Intelligent Character Reader Using Artificial Neural Networks

Submitted by

Dipayan Deb-13000113032

Drona Banerjee-13000113033

Karan Jaiswal-13000113039

Payal Kumari-13000113060

Under the guidance of Prof. Poulami Dutta

Submitted for the partial fulfilment for the degree of Bachelor of Technology in Computer Science and Engineering

Techno India
EM 4/1, Salt Lake, Sector – V, Kolkata – 700 091.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Prof. Poulami Dutta of the department of Computer Science and Engineering, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest she took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staff for the gracious hospitality they offered us.

Place: Techno India, Salt Lake

Date: 02/01/2017

Dipayan Deb

Drona Banerjee

Karan Jaiswal

Payal Kumari

Department of Computer Science and Engineering
Techno India, Salt Lake
Kolkata – 700 091
West Bengal, India.

Approval

This is to certify that the project report entitled "Intelligent Character Reader Using Artificial Neural Networks" prepared under my supervision by Dipayan Deb (13000113032), Drona Banerjee (13000113033), Karan Jaiswal (13000113039) and Payal Kumari (13000113060) be accepted in partial fulfilment for the degree of Bachelor of Technology in Computer Science and Engineering.

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

………………………………….	………………………………….
Mrs. Poulami Dutta	Mr. Chandan Kumar Bhattacharyya
Prof., CSE Dept.	HOD, CSE Dept.

Contents **Page Number**

1. Introduction 1-2
    1.1 Briefing 1
    1.2 Problem Domain 1
    1.3 Glossary 2
2. Problem Definition 2-3
    2.1 Scope 2
    2.2 Exclusions 3
    2.3 Assumptions 3
3. Related Study 3-4
4. Project Planning 5-9
    4.1 Software Life Cycle Model 5
    4.2 Scheduling 7
    4.3 Cost Analysis 9
5. Requirement Analysis 11-15
    5.1 Requirement Matrix 11
    5.2 Requirement Elaboration 12
6. Design 16-23
    6.1 Technical Environment 16
    6.2 Hierarchy of Modules 16
    6.3 Detailed Design 18-23
        6.3.1 Input Module 18
        6.3.2 Segmentation 18
        6.3.3 Image Processing 19
        6.3.4 Machine Learning 19
        6.3.5 Output Module 21
    6.4 Test Plan 23
7. Conclusion 23-24
    7.1 Project Benefits 23
    7.2 Future Scope for improvements 24
8. References/Bibliography 24-25
APPENDIX A: Prototype 26-32

List of Tables: **Page Number**
Glossary 2
Cost Plan 9
Test Plan 22

List of Figures: **Page Number**
Gantt Diagram 7-8
Level 0 and 1 DFD 16
Context Diagram 17

1. INTRODUCTION

1.1. BRIEFING

Intelligent Character Reader (ICR) is a software tool that enables us to convert our handwritten or printed texts into digital, searchable text formats by using self-learning algorithms which learn to recognize human handwriting in a way which is similar to how humans learn new concepts. It is an advanced Machine Learning application which automatically updates its recognition database for new handwriting patterns using its self-learning system, known as Neural Networks. When fed with input data in the form of an image file such as .PNG, .JPEG or any other image file formats, the system is expected to process the image, identify the texts in the image, segment the characters, classify the characters in accordance with the existing database and then produce an output in a searchable text format like .DOC, .PDF, etc. ICRs can be used for several other purposes like automated forms processing, letter sorting, etc.

1.2. PROBLEM DOMAIN

Intelligent character recognition (ICR) is an advanced optical character recognition (OCR) or rather more specific — handwriting recognition system that allows fonts and different styles of handwriting to be learned by a computer during processing to improve accuracy and recognition levels.
Most ICR software has a self-learning system referred to as a **neural network**, which automatically updates the recognition database for new handwriting patterns

Artificial Neural Network - In **machine learning**, **artificial neural networks** (**ANNs**) is a network inspired by biological neural networks (the central nervous systems of animals, in particular the brain) which are used to estimate or approximate functions that can depend on a large number of inputs that are generally unknown.

Machine Learning-Machine learning is a subfield of computer science(more particularly soft computing) that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. It's the field of study that gives computers the ability to learn without being explicitly programmed. Machine learning explores the study and construction of algorithms that can be trained and can perform predictive analysis on data.

1.3. GLOSSARY

Table 1

DEFINITION	***TERM***
The image format stores an image as 2D matrix. Each Pixel is represented by a bit Integer. Therefore highest value being 255(White) and lowest being 0(Black).	Greyscale Image
Transforming digital information representing images.	Image Processing
Image segmentation is a process of dividing an image into multiple parts. This is typically used to identify objects or other relevant information in digital images.	Segmentation
Machine Learning Algorithm uses a learning algorithm to learn from a dataset of correct solutions.	MNIST Dataset

2. PROBLEM DEFINITON

2.1. SCOPE

The purpose of this project is to take handwritten English characters as Input, process the character, train the neural network algorithm, to recognize the pattern and modify the character to a searchable, digital version of the input. This project is aimed at developing software which will be helpful in recognizing characters, numeric as well as Special Characters of English language. This project is not restricted to English characters only. The design of the project is such that by adding a trained module in any other language is enough to add support for the language characters. One of the primary means by which computers are endowed with human like abilities is through the use of neural network. Neural Networks are particularly for solving problems that cannot be expressed as a series of steps such as recognizing patterns classifying them into groups, series prediction and data mining.

2.2. **EXCLUSION**

The significant exclusions in this project are as follows:

- The software does not use Natural Language Processing (NLP) and is thus not capable of making sense out of sentences or words. Hence, predictive correction of errors based on meaning of a sentence is not possible.
- The system should process the input given by the user only if it is an image file (.jpg).

2.3. **ASSUMPTIONS**

- The quality of image provided as input is reasonably good.

3. **RELATED STUDY**

A Pattern is a set of objects or phenomenon or concepts where the elements of sets are similar to one another in certain ways or aspects [1]. Pattern Recognition is one of important and vast fields of computer intelligence, which focuses on the recognition of patterns and regularities of data. It is an act of taking input of raw data and extract specific or unique features to recognize that data.
1985(Watanabe) said that pattern recognition can be looked as categorization problem, as inductive process, as structure analysis, as discrimination method and so on [2]. Optical Character Recognition is an application of pattern recognition. OCR is a method of converting a scanned image of typewritten, handwritten or printed text into computer readable format [3]-[5] i.e. a format that is understood by machine. Now when a page is scanned, it is stored in TIF format. When an image is displayed on the monitor, humans can read it, but to a computer, it‘s just a sequence of black and white pixels. Computer does not recognize any word in image. The main aim of OCR is to look at the image and try to decide if these pixels are representing a particular letter or not.

An Artificial Neuron is basically an engineering approach of biological neuron [6]-[7] i.e. Neural Networks basically aim at mimicking the structure and functioning of the human brain, to create intelligent behaviour. A Neural Network is a massively parallel distributed processor made up of simple processing units which have natural propensity for storing experiential knowledge and making it available for use. It resembles to brain in two aspects. First, Knowledge is acquired by the Network from its environment through a learning process. Second, Interneuron connection strength is used to store acquired knowledge. In Neural Network, each node performs some simple computation and each connection conveys a signal from one node to another labelled by a number called —connection strength [8].

Learning is formally defined as a process by which free parameters of a Neural Networks are adapted through a process of simulation by the environment in which the network is embedded. Once the system begins to learn containing some initial weight values, as the learning process increase weight values keeps on changing and provide the final output at end. Learning can be classified as: First, Supervised Learning i.e. learning with Teacher, Second, Unsupervised Learning i.e. learning without Teacher. Typical pattern recognition systems are designed using two pass. The first pass is a feature extractor that finds features within the data which are specific to the task being solved. The second pass is the classifier, which is more general purpose and can be trained using a neural network and sample data sets.

As Optical Character Recognition is defined as a Multiclass Problem, amongst various classification methods [9]-[10], we have used, Multilayer Feed Forward Architecture, which contains an Input Layer, an Output Layer and one or more Hidden Layer [11]. As the number of Hidden Layer increases the complexity of network also increases. Back-Propagation Neural Network (BPNN) algorithm is the most popular and the oldest supervised learning multilayer feed-forward neural network algorithm proposed by Rumelhart, Hinton and Williams [12]-[14]. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by you. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities.

4. PROJECT PLANNING

4.1. SOFTWARE LIFE CYCLE MODEL

For our project, we will be using the Iterative Waterfall Model.
An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

The iterative model can be broadly classified into four phases which unlike the classical waterfall model, are totally intertwined:

- Requirement Analysis and Specification: In this phase of the cycle the different requirements gathered are as follows:
    1) Functional requirements – Recognition of handwritten or printed characters including special characters
    2) Non-functional requirements – Recognizing the characters as fast as possible in an efficient manner.
    3) Goal of implementation– To create efficient and effective software for Handwriting recognition.

    In this phase the requirements for the software are gathered and analysed. Iteration eventually resulted in a requirements phase that produces a complete and final specification of requirements.

- Feasibility Study: Analysis of the problem and collection of all the relevant information relating to the project was done. These collected data are analyzed to arrive at the following:
    1) An abstract problem definition
    2) Formulating different strategies for solving the problem like Image recognition by comparing pixels or Image recognition using Neural networks
    3) Feasibility, benefits and shortcomings of the strategies were evaluated.

- Design: Data Flow Diagram are created as shown in Appendix A according to the Requirement gathered in the previous phase of this life cycle.

- Coding and Testing: The software design is translated into source code in this phase. The final product is broken into number of modules like testing the Alphabets recognition, Numeric recognition and Special character recognition and each module is tested in isolation and then brought together to be tested as complete system. The machine learning module of the software was coded using Octave and the Graphical User Interface (GUI) has been programmed with Java.

- Review: In this phase, the software is evaluated, the current requirements are reviewed, and changes and additions to requirements are proposed.

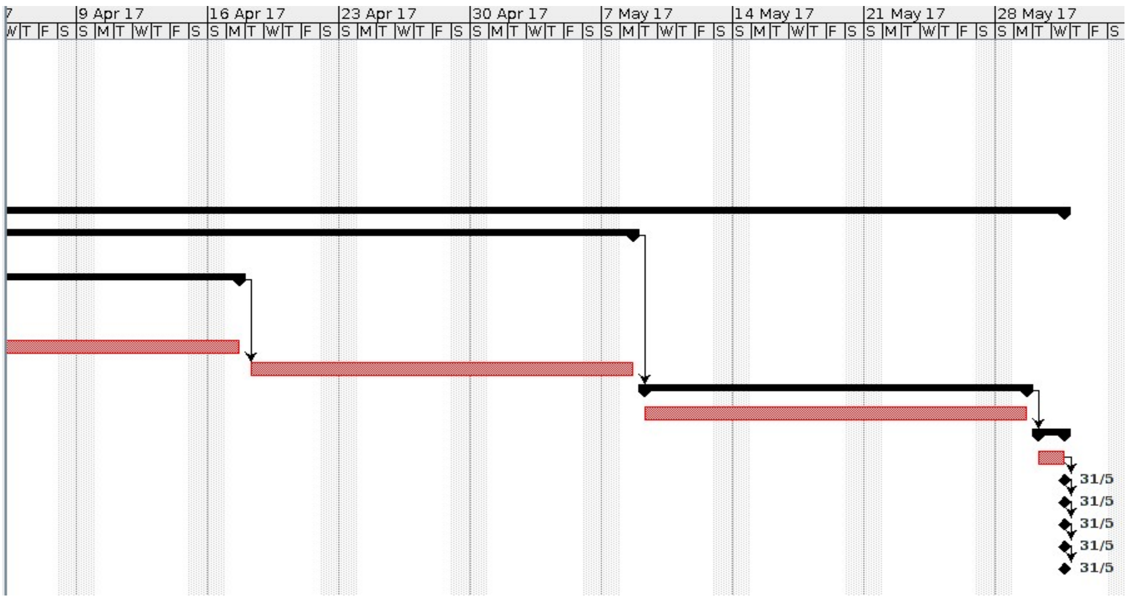
For each cycle of the model, a decision was made as to whether the software produced by the cycle will be discarded, or kept as a starting point for the next cycle. Eventually a point was reached where the requirements were complete and the software could be delivered, or it would have been impossible to enhance the software as required, and a fresh start would have been made.

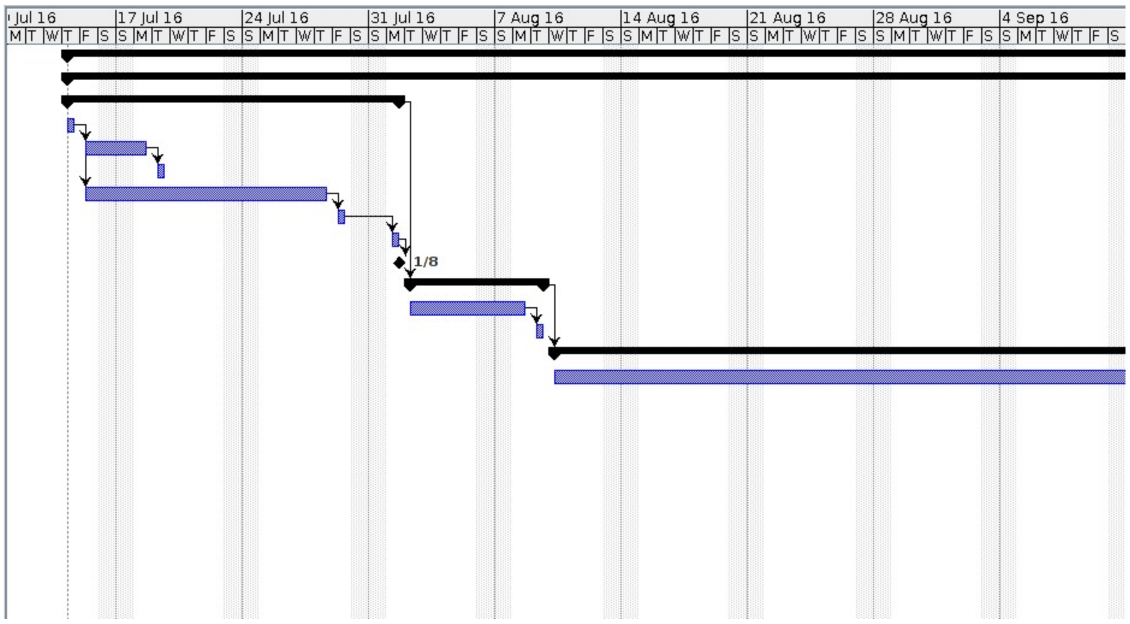
The iterative life cycle model can be likened to producing software by successive approximation. Drawing an analogy with mathematical methods that use successive approximation to arrive at a final solution, the benefit of such methods depends on how rapidly they converge on a solution.

The key to successful use of an iterative software development life cycle has been rigorous validation of requirements, and verification (including testing) of each version of the software against those requirements within each cycle of the model.

4.2. SCHEDULING

	ⓘ	Name	Duration	Start	Finish	Predecessors
1		CSE Academic Project	230 days	14/7/16 8:00 AM	31/5/17 5:00 PM	
2		Phase 1: 7th Semester Activities	123 days	14/7/16 8:00 AM	2/1/17 5:00 PM	
3		Project Startup	13 days	14/7/16 8:00 AM	1/8/16 5:00 PM	
4		Team Building	1 day	14/7/16 8:00 AM	14/7/16 5:00 PM	
5		Brainstorm on Project Topic	2 days	15/7/16 8:00 AM	18/7/16 5:00 PM	4
6		Project agreed with guide	1 day	19/7/16 8:00 AM	19/7/16 5:00 PM	5
7		Related Study & Doumentation	10 days	15/7/16 8:00 AM	28/7/16 5:00 PM	4
8		Deliver Project Synopsys for Guide's re...	1 day	29/7/16 8:00 AM	29/7/16 5:00 PM	7
9		Close review feedbacks	1 day	1/8/16 8:00 AM	1/8/16 5:00 PM	8
10		Project Synopsys Finalized	0 days	1/8/16 5:00 PM	1/8/16 5:00 PM	9
11		Requirement Analysis	6 days	2/8/16 8:00 AM	9/8/16 5:00 PM	3
12		Gather Requirements	5 days	2/8/16 8:00 AM	8/8/16 5:00 PM	
13		Prepare Draft Requirement Matrix	1 day	9/8/16 8:00 AM	9/8/16 5:00 PM	12
14		Elaborate Rqmt & Document	56 days	10/8/16 8:00 AM	26/10/16 5:00 PM	11
15		Artificial Neural Networks	50 days	10/8/16 8:00 AM	18/10/16 5:00 PM	
16		Java / Octave	5 days	19/10/16 8:00 AM	25/10/16 5:00 PM	15
17		Character Recognition	5 days	19/10/16 8:00 AM	25/10/16 5:00 PM	15
18		Update Requirement Matrix	1 day	26/10/16 8:00 AM	26/10/16 5:00 PM	16
19		Requirement Matrix Finalized	0 days	26/10/16 5:00 PM	26/10/16 5:00 PM	18
20		Design	37 days	26/10/16 8:00 AM	15/12/16 5:00 PM	
21		Detailed Design	26 days	26/10/16 8:00 AM	30/11/16 5:00 PM	16
22		Image Processing	5 days	26/10/16 8:00 AM	1/11/16 5:00 PM	
23		Machine learning module	20 days	2/11/16 8:00 AM	29/11/16 5:00 PM	22
24		Number Module	5 days	2/11/16 8:00 AM	8/11/16 5:00 PM	
25		Special character module	5 days	9/11/16 8:00 AM	15/11/16 5:00 PM	24
26		Character module	10 days	16/11/16 8:00 AM	29/11/16 5:00 PM	25
27		Training with MNIST	1 day	30/11/16 8:00 AM	30/11/16 5:00 PM	23
28		Test Plan Preparation	11 days	1/12/16 8:00 AM	15/12/16 5:00 PM	21
29		Preprocessing of Image	3 days	1/12/16 8:00 AM	5/12/16 5:00 PM	
30		Machine learning	3 days	6/12/16 8:00 AM	8/12/16 5:00 PM	29
31		Number Module	1 day	6/12/16 8:00 AM	6/12/16 5:00 PM	
32		Special Character Module	1 day	7/12/16 8:00 AM	7/12/16 5:00 PM	31
33		Character Module	1 day	8/12/16 8:00 AM	8/12/16 5:00 PM	32
34		Output	5 days	9/12/16 8:00 AM	15/12/16 5:00 PM	30
35		Phase 1 Closure	12 days	16/12/16 8:00 AM	2/1/17 5:00 PM	34
36		Prepare 7th Semester Project Report	11 days	16/12/16 8:00 AM	30/12/16 5:00 PM	
37		Updated Requirement Matrix	0 days	30/12/16 5:00 PM	30/12/16 5:00 PM	36
38		Updated Project Plan	0 days	30/12/16 5:00 PM	30/12/16 5:00 PM	37
39	▣	Project Viva	1 day	2/1/17 8:00 AM	2/1/17 5:00 PM	38
40	▣	Approved Project Report - 7th Semester	1 day	2/1/17 8:00 AM	2/1/17 5:00 PM	38
41	▣	Semester Gap	12 days	20/1/17 8:00 AM	6/2/17 5:00 PM	2
42		Phase 2: 8th Semester Activities	82 days	7/2/17 8:00 AM	31/5/17 5:00 PM	41
43		Coding & Unit Testing	65 days	7/2/17 8:00 AM	8/5/17 5:00 PM	
44		Image Processing	20 days	7/2/17 8:00 AM	6/3/17 5:00 PM	
45		Machine learning	30 days	7/3/17 8:00 AM	17/4/17 5:00 PM	44
46		Number Module	10 days	7/3/17 8:00 AM	20/3/17 5:00 PM	
47		Special Character Module	10 days	21/3/17 8:00 AM	3/4/17 5:00 PM	46
48		Character Module	10 days	4/4/17 8:00 AM	17/4/17 5:00 PM	47
49		Training with MNIST	15 days	18/4/17 8:00 AM	8/5/17 5:00 PM	45
50		System Integration Testing	15 days	9/5/17 8:00 AM	29/5/17 5:00 PM	43
51		Handwriting recognition with special ch...	15 days	9/5/17 8:00 AM	29/5/17 5:00 PM	
52		Project Closure	2 days	30/5/17 8:00 AM	31/5/17 5:00 PM	50
53		Prepare 8th Semester Project Report	2 days	30/5/17 8:00 AM	31/5/17 5:00 PM	
54		Updated Requirement Matrix	0 days	31/5/17 5:00 PM	31/5/17 5:00 PM	53
55		Updated Project Plan	0 days	31/5/17 5:00 PM	31/5/17 5:00 PM	54
56		Review by Internal Faculty	0 days	31/5/17 5:00 PM	31/5/17 5:00 PM	55
57		Review by Faculties	0 days	31/5/17 5:00 PM	31/5/17 5:00 PM	56
58		Approved Project Report -8th Semester	0 days	31/5/17 5:00 PM	31/5/17 5:00 PM	57


9 Apr 17
16 Apr 17
23 Apr 17
30 Apr 17
7 May 17
14 May 17
21 May 17
28 May 17
31/5
31/5
31/5
31/5
31/5


Jul 16
17 Jul 16
24 Jul 16
31 Jul 16
7 Aug 16
14 Aug 16
21 Aug 16
28 Aug 16
4 Sep 16
1/8

4.3.**COST ANALYSIS**

Software products are said to be feasible if they are developed within the budget constraints. The cost estimation models are used to predict the effort and cost required to develop a project. These models give a base to predict the cost for developing a software project. The cost estimation can be used to develop a product utilizing optimum resources.

Software cost estimation model is an indirect measure, which is used by software personnel to predict the cost of a project. The development of software product varies depending upon the environment in which it is being developed. For projects with familiar environment it is easy to predict the cost of the project. The estimation model is useful for trade-off between the developer and customer. Organization can realize of what is achievable and deliverable to the customer.

For the organization to develop a cost estimation model the following things are required.

- List important or critical cost drivers.
- Prepare a scaling model for each cost driver.
- Find projects with similar environments.
- Compare the project with previous familiar projects.
- Evaluate the project whether it is feasible within the budget constraints.
- Incorporate the critical features in an iterative manner

Cost drivers are those critical features which have an impact on the project. The cost drivers may vary the cost of building a project. The most important cost driver is size of the project. Size of the project is measured in Kilo lines of code (KLOC). Function points are the empirical measurement to measure size of the project. Function points may vary the size of the project due to the variation in: -

- Number of inputs
- Number of outputs
- Number of inquires
- Number of files
- Number of interfaces.

MATERIAL COST REQUIRED FOR SOFTWARE DEVELOPMENT:

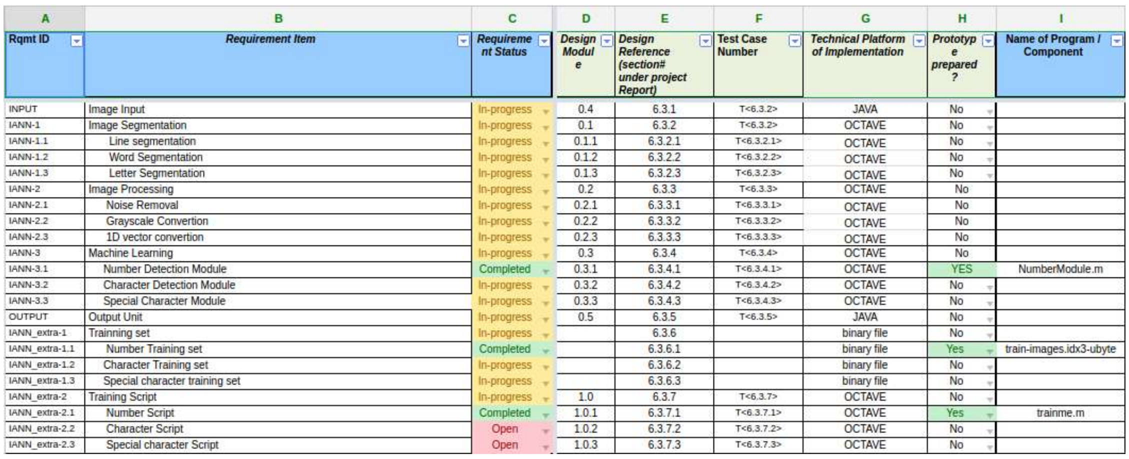
Item	Quantity	Cost(INR)
Microsoft Office 2016 for Home and Student (Word, Excel, PowerPoint)	1	5999.00
Windows 7 starter edition 32bit	1	1900.00

*The laptops or Personal computers are provided by the college to team members whose costing doesn’t come into consideration.

Total Cost: ₹7899.00

5.<u>REQUIREMENT ANALYSIS</u>

5.1,<u>REQUIREMENT MATRIX</u>



A	B	C	D	E	F	G	H	I
Rqmt ID	Requirement Item	Requirement Status	Design Module	Design Reference (section# under project Report)	Test Case Number	Technical Platform of Implementation	Prototype prepared ?	Name of Program / Component
INPUT	Image Input	In-progress	0.4	6.3.1	T<6.3.2>	JAVA	No	
IANN-1	Image Segmentation	In-progress	0.1	6.3.2	T<6.3.2>	OCTAVE	No	
IANN-1.1	Line segmentation	In-progress	0.1.1	6.3.2.1	T<6.3.2.1>	OCTAVE	No	
IANN-1.2	Word Segmentation	In-progress	0.1.2	6.3.2.2	T<6.3.2.2>	OCTAVE	No	
IANN-1.3	Letter Segmentation	In-progress	0.1.3	6.3.2.3	T<6.3.2.3>	OCTAVE	No	
IANN-2	Image Processing	In-progress	0.2	6.3.3	T<6.3.3>	OCTAVE	No	
IANN-2.1	Noise Removal	In-progress	0.2.1	6.3.3.1	T<6.3.3.1>	OCTAVE	No	
IANN-2.2	Grayscale Convertion	In-progress	0.2.2	6.3.3.2	T<6.3.3.2>	OCTAVE	No	
IANN-2.3	1D vector convertion	In-progress	0.2.3	6.3.3.3	T<6.3.3.3>	OCTAVE	No	
IANN-3	Machine Learning	In-progress	0.3	6.3.4	T<6.3.4>	OCTAVE	No	
IANN-3.1	Number Detection Module	Completed	0.3.1	6.3.4.1	T<6.3.4.1>	OCTAVE	YES	NumberModule.m
IANN-3.2	Character Detection Module	In-progress	0.3.2	6.3.4.2	T<6.3.4.2>	OCTAVE	No	
IANN-3.3	Special Character Module	In-progress	0.3.3	6.3.4.3	T<6.3.4.3>	OCTAVE	No	
OUTPUT	Output Unit	In-progress	0.5	6.3.5	T<6.3.5>	JAVA	No	
IANN_extra-1	Trainning set	In-progress		6.3.6		binary file	No	
IANN_extra-1.1	Number Training set	Completed		6.3.6.1		binary file	Yes	train-images.idx3-ubyte
IANN_extra-1.2	Character Training set	In-progress		6.3.6.2		binary file	No	
IANN_extra-1.3	Special character training set	In-progress		6.3.6.3		binary file	No	
IANN_extra-2	Training Script	In-progress	1.0	6.3.7	T<6.3.7>	OCTAVE	No	
IANN_extra-2.1	Number Script	Completed	1.0.1	6.3.7.1	T<6.3.7.1>	OCTAVE	Yes	trainme.m
IANN_extra-2.2	Character Script	Open	1.0.2	6.3.7.2	T<6.3.7.2>	OCTAVE	No	
IANN_extra-2.3	Special character Script	Open	1.0.3	6.3.7.3	T<6.3.7.3>	OCTAVE	No	

5.2.<u>REQUIREMENT ELABORATION</u>

5.2.1.<u>Image Input (INPUT)</u>

In this stage the software acquires an image as an input through a scanner or any suitable input method. This image should have a specific format such as .jpeg, .jpg, .png etc.

5.2.2. <u>Image Segmentation (IANN-1)</u>

Convert any image into a series of Black Text written on a white background. Thus, it induces uniformity to all the input images. This also reduces computational power as it to deal with two colours i.e. Black and White.

5.2.2.1. Line Segmentation (IANN-1.1)

Segmentation of a document image into its basic entities namely text lines and words, is a critical stage towards handwritten document recognition. The difficulties that arise in handwritten documents make the segmentation procedure a challenging task. There are many problems encountered in the segmentation procedure. For the text line segmentation procedure these include the difference in the skew angle between lines on the page or even along the same text line, overlapping words and adjacent text lines touching.

5.2.2.2. Word Segmentation (IANN-1.2)

Word Segmentation is an operation that seeks to decompose an image of a sequence of sentences into sub-images of individual words, separated by spaces. It is one of the decision processes in our system.

5.2.2.3. Letter Segmentation (IANN-1.3)

Character segmentation or Letter Segmentation is an operation that seeks to decompose an image of a sequence of characters into sub-images of individual symbols. It is one of the decision processes in our system.

5.2.3. Image Processing (IANN-2)

5.2.3.1. Noise Removal (IANN-2.1)

Noise removal is a topic in document analysis that has been dealt with extensively for typed or machine-printed documents. For handwritten documents, the connectivity of strokes has to be preserved. Digital capture of images can introduce noise from scanning devices and transmission media. Smoothing operations are often used to eliminate the arti-crafts introduced during image capture. One study, describes a method that performs selective and adaptive stroke "filling" with a neighbourhood operator which emphasizes stroke connectivity, while at the same time, conservatively check aggressive "over-filling.

5.2.3.2. **<u>Greyscale conversion (IANN-2.2)</u>**

In photography and computing, a greyscale or greyscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest.

Greyscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only two colours, black and white (also called bi-level or binary images). Greyscale images have many shades of gray in between.

Our system requires converting the true colour RGB image to the Greyscale intensity image.

5.2.3.3. **<u>1D Vector Conversion (IANN-2.3)</u>**

Our system requires converting multi-dimensional array into 1D vector for proper processing.

5.2.4. **<u>Machine Learning (IANN-3)</u>**

5.2.4.1. **<u>Number Detection Module (IANN-3.1)</u>**

In this module the system is expected to detect numbers like 1,2,7,9,etc.

5.2.4.2. **<u>Character Detection Module (IANN-3.2)</u>**

In this module the system is expected to detect regular characters like A, f, a, etc.

5.2.4.3. **<u>Special Character Module (IANN-3.3)</u>**

In this module the system is expected to detect special characters like @, !, &, #, etc.

5.2.5.**<u>Output Unit (OUTPUT)</u>**

In this unit, the system is expected to produce the required output for the input image in a user-friendly manner. The output needs to be a text file having formats like .txt, .doc, .pdf, etc.

5.2.6.**<u>Training Set (IANN_extra-1)</u>**

Before the recognition to be done, the Artificial Neural Network must be trained so that the network gets a potential of mapping various inputs to their corresponding output, so that the system classify various characters. For training the Neural Networks, we have different feature vectors obtained from the MNIST database using the feature extraction technique.

5.2.6.1.**<u>Number Training Set (IANN_extra-1.1)</u>**

The dataset is obtained from MNIST dataset. It has 60,000 datasets.

5.2.6.2. **<u>Character Training Set (IANN_extra-1.2)</u>**

This dataset is also obtained from MNIST.

5.2.6.3.**<u>Special Character Training Set (IANN_extra-1.3)</u>**

The special character dataset has to be developed by us.

5.2.7.**<u>Training Script (IANN_extra-2)</u>**

Training scripts are program that are the brains of the machine learning algorithm. These algorithms are used to train the computer to detect the desired objects. Example of machine learning algorithm is gradient descent.

5.2.7.1. **Number Training Script (IANN extra-2.1)**

This script is written in octave. The script uses gradient descent algorithm to train the algorithm in identifying number from images.

5.2.7.2. **Character Training Script (IANN_extra-2.2)**

This script is written is octave. The script uses back-propagation algorithm to train the algorithm in identifying number from images.

5.2.7.3.**Special Character Training Script (IANN_extra-2.3)**

This script is written in octave. The script uses gradient descent algorithm to train the algorithm in identifying special characters from images.

6. DESIGN

6.1. Technical Environment

For our projects we will be using multiple programming languages.

1. Java - It is a cross platform language. This will be used to create the Graphical User Interface of the application,
2. Octave - This is a mathematical programming language. This will be used to implement the Machine Learning and Neural Network components of the project.
3. Python - This is a very feature rich language in terms of Machine Learning and therefore can be used to further enhance the functionality of the project.

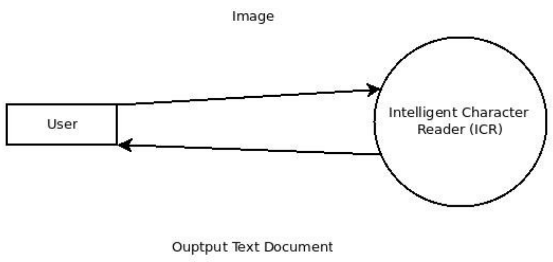
6.2. Hierarchy of Modules

The Hierarchy of the modules and sub-modules are as shown in the Level 0, Level 1 and Level 2 DFD diagram.

The Level 0 Diagram shows that our program accepts a Image as a input and outputs the text written in the image.

The Level 1 Diagram goes into a bit of more details. It shows that before the machine learning algorithm can take on the input the Image must be segmented and processed.

The Level 2 Diagram shows the exact working and data flow in our project. It shows that first the segmenter segments all the lines into different images. After that it segments each of the lines into images of different words and those words get segmented to images of letters.
After that noise reduction is done to make sure that the input image is of highest quality. After that it gets converted into greyscale image. Further it gets converted into a 1D vector containing the intensities of each pixel. The number, special character and character sub-modules detect the respective output and then output it via the output unit.



LEVEL 0 DFD (Context Diagram)

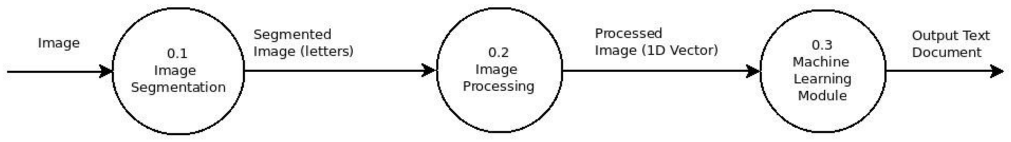


Image
0.1 Image Segmentation
Segmented Image (letters)
0.2 Image Processing
Processed Image (1D Vector)
0.3 Machine Learning Module
Output Text Document

LEVEL 1 DFD

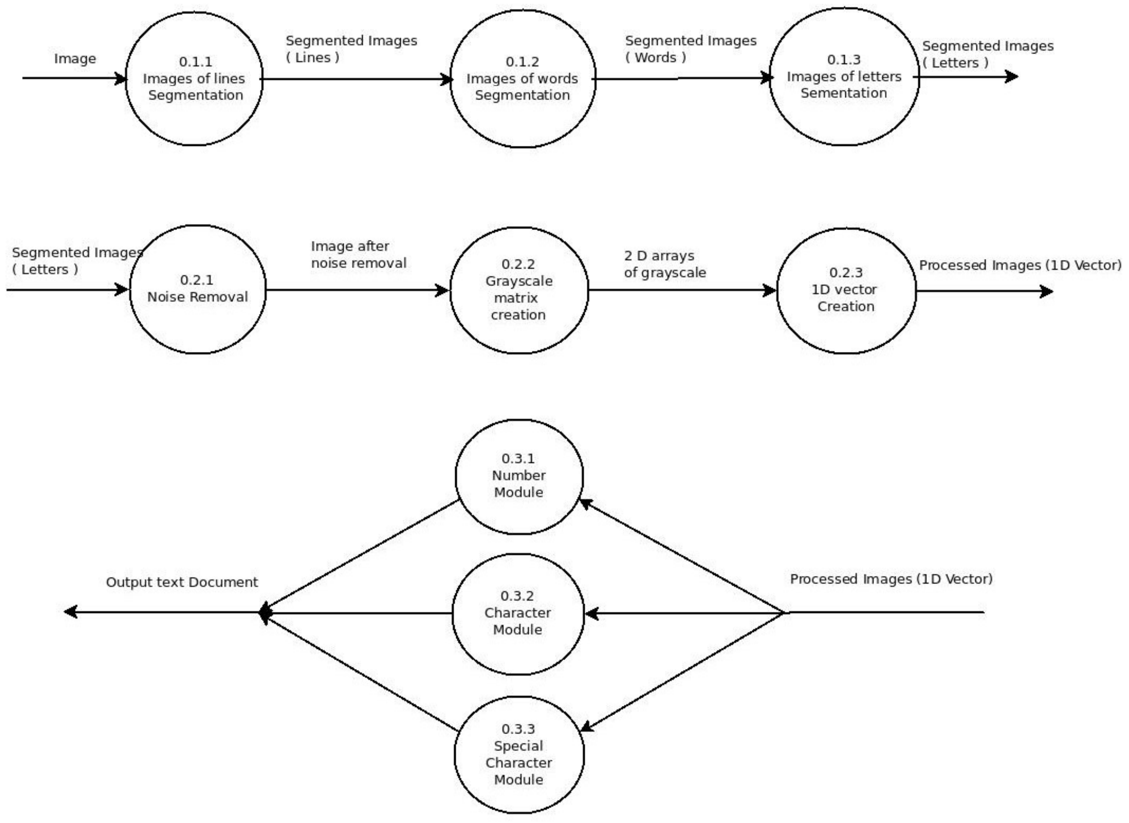
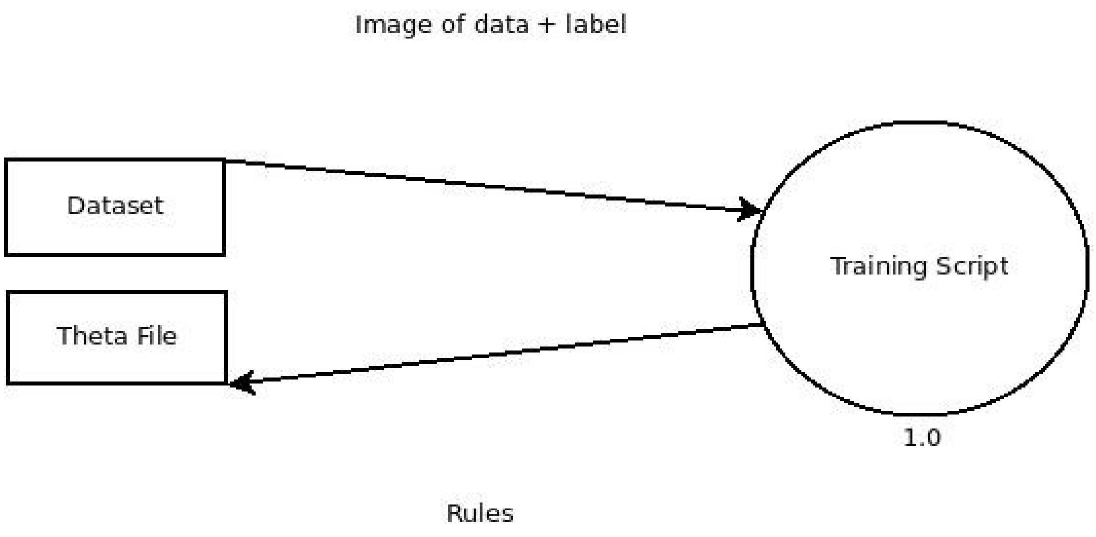


Image
0.1.1 Images of lines Segmentation
Segmented Images (Lines)
0.1.2 Images of words Segmentation
Segmented Images (Words)
0.1.3 Images of letters Sementation
Segmented Images (Letters)
Segmented Images (Letters)
0.2.1 Noise Removal
Image after noise removal
0.2.2 Grayscale matrix creation
2 D arrays of grayscale
0.2.3 1D vector Creation
Processed Images (1D Vector)
0.3.1 Number Module
0.3.2 Character Module
0.3.3 Special Character Module
Processed Images (1D Vector)
Output text Document

LEVEL 2 DFD



(a) LEVEL 0 DFD
(Context Diagram for Training Script)

6.3. Detailed Design

6.3.1. Input Module

This module takes a image as input which has text written in it. The text in the image will be extracted and stored in text form.

6.3.2. Segmentation

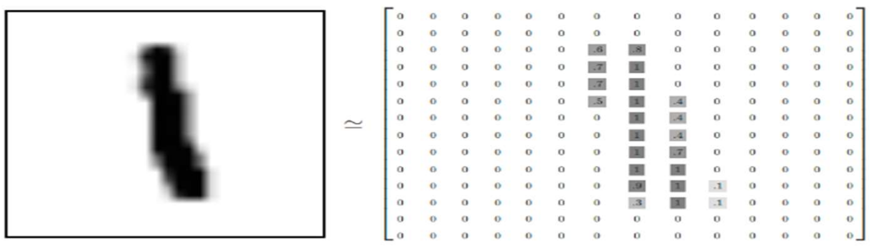
The input image consists of lines of text not just one character. This process will break down the image into images of the lines. Then it will further break down each line image into images of words. And finally each of those images of words will get broken into images of each letter of resolution 28 * 28.

6.3.3.**<u>Image Processing</u>**

The image of the letters needs to be processed before they can be used by the machine learning algorithm,

By default all the images are in RGB format. The RGB format is represented by a 3D matrix in the computer memory. The 3 2D matrices consists of the intensity of colours red, green and blue pixel by pixel.

For your purpose we will convert the Image into the greyscale format. This format is represented by a 2D array. We will remove all the necessary noise and then convert the values of the pixels between 0 and 1 for our algorithm to work.



Representation of 1 before vectorization.

After that we will have to convert the 2D array into an 1D vector so that we can apply a logistic regression.

6.3.4.**<u>Machine Learning</u>**

6.3.4.1.**<u>Number Module</u>**

The number module will convert a number in a given image into text. The number module uses the Logistic Regression. This is being treated as a classification problem.
Logistic Regression uses the sigmoid function. The value of this function is between 0 and 1.

The number module will have 10 parts or sub modules. Each sub module will have code to detect respective number from 0 to 9 . The input will first go to the submodule that detects 1 and so on until it reaches the last sub module that detects 9. Each submodule will give the result based on the sigmoid function. The number represented by the submodule that gives the result closest to 1 will the required answer.

Sigmoid Function.

g(z) = $1/(1+\ e^{-z})$

z=$\theta^T x$

$h_\theta(x)$=g($\theta^T x$)

θ^T- This is the transpose of the parameter vector . Each number has a different set of parameters.

x- This is a vector representing the 1D vector obtained from the input image.

The θ vector for a particular number is determined for a particular number with the help of algorithm called gradient descent.
A huge number of training set are taken and then provided to the gradient descent algorithm . This algorithm then uses the data set to properly the exact values of θ.

Gradient Descent

```
Repeat {

    $\theta_j = \theta_j$ - □((1/m) $\sum_1^m$ ($h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$)
}
```

6.3.4.2.**<u>The special characters module</u>**

This module is used to detect the special characters. This works in exactly the same way as the numbers module. It utilises the same algorithm. Just at the time of training a dataset of special characters is used instead of number dataset.

6.3.4.3.**<u>The characters module</u>**

This module will use a Neural Network to solve the problem. Each of the pixel intensities in the 1D array are used as inputs to the Neural Networks. We use a 3 layer Neural Network. The network consists of one hidden layer.

Shown above is a representation of a 3 layer neural networks. As we are using a 28 * 28 resolution Image as input there are 784 input neuron. Since there are 26 letters in the english language therefore there will be 26 output neurons. The above neural network has 1 hidden layer.

Similar to the previous algorithm the θ of the neural network are calculated using gradient descent algorithm with the help of backpropagation algorithm.

6.3.5.**<u>Output Unit</u>**

This module looks at the output of different machine learning module and then provides the desired output.

6.3.6.**<u>Training Set</u>**

Before the recognition to be done, the Artificial Neural Network must be trained so that the network gets a potential of mapping various inputs to their corresponding output, so that the system classify various characters. For training the Neural Networks, we have different feature vectors obtained from the MNIST database using the feature extraction technique.

6.3.6.1.**<u>Number Training Set</u>**

The dataset is obtained from MNIST dataset. It has 60,000 datasets.

6.3.6.2.**Character Training Set**

This dataset is also obtained from MNIST.

6.3.6.3.**Special Character Training Set**

The special character dataset has to be developed by us.

6.3.7.**Training Script**

Training scripts are program that are the brains of the machine learning algorithm. These algorithms are used to train the computer to detect the desired objects. Example of machine learning algorithm is gradient descent.

6.3.7.1. **Number Training Script**

This script is written in octave. The script uses gradient descent algorithm to train the algorithm in identifying number from images.

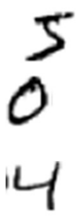
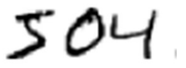
6.3.7.2. **Character Training Script**

This script is written is octave. The script uses back-propagation algorithm to train the algorithm in identifying number from images.

6.3.7.3.**Special Character Training Script**

This script is written in octave. The script uses gradient descent algorithm to train the algorithm in identifying special characters from images.

6.4. **Test Plan**

Output	Input	ID	Serial Number
		T-<6.3.2>-	1
[0,0,0,0,0,0,0,0,0.98 ,0.76,0,0, ….. 0,0,]		T-<6.3.3>-	2
98% chance of being 0	[0,0,0,0,0,0,0,0,0.98 ,0.76,0,0, ….. 0,0,]	T-<6.3.4>-	3
0	98% chance of being 0	T-<6.3.5>-	4

7. **CONCLUSION**

7.1. **Project Benefits**

Unlike OCRs, ICRs can recognise handwritten text with incredible accuracy because of the neural network based learning algorithms.
Today's letter sorting machines use OCRs. These machines try to read the pin code of a letter and classify them accordingly. Although this speeds up the process, but still a part of the process is done by hand. Human intervention is needed when the OCR fails to read the pin code. ICRs will eliminate the need for human intervention as it can have an accuracy of more than 97%

ICRs are also useful for automated forms processing. Many organization use handwritten forms. OCRs are not suitable as they are unable to read handwriting with good amount of accuracy. But ICRs are perfect for the job.

ICR can be used to help people with eye problems too read text that's not clearly visible.

E.g.: ATM pin provided by bank is often difficult to read due to the way they are printed. They often have black stains of ink blocking them. Therefore people with eye disability are unable to read properly. These people can use ICRs to read the required information.

7.2.**Future Scope of Improvement**

ICRs read character by character .Therefore they sometimes cannot read continuous handwriting. In order to overcome this problem Intelligent Word Readers can be built that process one word at a time. This can lead to much better performance in continuous handwriting.

The recognition of ICRs is based on the principle of eye sight i.e. it tries to mimic the way human eyes work. But there are cases when even if the human eye is not able to understand a particular letter in a word, the can human determine it based on the meaning of the sentence. This conscience can be implemented into the computer using Natural Language Processing, Artificial Intelligence and incredibly advanced machine learning.

8.**REFERENCES**

[1] Priyanka Sharma, Manavjeet Kaur, ―Classification in Pattern Recognition: A Review‖, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 4, April 2013.

[2] Jie Liu, Jigui Sun, Shengsheng Wang, ―Pattern Recognition: An Overview‖, International Journal of Computer Science and Network Security, VOL.6 No.6, June 2006.

[3] A.A Zaidan, B.B Zaidan, Hamid.A.Jalab, Hamdan.O.Alanazi and Rami Alnaqeib, ―Offline Arabic Handwriting Recognition Using Artificial Neural Network‖, Journal of Computer Science and Engineering, Volume 1, Issue 1, May 2010.

[4] V.Kalaichelvi, Ahammed Shamir Ali, ―Application of Neural Network in Character Recognition‖, International Journal of Computer Applications (0975 – 8887) Volume 52– No.12, August 2012.

[5] Divakar Yadav, Sonia Sánchez Cuadrado, Jorge Morato, ―Optical Character Recognition for Hindi Language Using a Neural-network Approach‖, J Inf Process Syst, Vol.9, No.1, March 2013.

[6] Vidushi Sharma, Sachin Rai, Anurag Dev, ―A Comprehensive Study of Artificial Neural Networks‖, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012.

[7] Jyoti Singh, Pritee Gupta, ―Advance Applications of Artificial Intelligence and Neural Networks: A Review‖, AKGEC JOURNAL OF TECHNOLOGY, vol.1, no.2.

[8] Chirag I Patel, Ripal Patel, Palak Patel, ―Handwritten Character Recognition using Neural Network‖, International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011.

[9] Guobin Ou, Yi Lu Murphey, ―Multi-class pattern classification using neural networks‖, A Journal of the Pattern Recognition Society‖, Vol 40, 2007,pg 4-18.
[10] Guobin Ou, Yi Lu Murphey, ―Multi-class pattern classification using neural networks‖, Journal of Pattern Recognition, Volume 40, Issue 1, January, 2007

[11] Omer Mahmoud, Farhat Anwar, Momoh Jimoh E. Salami, ―learning algorithm effect on multilayer feed Forward artificial neural network performance In image coding‖, Journal of Engineering Science and Technology Vol. 2, No. 2 (2007) 188 – 199.

[12] Dr. Rama Kishore, Taranjit Kaur, ―Backpropagation Algorithm: An Artificial Neural Network Approach for Pattern Recognition‖, International Journal of Scientific & Engineering Research, Volume 3, Issue 6, June-2012

[13] Shital Solanki, H. B. Jethva, ―A Review on Back Propagation Algorithms for Feed Forward Networks‖, GRA – Global Research Analysis volume: 2, Issue: 1, Jan 2013.

[14] Saduf, Mohd Arif Wani, ―Comparative Study of Back Propagation Learning Algorithms for Neural Networks‖, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 12, December 2013.

APPENDIX-A

CODE:

trainme.m

```
# X will be generated in the program named LoadMNISTImages function.

X=loadMNISTImages('train-images.idx3-ubyte');


# We use 1000 datasets for our training. i.e m=1000.
#X=X(:,1:10000);
A = ones(1,60000);
#Adding x0=1 as for constant.
X=[A;X];


Y=loadMNISTLabels('train-labels.idx1-ubyte');
Y=Y';

# We are considering the result as a binary thingy.
for buga=1:60000,
  if Y(buga)==0,
    Y(buga)=1;
  else
  Y(buga)=0;
 end;
end;


# This function calculates the hypothesis fucntion.
# O-A horizontal vector of the parameters that the gradient descent will find out.
# X- A 2d array that contains all the Training set column wise. i.e each column represents
a particular picture.
# Returns a horizontal vector containing the result of hypothesis(b).

function b = h(O,X)

z=O*X;
b=1./(1+e.^-z);

end


#New set of thetas or O's.
# Y - this is a horizonal vector containing the actual result.
# This function calculates theta for gradient descent.
# m is the number of elements in the training set.
# aplha is the learning rate.
function boo = newTheta(O,b,Y,X,m,alpha)

boo= O-((alpha*(1./m))*((b-Y)*X'));

end

# Now we write the gradient descent loop.
function Z = gradientDescent(O,Y,X)
odash=ones(1,785);
truthvalue=true;
count=0;
 while truthvalue,
   k=newTheta(O,h(O,X),Y,X,60000,0.01);
   odash=O;
   O=k;
        if size(O(odash-O>0.00001),2)<300,
                truthvalue=false;
        end;
```

```
          count=count+1;
          display(count);
  end;
    Z=O;
end

O0=rand(1,785);

O0=gradientDescent(O0,Y,X);

O0

save Thetafor0.txt O0 -ascii;
save Thetafor0Binary.dat O0;
```

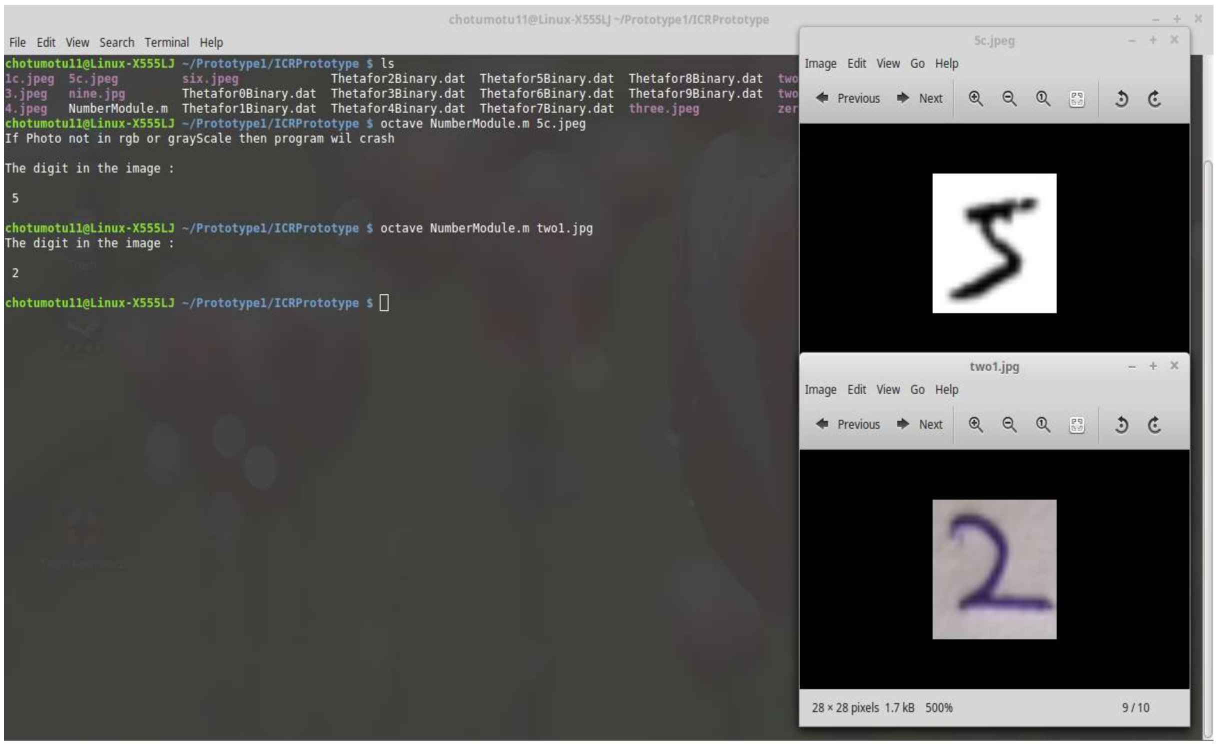
NumberModule.m

```
pkg load image;

n=argv(){1};
Photo= imread(n);
if size(Photo,3)==3,
    Photo=rgb2gray(Photo);
else
  display("If Photo not in rgb or grayScale then program wil crash");
end;

level = graythresh(Photo);
j2 = im2bw(Photo,level);

for i=1:28,
  for j=1:28,
    if j2(i,j)==1,
      j2(i,j)=0;
    else
      j2(i,j)=1;
    end;
  end;
end;


grayj2=uint8(j2);
```

```
for i=1:28,
  for j=1:28,
    if j2(i,j)==1,
        grayj2(i,j)=Photo(i,j)+50;
      end;
    end;
  end;

Photo=grayj2;

Photo = double(Photo);
#for i=1:28,
#  for j=1:28,
#    if Photo(i,j)<40.0,
#      Photo(i,j)=0.0;
#    end;
#  end;
#end;

#save boo.dat Photo;
Photo=Photo/255;

Photo=reshape(Photo,1,784);
Photo=Photo';
Photo = [1;Photo];
Photo = Photo';

#save Test.dat Photo;

load Thetafor0Binary.dat
load Thetafor1Binary.dat
load Thetafor2Binary.dat
load Thetafor3Binary.dat
load Thetafor4Binary.dat
load Thetafor5Binary.dat
load Thetafor6Binary.dat
load Thetafor7Binary.dat
load Thetafor8Binary.dat
load Thetafor9Binary.dat

function per = h1(O,X)
```

```
  z=O*X';
  per=1./(1+e.^-z);

end

k0=h1(O0,Photo);
#k0
k=h1(O,Photo);
#k
k2=h1(O2,Photo);
#k2
k3=h1(O3,Photo);
#k3
k4=h1(O4,Photo);
#k4
k5=h1(O5,Photo);
#k5
k6=h1(O6,Photo);
#k6
k7=h1(O7,Photo);
#k7
k8=h1(O8,Photo);
#k8
k9=h1(O9,Photo);
#k9

max=0;
ans=-1;

if k0>max,
    max=k0;
    ans=0;
end;
if k>max,
     max=k;
     ans=1;
end;
if k2>max,
     max=k2;
     ans=2;
end;
if k3>max,
    max=k3;
    ans=3;
end;
if k4>max,
    max=k4;
    ans=4;
end;
if k5>max,
    max=k5;
    ans=5;
end;
if k6>max,
    max=k6;
    ans=6;
end;
if k7>max,
    max=k7;
    ans=7;
end;
if k8>max,
    max=k8;
    ans=8;
end;
if k9>max,
    max=k9;
    ans=9;
end;

display(ans);
```

SCREENS :


chotumotu11@Linux-X555LJ ~/Prototype1/ICRPrototype
File Edit View Search Terminal Help
5c.jpeg
Image Edit View Go Help
Previous Next
two1.jpg
Image Edit View Go Help
Previous Next
28 × 28 pixels 1.7 kB 500%
9 / 10

```
chotumotu11@Linux-X555LJ ~/Prototype1/ICRPrototype $ ls
1c.jpeg  5c.jpeg         six.jpeg              Thetafor2Binary.dat  Thetafor5Binary.dat  Thetafor8Binary.dat  two
3.jpeg   nine.jpg        Thetafor0Binary.dat   Thetafor3Binary.dat  Thetafor6Binary.dat  Thetafor9Binary.dat  two
4.jpeg   NumberModule.m  Thetafor1Binary.dat   Thetafor4Binary.dat  Thetafor7Binary.dat  three.jpeg           zer
chotumotu11@Linux-X555LJ ~/Prototype1/ICRPrototype $ octave NumberModule.m 5c.jpeg
If Photo not in rgb or grayScale then program wil crash

The digit in the image :

 5

chotumotu11@Linux-X555LJ ~/Prototype1/ICRPrototype $ octave NumberModule.m two1.jpg
The digit in the image :

 2

chotumotu11@Linux-X555LJ ~/Prototype1/ICRPrototype $ 
```

TEST DATA :

SAMPLE OUTPUT:

5c.jpeg

$ octave NumberModule.m 5c.jpeg

$ The digit in the image: 5