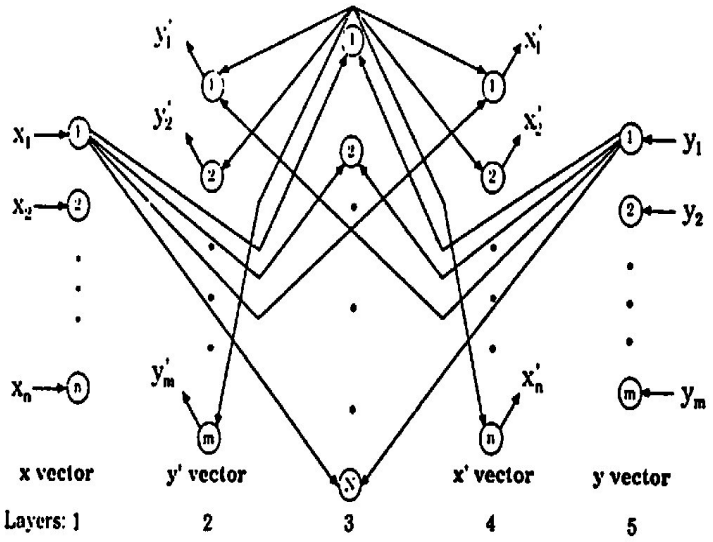

Figure 2.2: Topology of counter propagation network

2.3 Architecture

The proposed CPN architecture figure 2.3 comprising of a Kohonen learning at Instar layer and Grossberg learning at Outstar layer.

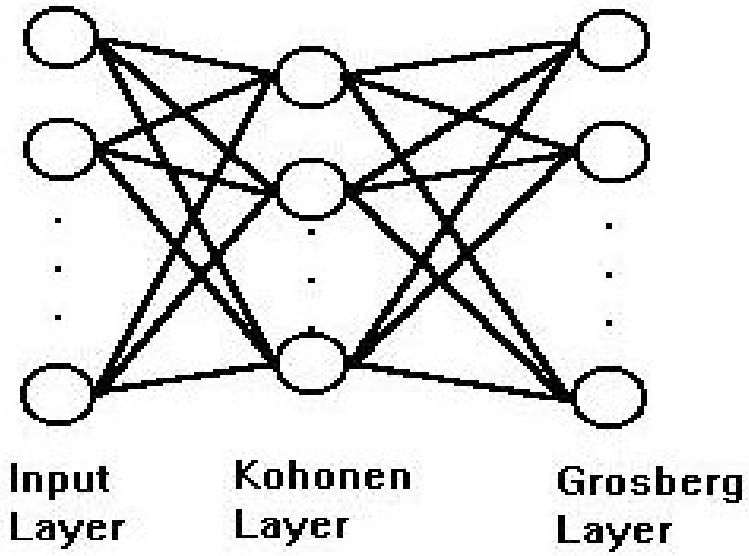

Figure 2.3: counter propagation network architecture

2.3.1 Instar Layer

The Instar layer in a CPN is performed using Kohonen learning.

Kohonen Neural network

The Kohonen neural network differs considerably from the feed-forward back propagation neural network. The Kohonen neural network differs both in how it is trained and how it recalls a pattern. The Kohonen neural network does not use any sort of activation function. Further, the Kohonen neural network does not use any sort of a bias weight.Output from the Kohonen neural network does not consist of the output of several neurons. When a pattern is presented to a Kohonen network one of the output neurons is selected as a "winner". This "winning" neuron is the output from the Kohonen network. Often these "winning" neurons represent groups in the data that is presented to the Kohonen network. The most significant difference between the Kohonen neural network and the feed forward back propagation neural network is that the Kohonen network trained in an unsupervised mode. This means that the Kohonen network is presented with data, but the correct output that corresponds to that data is not specified. Using the Kohonen network this data can be classified into groups. We will begin our review of the Kohonen network by examining the training process.

How a Kohonen Network Recognizes

I will now show you how the Kohonen neural network recognizes a pattern. We will begin by examining the structure of the Kohonen neural network. Once you understand the structure of the Kohonen neural network, and how it recognizes patterns, you will be shown how to train the Kohonen neural network to properly recognize the patterns you desire. We will begin by examining the structure of the Kohonen neural network.

The Structure of the Kohonen Neural Network

The Kohonen neural network works differently than the feed forward neural network. The Kohonen neural network contains only an input and output layer of neurons. There is no hidden layer in a Kohonen neural network. First we will examine the input and output to a Kohonen neural network.The input to a Kohonen neural network is given to the neural network using the input neurons. These input neurons are each given the floating point numbers that make up the input pattern to the network. A Kohonen neural network requires that these inputs be normalized to the range between -1 and 1. Presenting an input pattern to the network will cause a reaction from the output neurons.The output of a Kohonen neural network is very different from the output of a feed forward neural network. if we had a neural network with five output neurons we would be given an output

that consisted of five values. This is not the case with the Kohonen neural network. In a Kohonen neural network only one of the output neurons actually produces a value. Additionally, this single value is either true or false. When the pattern is presented to the Kohonen neural network, one single output neuron is chosen as the output neuron. Therefore, the output from the Kohonen neural network is usually the index of the neuron (i.e. Neuron #5) that fired. The structure of a typical Kohonen neural network is shown in Figure 2.4.

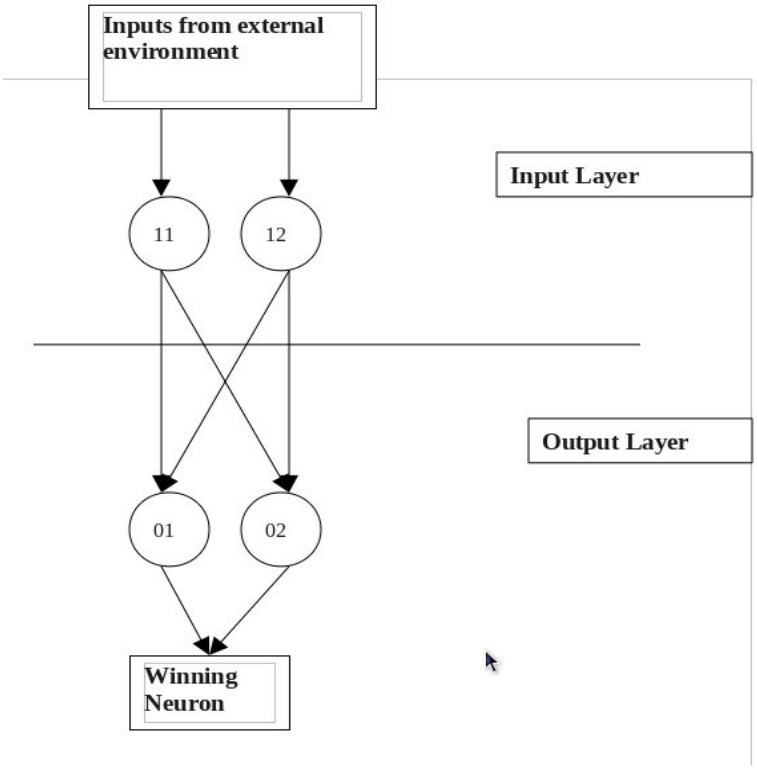

Figure 2.4: Kohonen Neural Network

How a Kohonen Network Learns

In this section we describe how we will learn to train a Kohonen neural network. There several steps involved in this training process. Overall the process for training a Kohonen neural network involves stepping through several epochs until the error of the Kohonen neural network is below acceptable level. In this section we will learn these individual processes. You'll learn how to calculate the error rate for Kohonen neural network, you'll learn how to adjust the weights for each epoch. You will also learn to determine when no more epochs are necessary to further train the neural network. The training process for

the Kohonen neural network is competitive. For each training set one neuron will "win". This winning neuron will have its weight adjusted so that it will react even more strongly to the input the next time. As different neurons win for different patterns, their ability to recognize that particular pattern will be increased. We will first examine the overall process involving training the Kohonen neural network. These individual steps are summarized in figure 2.5.

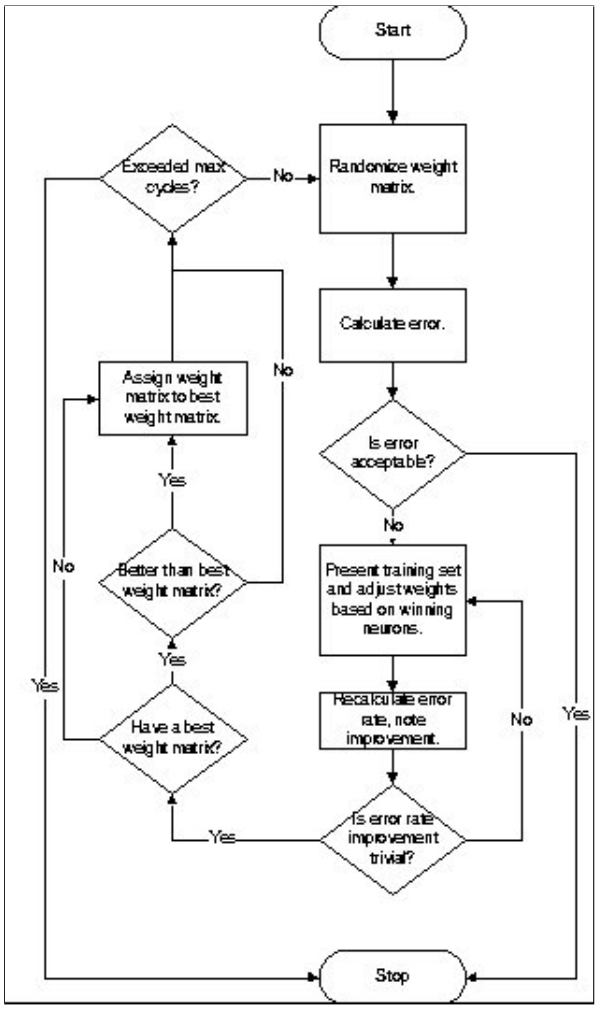

Figure 2.5: Training Kohonen Neural Network

Adjusting Weights

The entire memory of the Kohonen neural network is stored inside of the weighted connections between the input and output layer. The weights are adjusted in each epoc. An epoch occurs when training data is presented to the Kohonen neural network and the

weights are adjusted based on the results of this item of training data. The adjustments to the weights should produce a network that will yield more favorable results the next time the same training data is presented. Epochs continue as more and more data is presented to the network and the weights are adjusted.Eventually the return on these weight adjustments will diminish to the point that it is no longer valuable to continue with this particular set of weights. When this happens the entire weight matrix is reset to new random values. This forms a new cycle. The final weight matrix that will be used will be the best weight matrix determined from each of the cycles. We will now examine how these weights are transformed. The original method for calculating the changes to weights, which was proposed by Kohonen, is often called the additive method. This method uses the following equation.

$$w^{t+1} = \frac{w^t + \alpha x}{||w^t + \alpha x||} \tag{2.5}$$

The variable x is the training vector that was presented to the network. The variable w^t is the weight of the winning neuron,and the variable w^{t+1} is the new weight.The double vertical bars represent the vector length.

The additive method generally works well for Kohonen neural networks. Though in cases where the additive method shows excessive instability, and fails to converge, an alternate method can be used. This method is called the subtractive method. The subtractive method uses the following equations.

$$e = x - w^t \tag{2.6}$$

$$w^{t+1} = w^t + \alpha e \tag{2.7}$$

2.3.2 Outstar Layer

Supervised learning at Outstar layer is starts after completion of training at Instar layer.The Outstar learning rule at Outstar layer is Grossberg learning.The rule is to provide learning of repetitive and characteristic properties of input/output relationships. This rule is concerned with supervised learning It is supposed to allow the network to extract statistical properties of the input and output signals.The weight adjustments in this rule are computed as follows

$$\Delta w_{mj} = \beta(d - w_j) \tag{2.8}$$

or, the individual weight adjustments are

$$\Delta w_{mj} = \beta(d_m - w_{mj}), \text{for m=1,2,...,p} \tag{2.9}$$

Chapter 3

Counter Propagation Neural Network for Telugu

In this chapter, we have used CPN to recognize Telugu characters. This task is challenging as Telugu has a large number of classes (around 390). One critical requirement for kohonen layer of CPN is that initial weight vectors for different classes should be well separated in the hyper-space for proper training. If the number of classes is large, this requirement may not be satisfied.

Muhammad et al, [1] used CPN architecture to recognize handwritten English characters and they have considered only captial letters which are only 26. To the best of our knowledge, no one has used CPN to recognize Telugu characters and therefore, in the present work, we have attempted this problem.

3.1 Collection of Samples for training Data

To perform training, we have collected samples for all the classes from various Telugu documents. In the process, around 100 samples for the more frequent classes and around 10 samples for the remaining classes are obtained. We have followed a semi-automated approach to collect those samples. In this approach, a sample is fed to our existing K-NN classifier and its label is inspected and is corrected only if it is given wrong label. Around 8500 samples have been collected and we have shown few samples in figures 3.1 ,3.2 and 3.3.

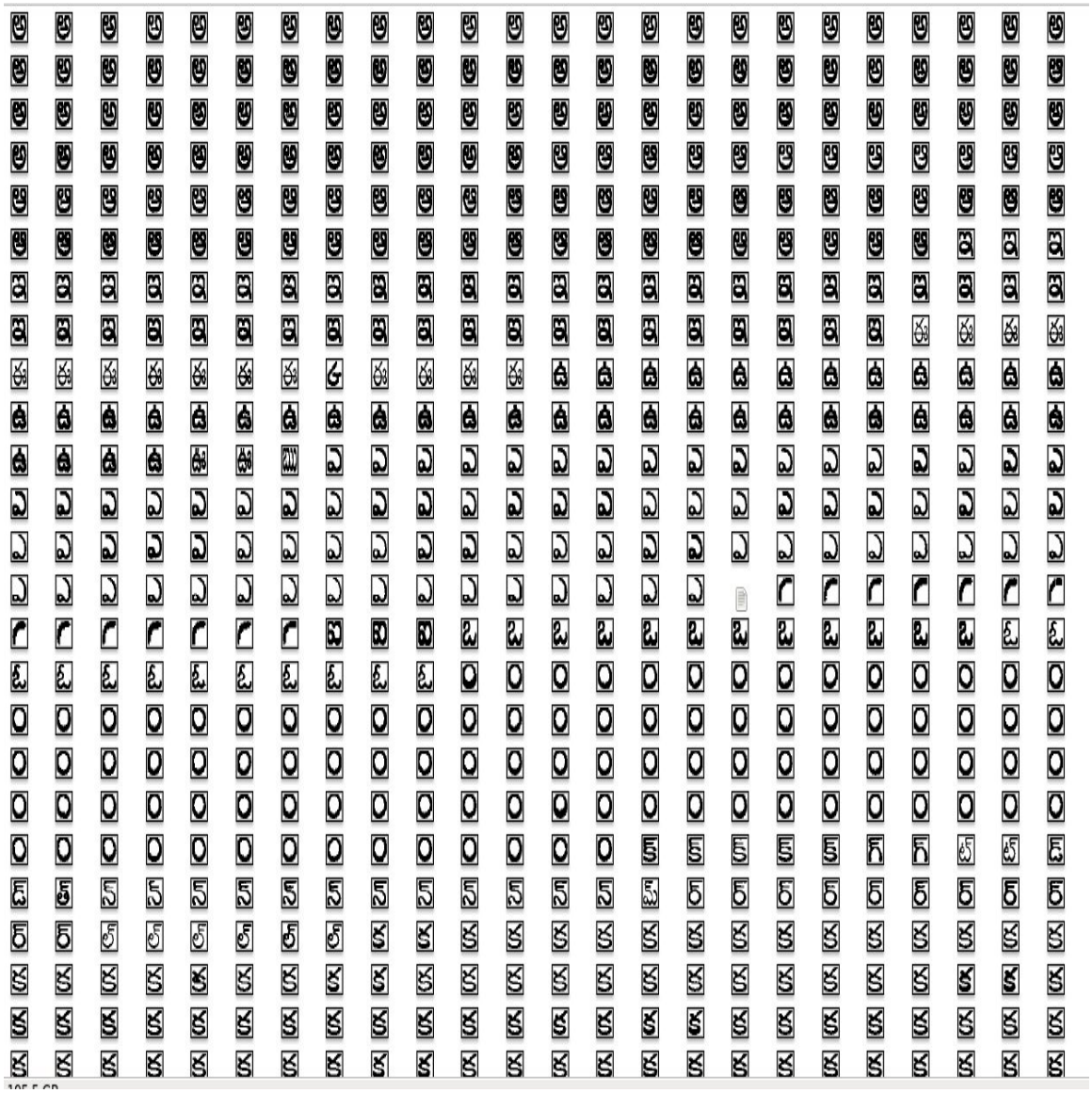

Figure 3.1: Collection of samples

3.2 Architecture

As discussed in the previous chapter and shown in Fig. 3.4, CPN architecture has three layers: Input Layer, Instar layer and Outstar layer. These layers will be discussed in the following sub-sections.

3.2.1 Input Layer

Each sample image is normalized to 32×32 size and is converted to one-dimensional (1D) vector of size 1024. This 1D representation of the image is normalized and used as a feature vector which is fed to the instar layer. If $\vec{u}$ is a 1D input vector, it is normalized using the

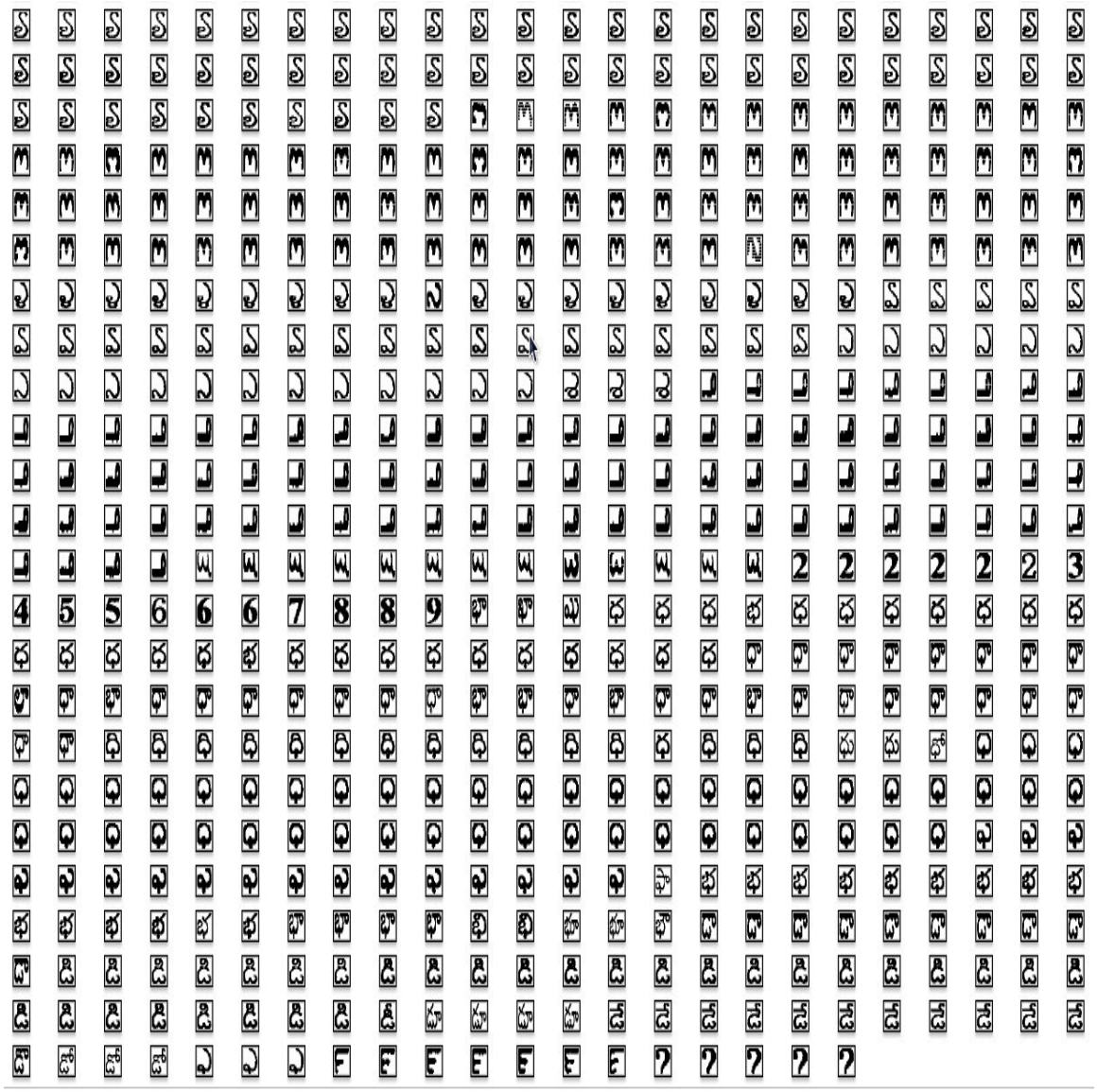

Figure 3.2: Collection of samples

equation given below:

$$N(\vec{u}) = \frac{\vec{u}}{||\vec{u}||} \tag{3.1}$$

Here, $N(\vec{u})$ is the resultant normalized vector and $||\vec{u}||$ is the magnitude of $\vec{u}$. For example, consider two neurons in the input layer whose input values are shown in Table **??**.

Table 3.1: Sample inputs

Input Neuron 1	0.5
Input Neuron 2	0.75

Now the magnitude of the input vector is calculated by adding the squares of the two

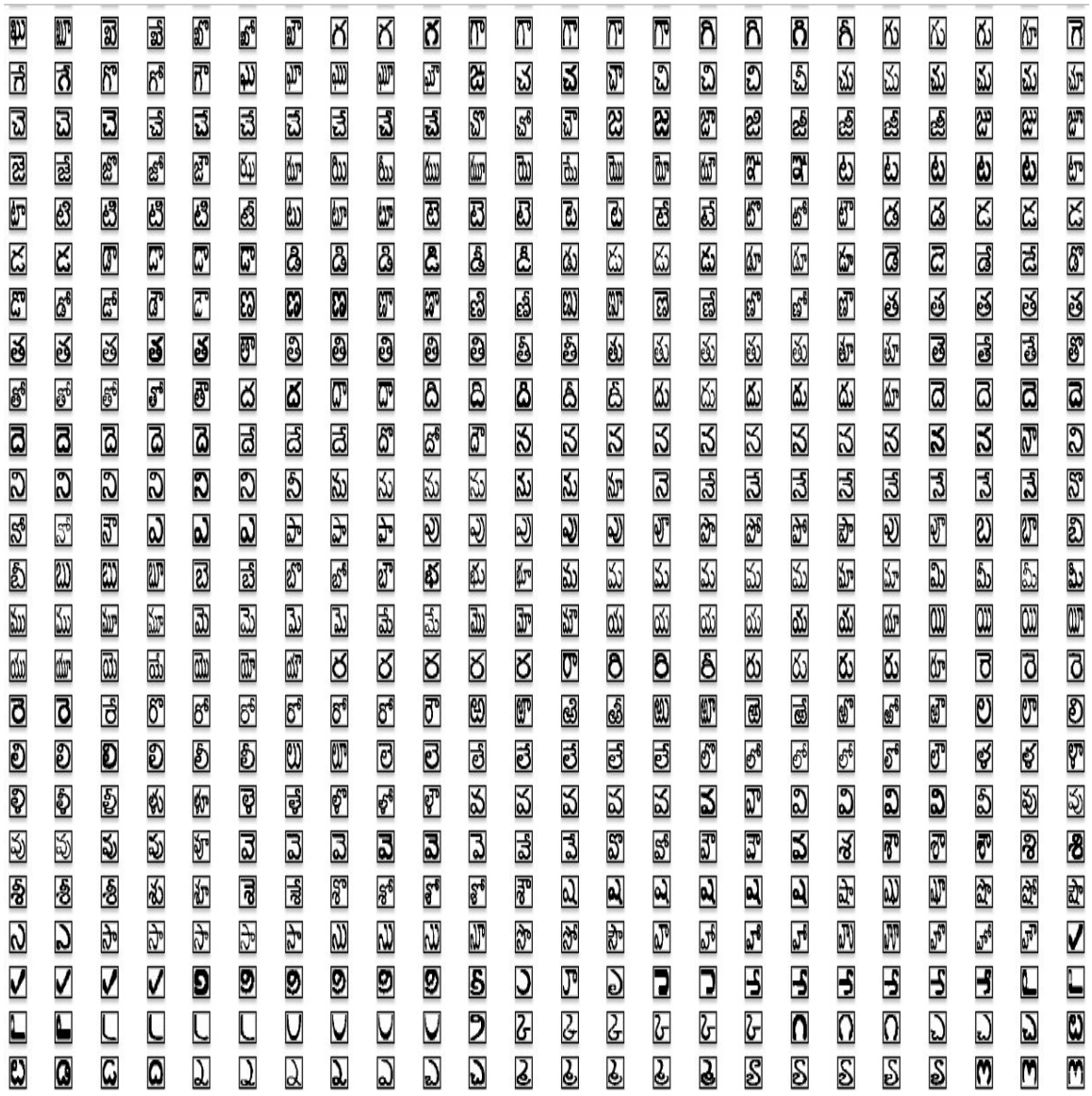

Figure 3.3: Collection of samples

inputs and taking the square root of the sum $\sqrt{(0.5 * 0.5) + (0.75 * 0.75)} = \sqrt{(0.8125)}$. The normalization factor is the reciprocal of this value $\frac{1}{\sqrt{0.8125}} = 1.1094$.

3.2.2 Instar Layer

As explained in the previous chapter, unsupervised training is performed in this layer. The number of neurons in this layer is equal to the number of classes which is 388 for Telugu. As shown in figure 3.4, all neurons from the input layer are connected to each neuron in this layer. Initial weights of the connections from input layer to each neuron in this layer are taken as the feature values of some sample picked from the class corresponding to that neuron. This ensures that initial weight vectors are well separated in hyper-space. The

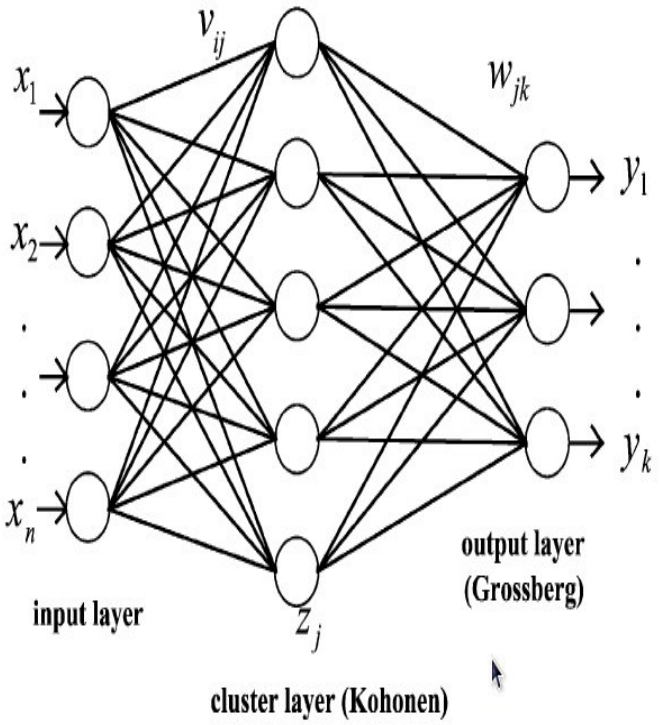

Figure 3.4: CPN

output y_j of each neuron is computed by taking the dot product between the input vector and the weight vector (of that neuron) and is given by the equation shown below:

$$y_j = \sum_{i,j} (w_{ij} * x_i) \tag{3.2}$$

Here, **Winner-Take-All** approach is followed to update the weights associated with the neurons. That means, only the weights of the neuron with maximum output are updated. The weight updation for that neuron is performed using the equation given below:

$$\Delta w_{i,j} = x_i - w_{i,j}$$

$$w_{i,j+1} = w_{i,j} + \alpha \Delta w_{i,j} \tag{3.3}$$

Where,

α is Learning Rate

$w_{i,j+1}$ is the updated weight

$w_{i,j}$ is current weight

Δw is the change in the weights

x_i is input to instar layer

3.2.3 Outstar Layer

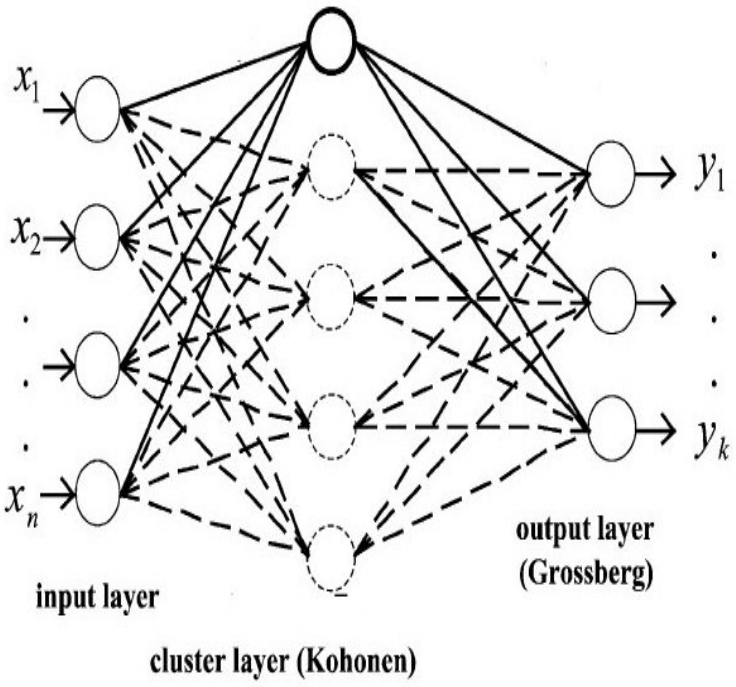

Figure 3.5: Architecture of CPN

As shown in figure 3.5, all the neurons from the instar layer are connected to all the neurons in the outstar layer. The number of neurons in this layer is equal to the dimensionality of the target vector. We have used binary encoding of class labels as target vectors. As we have a total of 388 classes, 9 bits are needed to encode them and hence 9 neurons are present in the outstar layer. Weights in this layer are initialized using the target vectors. After the completion of instar layer training, outstar training is performed in a supervised manner. In the training process (as shown in figure 3.5), input is fed to the instar layer and the winner neuron is determined. The weights $v_{i,j}$ associated with the connections from this neuron to the neurons in the outstar layer are only updated as follows:

$$\Delta v_{i,j} = d_i - v_{i,j} \tag{3.4}$$

$$v_{i,j+1} = v_{i,j} + \beta \Delta v_{i,j}$$

where,

d_i is the desired output.

β is Learning Rate.

$\Delta v_{i,j}$ is the change in the weights.

Chapter 4

Results on Telugu Characters

As discussed in the previous chapters, in our work, we have explored the learning ability of the Counter propagation Neural Network for large number of classes of Telugu. In the process, we have conducted several experiments by changing different parameters like input vector size, number of epochs etc, for training and in this chapter, we have presented the corresponding results.

Learning Rate $\alpha = 0.30$

$\beta = 0.44$

4.1 Experiment 1

In this experiment, a total of 762 samples covering 388 classes, each of size 32×32 are taken. For different number of epochs, we have computed accuracies on the training data and are shown in Table 4.1. Accuracy on the training data is computed as the ratio of number of training samples correctly classified by the network to the total number of training samples.

4.2 Experiment 2

In this experiment, a total of 762 samples covering 388 classes are taken, but the size of the sample images are taken as 64×64. Accuracies on the training data for different number of epochs are shown in Table 4.2.

Table 4.1: Accuracy on Training Data of 388 classes

Number of Epochs	Accuracy
100	18.635
200	19.29
500	18.24
900	17.71
1000	17.29
1400	17.716

Table 4.2: Accuracy on Training Data of 388 classes for 64x64

Number of Epochs	Accuracy
500	18.71
600	17.19
700	17.71
800	18.23
1000	17.54

4.3 Experiment 3

As promising results are not obtained for 388 classes, we have reduced the number of classes to 289. A total of 562 training samples from 289 classes, each of size 64×64 are considered and the accuracies on the training data for different number of epochs are shown in Table 4.3.

4.4 Experiment 4

Here, a total of 116 training samples, each of size 64×64 and covering 53 different classes (shown in figure 4.1) are considered. The accuracies on the training data ae shown in Table 4.4.

Table 4.3: Accuracy on Training Data of 289 classes for 64x64

Number of Epochs	Accuracy
500	20.106
600	19.25
700	19.73
800	20.43
1000	19.21
1400	17.716

Table 4.4: Accuracy on Training Data of 53 classes for 64x64

Number of Epochs	Accuracy
100	54.31
200	55.172
300	55.172
500	51.724
600	54.31
700	54.31
800	52.88
1000	52.58

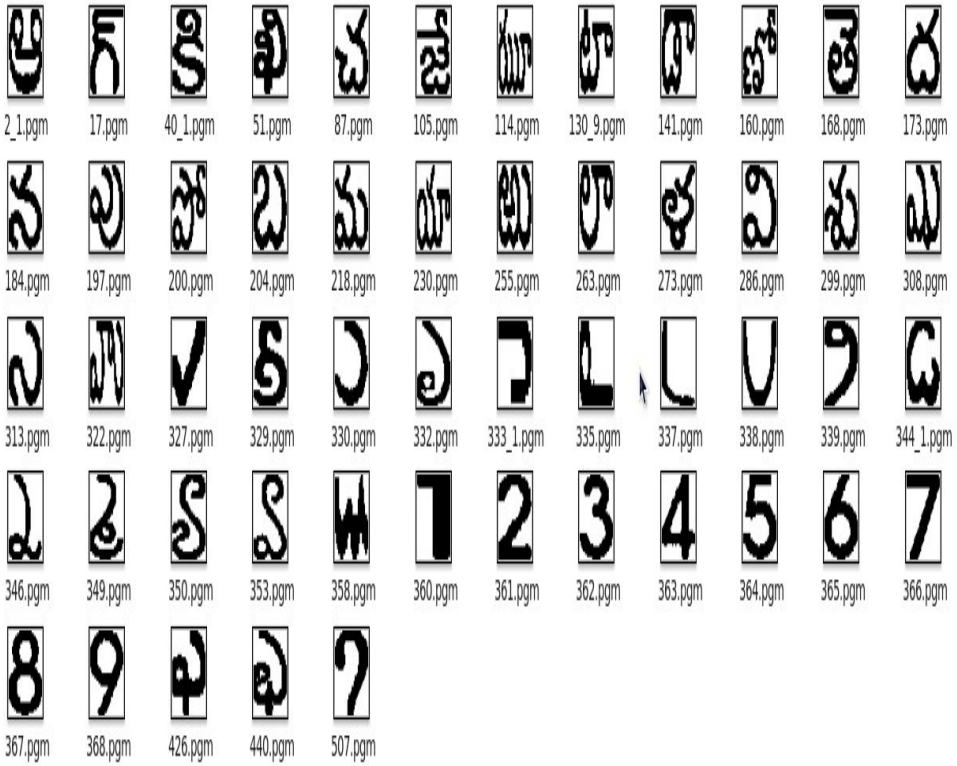

Figure 4.1: 53 different class samples taken for weight initialization

Chapter 5

Results on English Characters

As promising results are not obtained for Telugu characters, we have tried on English characters to find whether the network is not performing well due to large number of classes.

5.1 Experiment 1

This experiment has been conducted using the following tunable parameters of the network and the accuracies are shown in Table 5.1.

1. Size of input image - 32×32
2. Number of classes - 26 (capital letters)
3. Number of training samples - 130 English Letter Images
4. Target output vector size - 5 (encoding of 26 classes)
5. Initial weights - random
6. Activation function - sigmoid function
7. Learning Rate at Instar layer $\alpha = 0.10$
8. Learning Rate at Outstar layer $\beta = 0.0001$

Table 5.1: Accuracy on Training Data on26 classes for 32x32 with Random weight initialization and Squashing function as Sigmoid function

Number of Epochs	Accuracy
1	4.615
10	4.615
50	6.9230
100	5.384
200	5.3846
300	3.846
400	5.3846
500	6.1538
600	4.615
700	4.615
1000	4.615
1500	5.385

5.2 Experiment 2

This experiment has used the following parameters and the corresponding accuracies are shown in Table 5.2.

1. Size of input image - 32×32
2. Number of classes - 26(capital letters)
3. Number of training samples - 130 English Letter Images
4. Target output vector size - 5 (encoding of 26 classes)
5. Initial weights - random
6. Activation function - Linear function
7. Learning Rate at Instar layer $\alpha = 0.10$
8. Learning Rate at Outstar layer $\beta = 0.10$

Table 5.2: Accuracy on Training Data on 26 classes for 32x32 with Random weight initialization with squashing function as Linear function

Number of Epochs	Accuracy
1	4.615
10	4.615
50	6.9230
100	5.384
200	5.3846
300	3.846
400	5.3846
500	6.1538
600	4.615
700	4.615
1000	4.615
1500	5.385

5.3 Experiment 3

This experiment has used the following parameters and the corresponding accuracies are shown in Table 5.3.

1. Size of input image - 32×32
2. Number of classes - 26(capital letters)
3. Number of training samples - 130 English Letter Images
4. Target output vector size - 5 (encoding of 26 classes)
5. Initial weights - from samples
6. Activation function - Linear function
7. Learning Rate at Instar layer $\alpha = 0.10$
8. Learning Rate at Outstar layer $\beta = 0.0001$

5.4 Experiment 4

This experiment has used the following parameters and the corresponding accuracies are shown in Table 5.4.

1. Size of input image - 32×32
2. Number of classes - 52 (all letters)
3. Number of training samples - 780 English Letter Images
4. Target output vector size - 5 (encoding of 26 classes)
5. Initial weights - from samples
6. Activation function - Linear function
7. Learning Rate at Instar layer $\alpha = 0.10$
8. Learning Rate at Outstar layer $\beta = 0.0001$

Table 5.3: Accuracy on Training Data on 26 classes for 32x32 with weights initialized by image label itself and squashing function as Linear function

Number of Epochs	Accuracy
10	86.1538
12	87.692307
13	87.692307
14	88.461533
15	88.461533
20	88.461533
50	88.461533
100	88.461533
500	88.461533
600	88.461533
650	88.461533
675	88.461533
688	80.761226
699	80.761226
700	80.761226
1000	80.761226
1500	23.076923
2000	23.076923
2500	23.076923
3000	23.076923

Table 5.4: Accuracy on Training Data on 52 classes for 32x32 at image as weight initialization

Number of Epochs	Accuracy
1	19.487181
2	19.615385
3	19.743589
4	19.230770
5	19.743559
6	19.487181
10	18.076929
100	17.820513

Chapter 6

Analysis and Discussion

In our experimentation, it is observed that several parameters affect the performance of CPN which are listed below:

1. Number of classes
2. Input size
3. Features
4. Training set size

6.1 Observations

We have made the following observations from our experiments:

1. **CPN may not be easy to implement for large number of classes**
   As shown in the previous chapter, CPN architecture is not performing well for large number of classes of Telugu. If the number of classes is taken as 388, the accuracy on the training data is only around 17%. If the number of classes is reduced to 290, an accuracy of around 19% is obtained. If it is further reduced to 55, an accuracy of around 55% is obtained.

   Similarly, for English, if only capital letters are considered (26 classes), an accuracy of around 88% is obtained as presented in the earlier chapter. But, if all the English characters (52) are considered, an accuracy of around 19% is obtained. From the above results, it can be concluded that CPN architecture is difficult to implement for large number of classes.

2. **Importance of choosing correct values for α and β**

   Learning parameters play crucial role in the training process. As can be seen from the previous chapter, if the α and β values are 0.3 and 0.44 respectively, an accuracy of around 23% is obtained for English capital letters. If their values are 0.1 and 0.0001, an accuracy of around 88% is obtained on the same data. Therefore, there is no way to determine these values other than using trial and error method.

3. **Is the input size important?**

   In our experiments, it is observed CPN performance is not affected by the input size in the sense, it has shown similar performance for all the input sizes 20×20, 32×32 and 64×64.

4. **Initial weights and their importance in the training process**

   As presented in the previous chapter, CPN is performing better if the initial weights are assigned using class prototypes than if they are assigned random values (0 or 1).

5. **Training set size and its impact on the performance**

   To determine, whether training set size affects the performance of CPN or not, needs more experimentation.

6. **Features**

   In all of our experiments, we have used only image pixel values as features. More sophisticated features have yet to be experimented.

6.2 Summary

From the above observations, it can be concluded that choosing learning parameter values is difficult and empirical. The impact of training set size and different set of features on the performance of CPN needs to be investigated.

Chapter 7

Conclusion

7.1 Conclusion

In the present work, we have investigated CPN architecture for large number of classes of Telugu. In our experimentation, we have observed that the naive CPN architecture is not sufficient to handle large number classes and there is a need to adapt its architecture according to the problem complexity.

7.2 Future work

In future, architecture of CPN shall be adapted in such a way that it would be able to handle 2D inputs with more sophisticated features.

Bibliography

[1] Muhammad Faisal Zafar,Dzulkifli Mohamad, and Razib M. Othman : *On-line Handwritten Character Recognition: An Implementation of Counterpropagation Neural Net*, World Academy of Sciences, Engineering Technology (10 2005) .

[2] Anoop M. Namboodiri, Anil K. Jain.(2004) : *Online Handwritten Script Recognition*, IEEE Trans. PAMI. 26(1) 124–130.

[3] Bishop M. (1995) :*Neural Networks for Pattern Recognition*, Oxford Univ. Press, Oxford-U.K .

[4] Ahmed,S.M.,et.al, (Nov.1995). *A survey on offline Cursive Word Recognition.*Pattern recognition, 35: 1433.

[5] Rumelhart, D.E.and McClelland, J.L.(1986-1987). *Parallel distributed processing: Explorations in the microstructure of cognition*, Volume 1, 2, 3. Cambridge, MA: MIT Press.

[6] Kohonen, T.(1984). *Self-organization and associative memory*, New York: Springler-Verlag.

[7] LeCun Y., Bottou L., Bengio Y., and Haffner P. (1998b). *Gradient-Based Laerning Applied to Document Recognition*, Proc. IEEE, 86(11): 2278-2324.

[8] Hecht-Nielson, R.(1987). *Counterpropagation networks*, Proc. Of IEEE First Int'1 Conference on Neural Networks. 1987 II: 19-32.

[9] LeCun Y., Bengio Y. *Pattern recognition and Neural networks in the Handbook of Brain Theory and Neural Networks*, MIT Press(1995).

[10] Jacek M. Zurada : *Introduction to Artificial Neural Systems*,10th edition(2006),West Pubishing Company.

[11] B. Yagnanarayan. *Artificial Neural networks*, PHI Learning Private Limited(1999).

[12] Velappa Ganapathy, and Kok Leong Liew. *Handwritten Character Recognition Using Multiscale Neural Network Training Technique*, World Academy of Sciences, Engineering and Technology 39(2008).

[13] Hecht-Nielson, R.(1988). *Application of Counterpropagation networks*, Neural Network, 1, 131-139.

[14] T. Mitchell. *Machine Learning*, McGraw Hill, 1997.

[15] R. Nagasri. *Developing Features for Telugu OCR Using Convolutional Neural networks*, Masters Thesis in Univ. Of Hyderabad(2007).

[16] Karsten Kutza *http://www.neural-networks-at-your-fingertips.com/cpn.html*, Counterpropagation Network Simulator(1996).