Odia Offline Character Recognition

Priyaranjan Behera
108CS021
B.Tech.(CSE), 2008-12

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

Odia Offline Character Recognition

Thesis submitted on

May, 2012

in partial fulfilment of the

requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Priyaranjan Behera

(Roll: 108CS021)

under the guidance of

Prof. Banshidhar Majhi

NIT Rourkela

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela-769 008, Odisha, India

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

May 14, 2012

Certificate

This is to certify that the work in the thesis entitled ***Odia Offline Character Recognition*** by ***Priyaranjan Behera*** is a record of an original research work carried out under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Banshidhar Majhi
Professor
CSE Department of NIT Rourkela

Acknowledgement

I am indebted to my mentor, Prof. Banshidhar Majhi for giving me an opportunity to work under his guidance. I am really thankful for his expert guidance and motivation during the course of the project.

I am also grateful to Prof. Ramesh Kumar Mohapatra for his constant support and timely suggestions, without which this project could not have seen the light of the day.

I convey my regards to all the other faculty members of Department of Computer Science and Engineering, NIT Rourkela for their valuable guidance and advices at appropriate times.

Finally, I would like to express my gratitude to my parents, brother and friends for their constant support and motivation throughout this project.

Priyaranjan Behera

Abstract

Optical Character Recognition (OCR) is a document image analysis method where scanned digital image that contains either machine printed or handwritten script are input into a system to translate it into an editable machine readable digital text format. Development of OCRs for Indian script is an active area of research today. This is an attempt towards making an OCR system for the Odia script. Odia language present great challenges to an OCR designer due to the large number of letters in the alphabet, the sophisticated ways in which they combine and many characters being roundish and similar in looks. The Biju Patnaik Central Library at NIT Rourkela is making a lot of effort in preserving Odia books. But, the space requirement is huge which can be compressed by the use of an Odia OCR.

In this project, an attempt is made to recognize the Odia characters by the use of the gradient features of a character image. The features extracted are then classified with the help of an Artificial Neural Network. Further, a method of nesting of Artificial Neural Networks is proposed for greater accuracy of the recognition algorithm and a faster numeral recognition algorithm for Odia numerals by using parts of the character instead of the whole image.

Keywords: Odia Script; Character Recognition; Gradient; Artificial Neural Network; Nested ANN; Multi-layer Perceptron; Backpropagation.

Contents

Certificate ii

Acknowledgement iii

Abstract iv

List of Figures vii

List of Tables viii

1 Introduction 1

1.1 Steps in an OCR System 2

1.1.1 Pre-Processing 2

1.1.2 Feature Extraction 2

1.1.3 Classification 3

1.2 Odia Script and Challenges Faced 3

1.3 Objective 5

1.4 Thesis Organization 5

2 Literature Survey 6

2.1 Kohonen Neural Network 6

2.2 Structural Feature Based 7

2.3 Direction Chain Coding 8

2.4 Gradient Based Features 9

2.5 ANN vs. Quadratic Classifier 9

3 Proposed Methodology 10

3.1 Feature Extraction 10

3.1.1 Steps Undertaken for Feature Extraction 10

3.1.2 Normalization 11
3.1.3 Robert's Cross Operator 11
3.1.4 Gaussian Reduction 12
3.1.5 Feature Extraction Flow Diagram 13
3.2 Classification 14
3.2.1 Artificial Neural Network 14
3.2.2 Multilayer Feedforward Network 15
3.2.3 Nesting Of Artificial Neural Network 16
3.2.4 Faster Odia Numeral Recognition 16

4 Implementation and Results **17**

4.1 Dataset 17
4.2 Training 19
4.3 Results 21

5 Conclusions and Future Work **26**

5.1 Conclusions 26
5.2 Future Work 27

Bibliography **28**

List of Figures

1.1 Block Diagram of OCR System 2
1.2 Odia Vowels 3
1.3 Odia Consonants 3
1.4 Odia Modified Characters 4
1.5 Odia Compound Characters 4
1.6 Similar Shaped Characters 4

2.1 Decision Tree for Structural Based Recognition 7
2.2 Chain Coding Directions 8

3.1 Robert's Cross Convolution Kernel 11
3.2 Image Pyramid 12
3.3 Feature Extraction Flow Diagram 13
3.4 Neural Network Overview 14
3.5 Multilayer Feedforward Network 15
3.6 One-Third Odia Numeral 16

4.1 Datasets used for Experiments 18
4.2 Data Collection Form 18
4.3 Comparison between Size of Hidden Layers 19
4.4 Neural Network Training 20
4.5 Confusion Matrix Example 21
4.6 Confusion Matrix of Single Network for MNIST Dataset 22
4.7 Confusion Matrices of Nested Network for MNIST Dataset 23
4.8 Confusion Matrices of Nested Network for Odia Dataset 24
4.9 Confusion Matrices of Nested Network for Odia Dataset 24

List of Tables

4.1 Comparison of Different Methods 22
4.2 Accuracy Increase by Nesting for MNIST Dataset 23
4.3 Accuracy Increase by Nesting for Odia Dataset 25
4.4 Performance of Faster Odia Numeral Recognition 25

Chapter 1

Introduction

Optical Character Recognition (OCR) is a process of automatic recognition of characters from optically scanned and digitized pages. It can contribute immensely to the advancement of the automation process and can improve the man-machine interface in many applications.
Some practical application potentials of OCR system are:

- Reading aid for the blind,
- Automatic text entry into the computer for desktop publication, library cataloging, ledgering, etc.
- Automatic reading for sorting of postal mail, bank cheques and other documents,
- Document data compression: from document image to ASCII format, etc.

Traditionally, OCR techniques are classified into:

- **Template based and**
- **Feature based approach** .

In template based approach, an unknown pattern is superimposed on the ideal template pattern and the degree of correlation between the two is used for the decision about the classification. Early OCR systems employed only template approach. But they become ineffective in the presence of noises, changes of handwriting, etc. Modern systems thus combine it with feature based approaches to obtain better results.

Feature-based approach derives important properties (features) from the test patterns and employs them in a more sophisticated classification model.
Any Feature-Based Character Recognition System constitutes:

- **Preprocessing**,
- **Feature Extraction** and
- **Classification**.

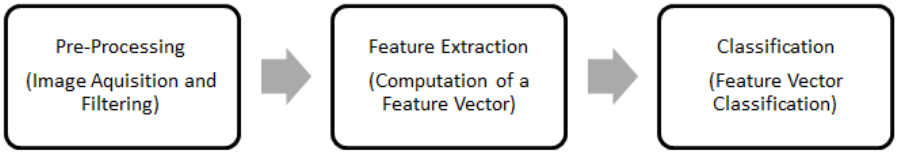

Figure 1.1: Block Diagram of OCR System

1.1 Steps in an OCR System

1.1.1 Pre-Processing

Before any recognition techniques is used the acquired document images have to go through a series of steps to make them suitable for recognition. Some of the pre-processing methods include:

- Filtering,
- Binarization,
- Skew Correction,
- Segmentation, etc.

1.1.2 Feature Extraction

In this step image features at various levels of complexity are being extracted from the image. Typical examples of such features are:

- Lines, edges and ridges,

- Localized interest points like corners, blob, points, etc.
- Complex features like texture, gradient, curvature, shape, etc.

1.1.3 Classification

This step uses the extracted features from the previous step to identify the characters. It is the problem of identifying to which set a particular new observation belongs based on the fact extracted from a training set.

1.2 Odia Script and Challenges Faced

Even though there has been rapid advancement in many of the foreign languages and some of the Indian scripts, research on Odia has been lagging behind. The problem statement of this project is nothing but the different properties of the Odia scripts and how they can be tackled.

Odia is mainly used in the state of Odisha in India[3]. The Odia script is developed from the Kalinga script which is a descendent of the Brahmi script of Ancient India. The modern Odia script has around 11 vowels and 41 consonants. These are called the basic characters.

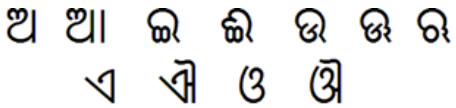

Figure 1.2: Odia Vowels

Figure 1.3: Odia Consonants

Writing of the script is from left to right and the concept of upper and lower cases is absent. Most of the Odia characters are roundish in structure and very similar to

each other in looks. This makes the classification of the characters even more difficult. There is no horizontal line (Shirarekha) as those present in Bengali and Hindi script which makes the segmentation task a little easier.

When a vowel follows a consonant takes a modified shape. Depending on vowel, the modified shape is positioned left, right, both left and right, or bottom of the character. The modified shapes are called Matras.

	ଅ a	ଆ ā	ଇ i	ଈ ī	ଉ u	ଊ ū	ଋ ṛ	ଏ e	ଐ ai	ଓ o	ଔ au
କ୍ k	କ	କା	କି	କୀ	କୁ	କୂ	କୃ	କେ	କୈ	କୋ	କୌ
ଖ୍ kh	ଖ	ଖା	ଖି	ଖୀ	ଖୁ	ଖୂ	ଖୃ	ଖେ	ଖୈ	ଖୋ	ଖୌ
ଗ୍ g	ଗ	ଗା	ଗି	ଗୀ	ଗୁ	ଗୂ	ଗୃ	ଗେ	ଗୈ	ଗୋ	ଗୌ
ଘ୍ gh	ଘ	ଘା	ଘି	ଘୀ	ଘୁ	ଘୂ	ଘୃ	ଘେ	ଘୈ	ଘୋ	ଘୌ

Figure 1.4: Odia Modified Characters

And when a consonant or vowel follows a consonant sometimes takes a compound shape called compound character (Juktakshara).

Figure 1.5: Odia Compound Characters

There are nearly 200 compound characters and thus summing up a total of nearly 300 individual classes to be classified are present.

One of the most challenging facts about the Odia script is the presence of the similarity in shape. Some of the similar shaped characters are as in the Fig: 1.6.

ଶ ଣ	ଉ ଊ ଭ	ଡ ତ
ଳ ଲ	ଚ ର	କ ଜ

Figure 1.6: Similar Shaped Characters

1.3 Objective

The objective of the project is to develop a Robust Odia Offline Character Recognition algorithm which can be adaptable in the situations of high dimensional spaces and similarity in the characters. This project can then be extended to build a new system by integrating it with different preprocessing steps.

1.4 Thesis Organization

This entire thesis is divided into 5 Chapters. While Chapter 1 was the Introduction, Chapter 2 is the Literature Survey that is focused on the different works that has already been undertaken in the field of Odia character recognition. Chapter 3 is about the approached that has been proposed in this project and several new ideas that has been introduced. The implementation details and the results that have been obtained through the proposed framework with comparison with the existing algorithms are presented in Chapter 4. Chapter 5 concludes the thesis with a special focus on the future work that can be undertaken.

Chapter 2

Literature Survey

A very little amount of work has been undertaken till date on the Odia character recognition field as this research domain for Odia script is still in its nascent days. Some of the noteworthy methods are:

- Kohonen Neural Network,
- Direction Chain Code Features,
- Structural Features,
- Stroke and Run-Number Based Features and
- Features Extracted from Water Overflow from a Reservoir Method.

2.1 Kohonen Neural Network

Mohanty [4] proposed a system to identify the characters using Kohonen neural network. In this technique instead of specific features the input pixels were fed into the Kohonen network to get the output in terms of a weighted sum formula. The class for which the character has the least function values is allocated with the character. But this system was implemented only on 5 different classes and thus the reliability of the system is uncertain.

2.2 Structural Feature Based

Chaudhuri et al. [5] proposed the use of different structures present in the Odia characters to distinguish between them. From the bounding box of the character image structural features are extracted with the help of concepts like stroke and run-number based feature extraction, water overflow from a reservoir, etc.

Some of the structural features include [2] :

- Upper Part Circular,
- A vertical line on the Right-most Part,
- Horizontal Run code,
- Vertical Run codes,
- Number of Holes and
- Position of Holes.

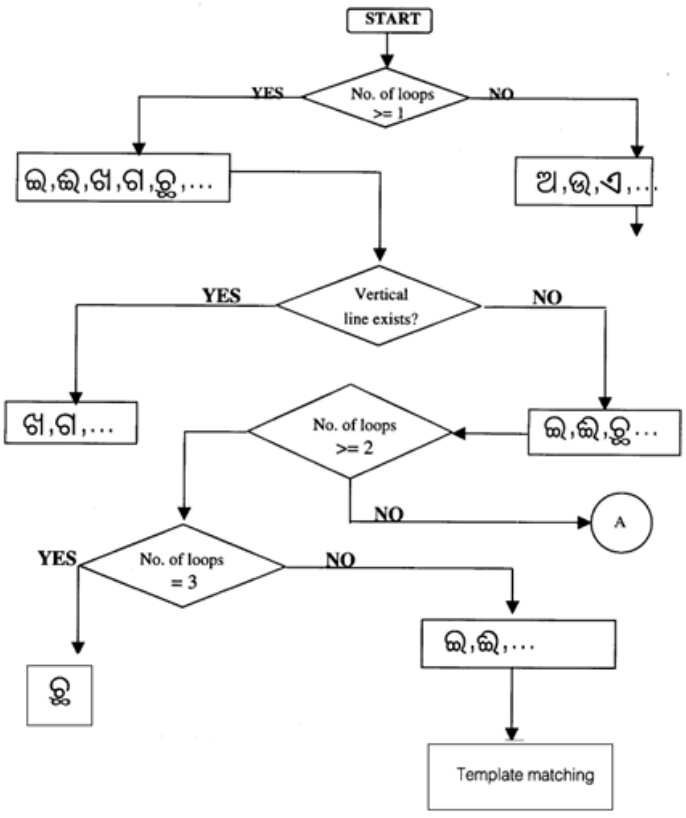

Figure 2.1: Decision Tree for Structural Based Recognition

For the classification, a decision tree based tree classifier is used. In this approach the characters are firstly grouped based on the structural similarity and then are classified based on the feature vectors. But owing to the similarity in the structures of the characters the accuracy of the system is not good enough.

2.3 Direction Chain Coding

Pal et al. [6] proposed is a fast character recognition system which consists of only 64 features. Firstly, the contour pixels of the characters are identified. This is done by the fact that if any one of the eight surrounding point of an object pixel is a background pixel then the pixel is a contour pixel. Then character image is divided into 7x7 blocks and separate histograms based on the 4 directions are calculated for each block based on the direction chain code of each contour pixel to constitute the features of the character image.

The different directions are identified by Fig: 2.2.

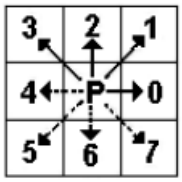

Figure 2.2: Chain Coding Directions

Here chain code of four directions only are used [directions 0 (horizontal), 1 (45 degree slanted), 2 (vertical) and 3 (135 degree slanted)]. It is assumed that the chain code of direction 0 and 4, 1 and 5, 2 and 6, 3 and 7, are same. Thus, for 7×7 blocks there are $7 \times 7 \times 4 = 196$ features. To reduce the feature dimension, after the histogram calculation in 7×7 blocks, the blocks are down sampled into 4×4 blocks by a Gaussian filter. As a result $4 \times 4 \times 4 = 64$ features for recognition are derived. These feature vectors are then fed to a quadratic classifier for classification.

2.4 Gradient Based Features

Pal et al. [6] [7] further proposed a 400 dimensional feature based recognition system where they divided the bounding box of the grayscale character image into 9×9 blocks and then found the directional features of the gradient of the image along 16 directions in each block. Thus, for the 9×9 blocks there are $9 \times 9 \times 16 = 1296$ features. The features are reduced with the help of a Gaussian filter to obtain $5 \times 5 \times 16 = 400$ features. Then a quadratic classifier was used for classification. This gave a significantly high accuracy than the other mentioned methods.

2.5 ANN vs. Quadratic Classifier

While most of the surveyed methods used quadratic classifier, Hamamoto et al. [8] appreciated the accuracy of the Artificial Neural Network as compared to Quadratic classifier when it comes to classification in high dimensional spaces. Liu et al. [9] echoed the same facts while prophesizing that a combination of multiple classifiers can be helpful to remove ambiguities. Thus based on the above findings we went for an approach with Gradient as feature and Artificial Neural Network as classifier. Further we mull over the use of an ensemble of ANN for better classification.

Chapter 3

Proposed Methodology

In my proposed methodology, the gradient properties at each pixel of the character image and multi-layered perceptron with backpropagation training are used as feature and classifier respectively.

3.1 Feature Extraction

The features used in the classifier are obtained from the directional information of the characters. For feature computation, the bounding box of a numeral is segmented into blocks and gradient in each direction [7] are computed for each of the blocks. These blocks are then down sampled by a Gaussian filter.

3.1.1 Steps Undertaken for Feature Extraction

Step 1: Convert the binary image of the character into grayscale image by applying mean filter on the image for 5 times.

Step 2: The image is then normalized so as to have zero mean and unit standard deviations.

Step 3: The direction of gradient is quantized to 16 levels with $\frac{\pi}{16}$ intervals.

Step 4: The strength of the gradient is accumulated in each of the 16 directions to get 9x9 local joint spectra of direction.

Step 5: So a total of $9 \times 9 \times 16 = 1296$ features are present. Now a Gaussian-Filter [1 4 6 4 1] is used to make spatial resolution and get $5 \times 5 \times 16 = 400$ features.

Step 6: This 400 dimensional feature vector is fed to a classifier.

3.1.2 Normalization

The normalization is done so as to have a zero mean and unit standard deviation. This is done to standardize all the input images and to have an independence from the size of the input image. Moreover, the ANNs are believed to work better if the feature values lie between -1 and 1. This is done by the following calculations.
To have a zero mean for each pixel in the image,

$$I'(x,y) = I(x,y) - \sum_{i=0}^{m}\sum_{j=0}^{n} I(i,j) \tag{3.1}$$

Where, $I(x,y)$ is the intensity value of the pixel at (x,y) and $I(x,y)$ is the new intensity values.
For unit standard deviation,

$$I''(x,y) = \frac{I'(x,y)}{\sigma} \tag{3.2}$$

Where, $I(x,y)$ is the new intensity value and σ is the standard deviation of the image $I(x,y)$.

3.1.3 Robert's Cross Operator

The operator consists of a pair of 2X2 convolution kernels as shown. One kernel is simply the other rotated by 90°. These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations [10].
Let I(x,y) be a point in the original image and G_x (x,y) be a point in an image formed by convolving with the first kernel and G_y(x,y) with second one.

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

Figure 3.1: Robert's Cross Convolution Kernel

G_x and G_y can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The Gradient at a pixel

is given by:

$$\Delta I(x,y) = G(x,y) = \sqrt{(G_x)^2 + (G_y)^2} \tag{3.3}$$

And the direction of the gradient is given by:

$$\Theta(x,y) = \arctan\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \tag{3.4}$$

Since the return value of arctan is in the range of $[\frac{-\pi}{2}, \frac{\pi}{2}]$ and when quantized into 16 directions we get an interval of $\frac{\pi}{16}$.

3.1.4 Gaussian Reduction

This idea was derived from the concept of Gaussian pyramid where the image pyramid is a collection of images with the base containing a high resolution image and as we move upward the resolution as well as the size of the image decrease. While the base level has a size of $S^J \times S^J$ or $N \times N$, where $= \log_2 N$, apex level is of size 1x1. Any general level j has a size $2^j \times 2^j$ [10].

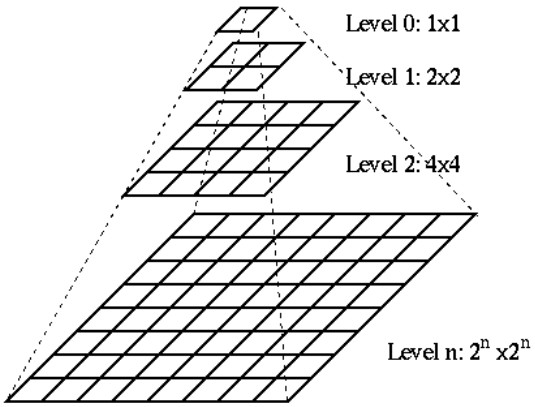

Figure 3.2: Image Pyramid

Gaussian pyramid is a hierarchy of low-pass filtered versions of original image. The low passing is done using convolution with a Gaussian filter kernel. The Gaussian pyramid is defined recursively as:

$$G_l(x,y) = \begin{cases} I(x,y) & \text{if } l = 0 \\ \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) G_{l-i}(2x+m, 2y+n) & \text{if } l > 0 \end{cases}$$

Where $w(m,n)$ is the weight function which closely approximates the Gaussian function and is given by:

$$w(m,n) = \frac{1}{16} \cdot [\ 1\ 4\ 6\ 4\ 1]$$

Thus to reduce the 3-Dimensional feature vector we use the formula,

$$G_l(x,y,z) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) G_{l-i}(2x+m, 2y+n, z) \tag{3.5}$$

Where, i and j are the pixels points on the image and k is the direction index.
After the feature extraction of all the images, say a total of N images are arranged in a Nx400 matrix, where each row constitutes the features of the character corresponding the that particular row.

3.1.5 Feature Extraction Flow Diagram

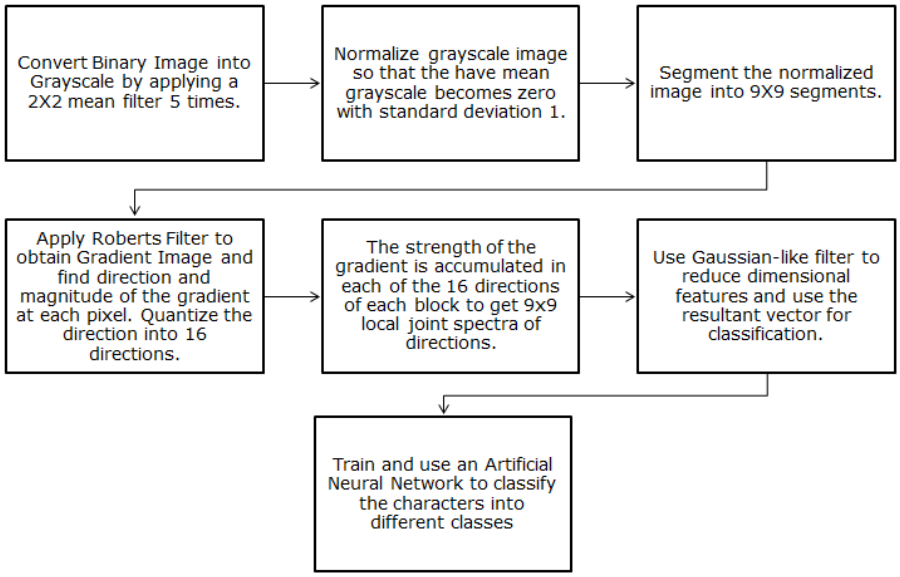

Figure 3.3: Feature Extraction Flow Diagram

3.2 Classification

The features extracted are then classified with the help of an Artificial Neural Network.

3.2.1 Artificial Neural Network

Neural Networks are the simple imitations of the human nervous systems and has been motivated by the kind of computing done by the human brain. The structural constituents of the brain are termed as the neurons which perform computations such as cognition, logical inference, pattern recognition, etc. and the technology built to imitate this computation is known as Artificial Neural Network [11].

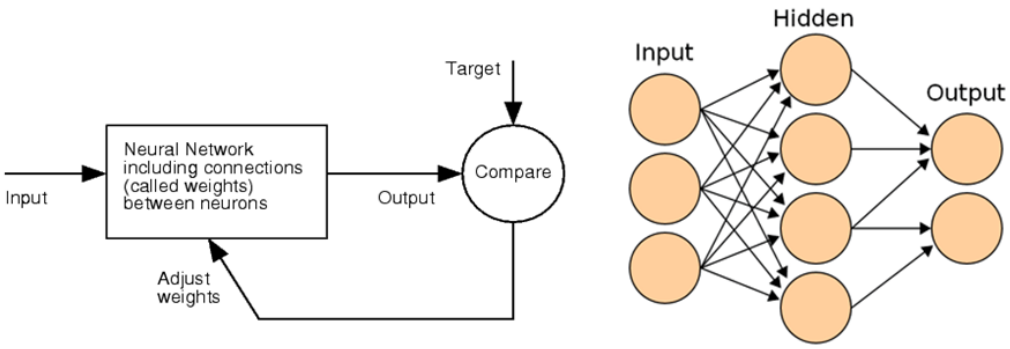

Figure 3.4: Neural Network Overview

So, basically Neural Networks are composed of simple elements (neurons) operating in parallel. They can be trained by adjusting the values of the connections (weights) between elements. In training the networks are adjusted or trained so that a particular input leads to a specific target output.
The learning methods in Neural Networks are broadly classified into:

- **Supervised Learning** - In this, every input pattern that is used to train the network is associated with an output pattern which is the target. A teacher is assumed to be present during the learning process, when a comparison is made between the network's computed output and the correct expected output to determine the error and use the error to adjust the weights of the network.

- **Unsupervised Learning** - In this method, the target output is not presented

to the network and no teacher is present. Thus, the system learns of its own by discovering and adapting to the input patterns.

As prior ideas about the target is known, supervised learning used in this approach.

3.2.2 Multilayer Feedforward Network

In addition to the input set of neurons (input layer) to receive the input signals and output neurons (output layer) to provide the output, there are one or more intermediary layers (hidden layers) in this network. They perform useful intermediary computation before directing the input to the output layers.

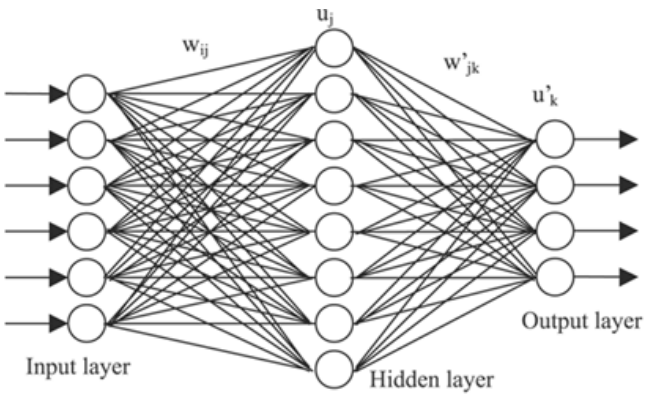

Figure 3.5: Multilayer Feedforward Network

To the Multilayer Feedforward Network we employ Backpropagation Learning method. **Backpropagation** is a systematic method of training multilayer artificial neural network. It is a supervised learning algorithm consisting of the following two steps:

- **Propagation:**
  1. Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
  2. Calculate the error The difference between the network output and the target output
- **Weight Update:**
  1. Error signals propagate backwards through the network where they are used to adjust the weights.

The training algorithm continues till we get a satisfactory level of low error or it exceeds a certain number of pre-determined iterations.

3.2.3 Nesting Of Artificial Neural Network

One of the main challenges in the Odia script is the similarity of the characters. To counter with this problem a method of nesting the artificial neural network is proposed. In this method similar looking characters are grouped based on the errors detected in the confusion matrix if only a single neural network is taken for classification.

Then root neural network with the similar characters treated as a single class is operated. After that nested NN to classify between the similar looking characters is used. Similarly this technique can be used for multiple levels for better results.

3.2.4 Faster Odia Numeral Recognition

There are many cases where faster detection of characters is needed. Detection on postal packages, bank cheques are to name a few. To accomplish this, a faster Odia numeral recognition method is proposed. In this method instead of the extracting the feature of the whole character just the feature of the top one-third of the character is extracted. This method is ideal to be implemented in the case of printed characters. As Odia numerals have unique looking top parts, this procedure can have faster recognition and considerable accuracy.

Figure 3.6: One-Third Odia Numeral

Chapter 4

Implementation and Results

The experiments were performed on various datasets with the help of *MATLAB*.

4.1 Dataset

The algorithms were tested on three different types of datasets. They are English high quality fonts [12], low quality English handwritten characters [13] and Odia handwritten characters. While the former two sets were used to compare the effectiveness of the algorithm with respect to other algorithms. The third set justifies the effectiveness on the Odia script.

- High Quality English Font
  - English Numbers and Alphabets
  - Size - 128×128
- MNIST Handwritten Digit Database
  - Low Quality Digits
  - Size - 28×28
- Odia Handwritten Digit Database
  - Medium Quality Digits
  - Size - 70×70 (approx.)

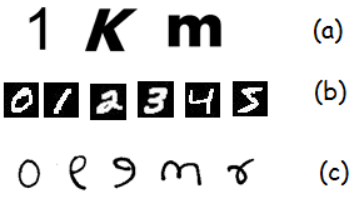

Figure 4.1: Datasets - (a)High Quality English Fonts, (b)Low Quality English Digits, (c)Medium Quality Odia Digits

Due to the unavailability of any public Odia character dataset, this dataset is developed at IPCC Lab, NIT Rourkela. A form with blocks where characters are to be written was developed which was filled by 20 different students. Then each character was segmented manually to make the dataset.

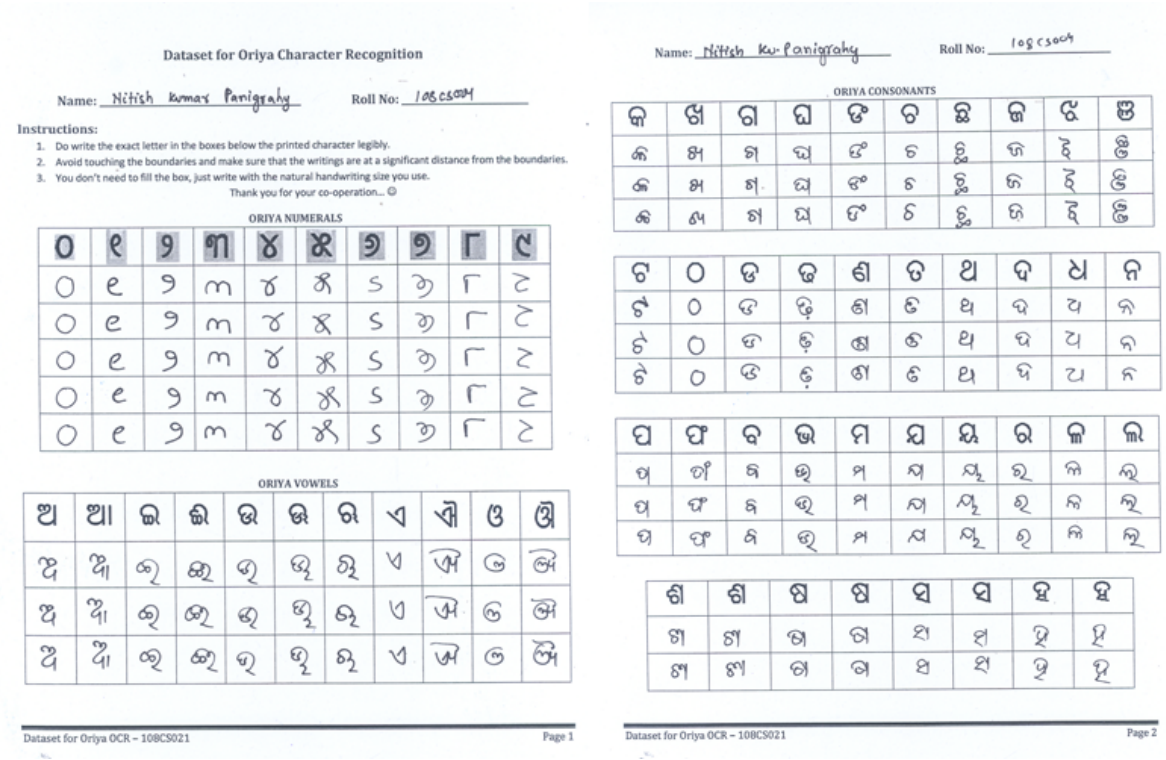

Figure 4.2: Data Collection Form

After the dataset has been collected the features of each character image are calculated as per the feature extracted steps and assembled in an $N \times 400$ matrix

where N is the number of character images taken and used for the training purposes.

4.2 Training

The Matlab Neural Network Toolbox [14] *nprtool* has been used for the construction of the neural network which uses *nntraintool* for the training of the neural networks.

Size of the Neural Layers:

- **Input Layer** Since there is a total of 400 features, the size of the input layers has to be taken as 400.
- **Output Layer** The output layer size is the number of classes to be classified into as each output bit is allocated to a corresponding output class. For experiment, since 10 classes are taken, the output layer size is 10.
- **Hidden Layer** - Ideally, the number of hidden layers should be between L and 2L where L is the size of the output layer. As we take the output layer size to be 10, the hidden layer size should be between 10 and 20 [11]. Thus, the accuracy obtained for each output layer size is taken and plotted to get Fig: 4.3.

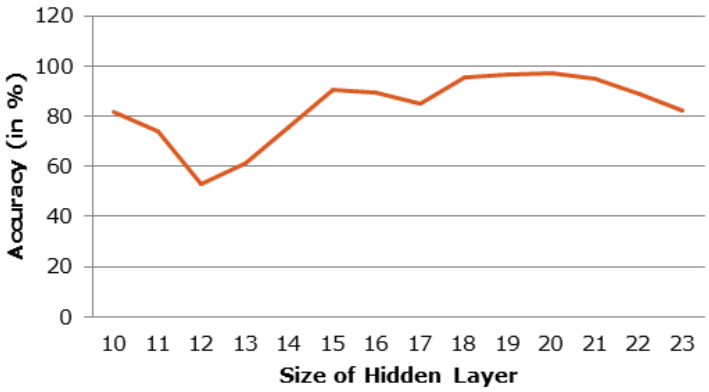

Figure 4.3: Comparison between Size of Hidden Layers

The highest accuracy is obtained when the hidden layer size is 20. Thus, in all the following methods, the hidden layer size is taken to be 20.

The entire feature set is divided into:

- Training - 70%

- Testing - 15%
- Validation - 15%

While Testing set is used to have a complete independent test of whether the network is generalizing or not, Validation is used to see of the training process is not overfitting the network.

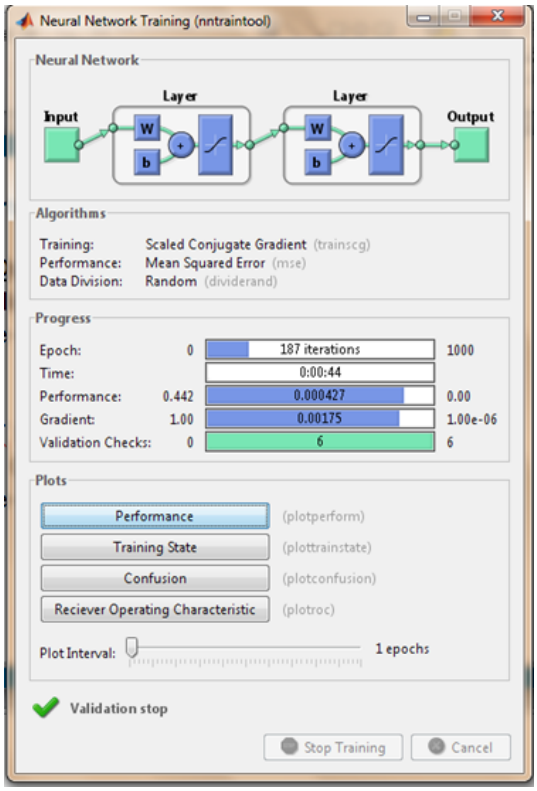

Figure 4.4: Neural Network Training

The performance of the network is measured by the mean square error given by:

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2 \tag{4.1}$$

Where t_i is the target and a_i is the output of the output neuron i.

In each step the weights of the connections are adjusted by using the formula:

$$x_{k+1} = x_k - \alpha_k g_k \tag{4.2}$$

Where α_k is learning factor, g_k is the gradient and x_{k+1} and x_k are new and previous weights.

The training methodology being used in Scaled Conjugate Gradient Method which is a faster training algorithm and where the direction is chosen in such a way that it doesn't cancel out the previous moves.

The training of the neural network stops when the gradient reaches a value of 10-6 or the number of consecutive validation checks reaches 6 i.e., there is no good improvement of the gradient for six continuous iteration measured by the use of the validation set of features [14].

4.3 Results

The accuracy of a character recognition algorithm is measured by the use of confusion matrix as in 4.5 [14].

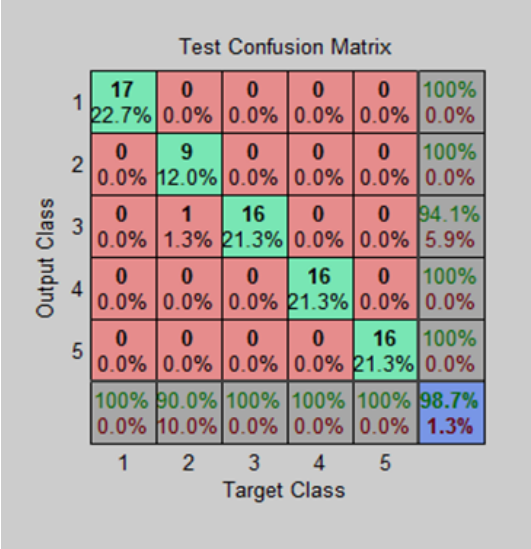

Figure 4.5: Confusion Matrix Example

The element $I(x, y)$ represents the number of elements with target class y and has been classified as x. Thus, the diagonal elements represent the correctly recognized characters and other block shows the misclassified ones. The overall accuracy can be calculated by dividing the sum of the diagonal elements with the total number samples taken.

The accuracy and the time efficiency of the different implemented methods are as in the Table: 4.1.

Table 4.1: Comparison of Different Methods

Algorithm and Dataset	**Time to Extract Features (in s)**	**Time to Train (in s)**	**Number of Iterations**	**Accuracy (in %)**
Chain Coding for MNIST (Classes-10, Samples-100)	15.76	17	89	91.7
Chain Coding for Eng Font (Classes10, Samples-100)	55.25	29	142	99.6
Gradient for MNIST (Classes10, Samples-100)	51.98	36	151	95.9
Gradient for Eng Font (Classes10, Samples-100)	60.12	20	93	99.5
Chain Coding for Eng Font (Classes-30, Samples-100)	218	321	183	97.83
Gradient for Eng Font (Classes30, Samples-100)	334.46	403	183	97.86
Chain Coding for Odia Chars (Classes10, Samples-15)	2.96	8	64	88.7
Gradient for Odia Chars (Classes10, Samples-15)	3.02	3	44	91.33

The confusion matrix obtained when the Gradient method is applied to the MNIST dataset is as in the Fig: 4.6.

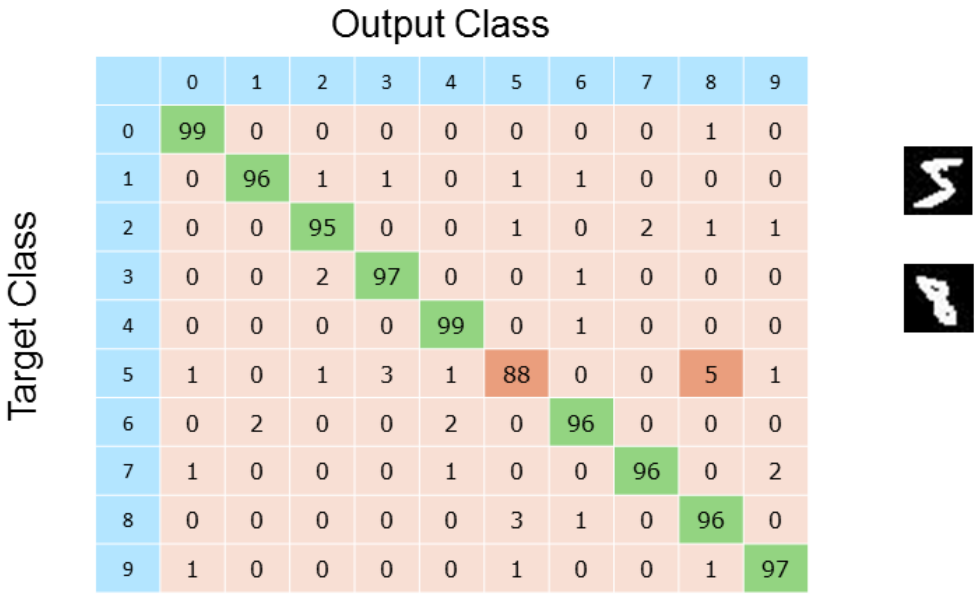

Figure 4.6: Confusion Matrix of Single Network for MNIST Dataset

It is found that the class 5 has been misclassified as 8 in 5 different occasions. Now the nesting of the neural networks is done to get the confusion matrices as in Fig: 4.7.

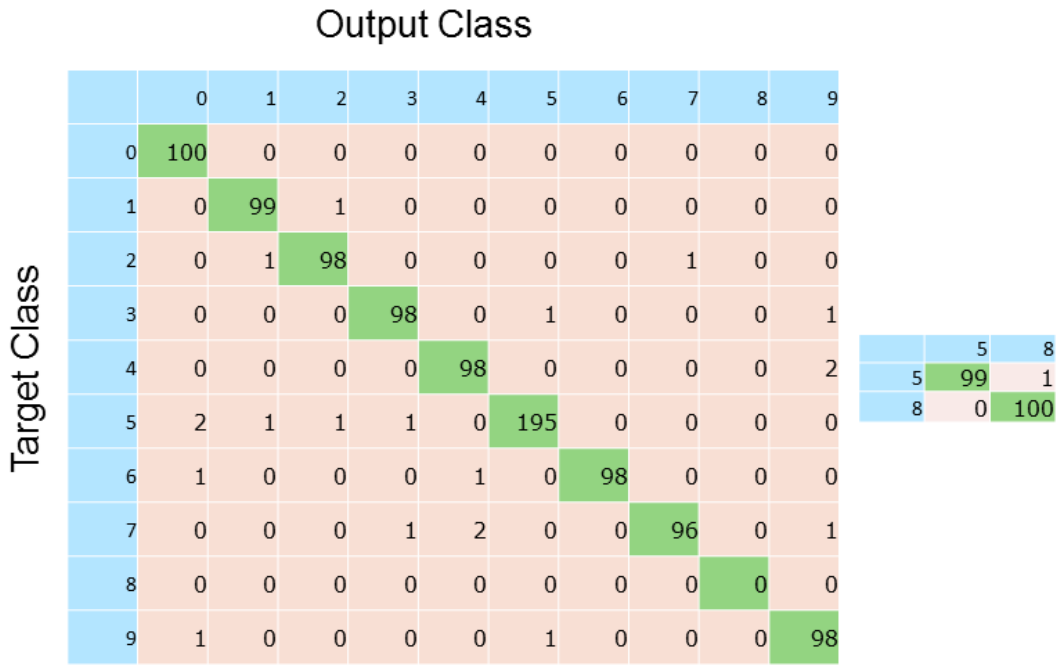

Figure 4.7: Confusion Matrices of Nested Network for MNIST Dataset

Thus from the confusion matrices the accuracy has been inferred as in Table: 4.2.

Table 4.2: Accuracy Increase by Nesting for MNIST Dataset

Methods	**Accuracy (in %)**
Gradient for MNIST without Nesting	95.9
Gradient for MNIST with Nesting	98.4

Similarly for the Odia numerical this method is used to get the initial confusion matrix as in Fig: 4.8

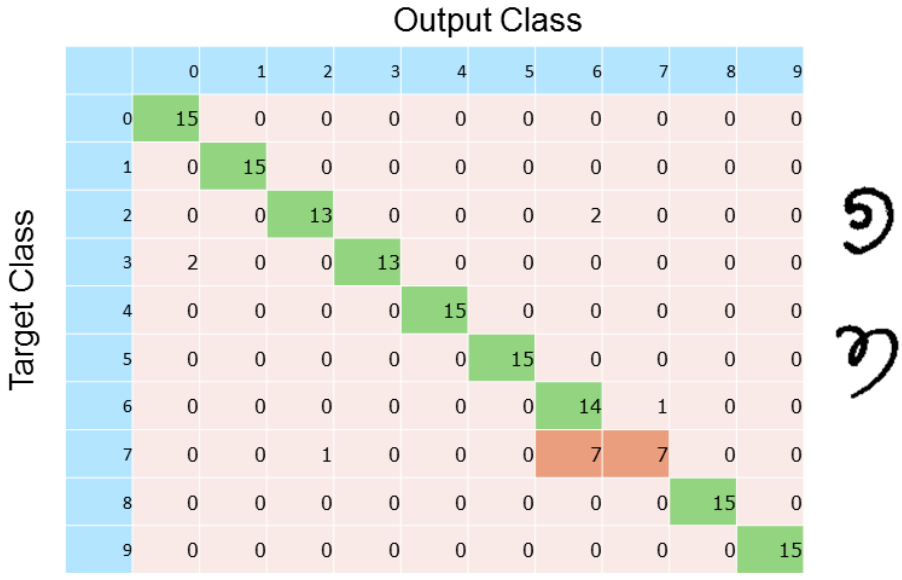

Figure 4.8: Confusion Matrices of Nested Network for Odia Dataset

It is seen that the number 7 has been misclassified as 6 in 7 occasions. Now nesting of the neural networks is done to get the confusion matrices as in Fig: 4.9.

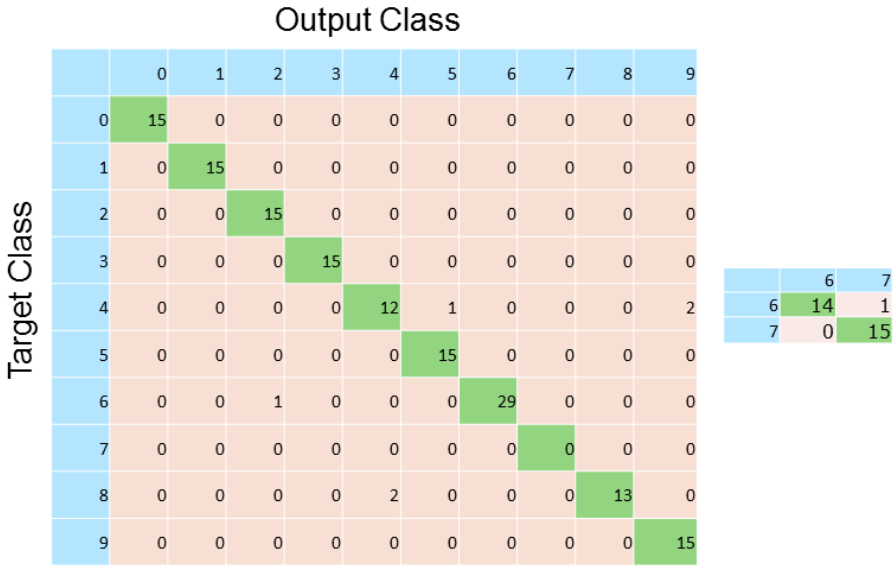

Figure 4.9: Confusion Matrices of Nested Network for Odia Dataset

Thus from the confusion matrices the accuracy has been inferred as in Table: 4.3.

Table 4.3: Accuracy Increase by Nesting for Odia Dataset

Methods	**Accuracy (in %)**
Gradient for Odia without Nesting	91.33
Gradient for Odia with Nesting	95.33

Furthermore, when the faster Odia numerical recognition method is used where we take only the top one-third of the image we get the results as in the Table: 4.4

Table 4.4: Performance of Faster Odia Numeral Recognition

Methods	**Times to Extract (in s)**	**Accuracy (in %)**
Gradient for Odia - Whole Character	3.12	91.33
Gradient for Odia - One-Third Character	2.13	87.33

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The thesis approaches towards developing a better character recognition algorithm for Odia script. This can help to make an improved man-machine interface with the help of a native language. But as mentioned in Chapter 1, the Odia script is very challenging due to the presence of a large number of characters and with the similarity between them.

Thus a new approach is proposed in Chapter 3 based on all the problems that can be faced which includes Gradient Information as features and Artificial Neural Network as a classifier which was based on several findings like the higher efficiency of the gradient features and the better working of the ANN in higher dimensional spaces. The different features were analyzed in Chapter 4 to infer that gradient is a culpable option .

Further, an approach where nesting of the ANNs is proposed in Section 3.2.3 which is very useful in cases of similar characters. This showed a noteworthy increase in the classification accuracy and showed a lot of promise to classify and larger dimensional space with similar classes.

A faster numerical recognition algorithm was proposed in Section 3.2.4 which in cases of printed Odia characters can show a appreciable accuracy taking lesser amount of time.

5.2 Future Work

Recognition algorithm of satisfying accuracy has been developed for simple Odia characters. Further, the algorithm can be extended to include modified and compound characters.

Before any extraction and recognition part the preprocessing of the images has to be done which was taken as a constraint in this thesis.

Some major preprocessing includes:

- Binarization
- Skew Correction
- Segmentation

Finally, this method can be extended to recognize a group of characters by using character segmentation and then to build a complete system to convert a whole page of data into its compressed ASCII format.

Bibliography

[1] Y. Fujisawa, Meng Shi, T. Wakabayashi, and F. Kimura. Handwritten numeral recognition using gradient and curvature of gray scale image. In *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 277 –280, sep 1999.

[2] S. Mishra, D. Nanda, and S. Mohanty. Oriya character recognition using neural networks. *International Journal of Computer and Communication Technology*, 2(2,3,4):88 –92, December 2010.

[3] U. Pal and B. B. Chaudhuri. Indian script character recognition: a survey. *Pattern Recognition*, 37(9):1887–1899, 2004.

[4] Sanghamitra Mohanty. Pattern recognition in alphabets of oriya language using kohonen neural network. *IJPRAI*, 12(7):1007–1015, 1998.

[5] B.B. Chaudhuri, U. Pal, and M. Mitra. Automatic recognition of printed oriya script. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 795 –799, 2001.

[6] U. Pal, N. Sharma, T. Wakabayashi, and F. Kimura. Handwritten numeral recognition of six popular indian scripts. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*, ICDAR '07, pages 749–753, Washington, DC, USA, 2007. IEEE Computer Society.

[7] U. Pal, T. Wakabayashi, and F. Kimura. A system for off-line oriya handwritten character recognition using curvature feature. In *Information Technology, (ICIT 2007). 10th International Conference on*, pages 227 –229, dec. 2007.

[8] Y. Hamamoto, S. Uchimura, and S. Tomita. On the behavior of artificial neural network classifiers in high-dimensional spaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(5):571 –574, may 1996.

[9] Cheng-Lin Liu and Hiromichi Fujisawa. Classification and learning methods for character recognition: Advances and remaining problems. In Simone Marinai and Hiromichi Fujisawa, editors, *Machine Learning in Document Analysis and Recognition*, volume 90 of *Studies in Computational Intelligence*, pages 139–161. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-76280-5-6.

[10] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[11] S. Rajasekaran and G.A.V. Pai. *Neural Networks, Fuzzy Logic and Genetic Algorithms: Synthesis and Applications.* Prentice-Hall of India, 2004.

[12] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.

[13] Haiyan Wang and Samy Bengio. The mnist database of handwritten upper-case letters. Idiap-Com Idiap-Com-04-2002, IDIAP, 2002.

[14] MATLAB. *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.