**Chapter-1**

**ABSTRACT**

The aim of the project **‘OCR for Devanagari Font**’ is to develop OCR software for recognizing the Devanagari Script Characters from the scanned digital image. OCR is an Optical character recognition and is the mechanical or electronic translation of images of handwritten or typewritten text (usually captured by a scanner) into machine-editable text. OCR is a field of research in pattern recognition, artificial intelligence and machine vision.

Handwritten and Typewritten recognition is used most often to describe the ability of a computer to translate human writing into text. This may take in one of the two ways, either by scanning of written text or by writing directly on peripheral input devices.

**Chapter-2**

**PROJECT OVERVIEW**

Machine replicating human functions, like reading, is an old dream. However, over the last five decades, machine reading has grown from a dream to reality. Machine reading uses the principles of Optical Character Recognition (OCR). OCR has also become one of the most successful applications of technology in the field of pattern recognition and artificial intelligence. Since the mid 1950’s, OCR has been a very active field of research and development. While the OCR technology for some scripts like Latin is fairly mature and commercial OCR systems like Nuance OmniPage Pro or ABBYY FineReader are available which can perform with high accuracy, it is still under development for other scripts like Chinese and Devangari. Although a great deal of research has been done for OCR applications for Latin script, even theses   
OCR based machines are still not able to compete with human reading capabilities. This problem is more prominent for other scripts for which OCR technology is relatively newer. Typefaces are very important in determining the performance of the OCR technology. Hence in order to improve the accuracy of the OCR system, typefaces which are specially designed for OCR are required. For Latin script, quite a few typefaces have been designed which are optimized for OCR. These specially designed typefaces have a unique and well defined character set which allows for greater accuracy in recognition. This in turn helps in building low cost systems which can recognize characters using simple algorithms. However, no Devanagari script font is available which is designed specifically for machine reading and we address this problem in this report. In general, documents contain text, graphics, and images. The procedure of reading the text component in such a document can be divided into three steps:

1. Document layout analysis in which the text component of the document is extracted.

2. Segmentation, i.e. extraction of characters from the text component of the document.

3. Recognition of the segmented characters.

Typically, the OCR character segmentation stage needs to be redesigned for each new script, while   
the other stages are easier to port from one script to another and can be generalized over large   
classes of languages. There is a great need for OCR related research in Indian languages as there   
are many technical challenges which are specific to Devanagari script. With the spread of computers in organizations and offices, automatic processing and machine reading of paper documents is gaining importance in India. Although a lot of research is going on Devanagari script recognition, there is no commercial OCR systems focusing on Devanagari based languages.

**Chapter-3**

**INTRODUCTION**

Optical Character Recognition (OCR) is a form of computer vision that extracts alphanumeric characters from a digital image. The technology can be used for digitizing printed text, handwriting recognition, and making digital images searchable for text. Optical Character Recognition is an important and practical technology in the computer age. More people than ever before are using personal computers, laptop, tablets, and e-readers to read documents and books. This means that old print media must be scanned and converted to a digital format in order to be accessed from these devices. Optical Character Recognition (OCR) programs are used to read scanned images and convert them into a digital character-based format.

OCR work on printed Devnagari script started in early 1970s. Earlier studies on Devnagari script presented a Devnagari hand-printed numeral recognition system based on binary decision tree classifier. The study investigates the direction of the Devnagari Optical Character Recognition research (DOCR), analyzing the limitations of methodologies for the systems which can be classified based upon two major criteria: the data acquisition process (online or off-line) and the text type (machine-printed or hand-written). No matter which class the problem belongs, in general there are five major stages in the DOCR problem:

1. Pre-processing

2. Segmentation

3. Feature Extraction

4. Recognition

5. Classification

The off-line and on-line character recognition techniques have different approaches; they share a lot of common problems and solutions. Since it is relatively more complex and requires more research compared to on-line and machine-printed recognition, offline handwritten character recognition is selected as a focus of attention Handwriting Recognition Technology has been improving much under the purview of pattern recognition and image processing since a few decades. Hence various soft computing methods involved in other types of pattern and image recognition can as well be used for DOCR. Optical Character Recognition is a process by which we convert printed document or scanned page to ASCII character that a computer can recognize. The document image itself can be either machine printed or handwritten, or the combination of two. Computer system equipped with such an OCR system can improve the speed of input operation and decrease some possible human errors. Recognition of printed characters is itself a challenging problem since there is a variation of the same character due to change of fonts or introduction of different types of noises. Most of the Indian scripts are composed in two dimensions that make them different from Roman script. Therefore, the algorithms developed for Roman script are not directly applicable to Indian scripts. Many works on Indian scripts OCR have been reported. However, none of these works have considered real-life printed text in Devanagari. Consisting of character fusions and noisy environment. The present a complete OCR for printed text that is written in Devanagari script. The OCR has been tested on samples from various magazines and newspapers.

In a multilingual country like India, a document may contain text words in more than one language. For a multilingual environment in order to reach a larger cross section of people, it is necessary that a document should be composed of text contents in different languages. But on the other hand, this causes practical difficulty in OCR such a document, because the language type of the text should be pre-determined, before employing a particular OCR. It is perhaps impossible to design a single recognizer which can identify a large number of scripts/languages. So, it is necessary to identify the language region of the document before feeding the document to the corresponding Optical Character Recognition (OCR) system. Identification aims to extract information presented in digital documents namely articles, newspapers, magazines and e-books. This has given rise to many language identification systems. The objective of this paper is to develop visual clues based procedure to identify different text portions of a document. One important task of document image analysis is automatic reading of text information from the document image. The tool Optical Character Recognition (OCR) performs this, which is broadly defined as the process of reading the optically scanned text by the machine. Almost all existing works on OCR make an important implicit assumption that the script type of the document to be processed is known beforehand. In an automated multilingual environment, such document processing systems relying on OCR would clearly need human Intervention to select the appropriate OCR package, which is certainly inefficient, undesirable and impractical. The ability to reliably identify the script type using the least amount of textual data is essential when dealing with document pages that contain text words of different script. It is difficult to feed a document as an input to OCR unless the language type of the text in it is pre-determined since a single OCR cannot recognize multiple languages. This can be solved by developing script identification systems. This addresses the need of developing tools that can recognize and analyze varied documents.

**Chapter-4**

**PROBLEM STATEMENT**

Currently there are many systems are available for performing the optical character Recognition.

All these systems are built for standard foreign languages like English, French, Spanish and German. They does not support the Recognition of regional languages like Marathi, Sanskrit, Hindi etc. We are going to build a Software which is capable of recognizing Devanagari Script Characters.

Our main aim is to recognize these characters and to store or save them into a standard text file so that the user can easily go through them and can perform various text related operations like Copy, Paste, Find, Replace etc.

**Chapter-5**

**REQUIREMENT ANALYSIS**

**5.1 FEASIBILITY STUDY**

All projects are feasible, given unlimited resources and infinite time. But the development of software is plagued by the scarcity of resources and difficult delivery rates. It is prudent to evaluate the feasibility of the project at the earliest possible time.

**5.1.1 TECHNICAL FEASIBILITY**

Technical feasibility centers on the existing computer system (Hardware, Software etc.,) and to what extent it can support the proposed addition. If the budget is a serious constraint, then the project is judged not feasible.

**5.1.2 ECONOMIC FEASIBILITY**

This procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. It benefits outweigh costs, and then the decision is made to design and implement the system. Otherwise, further justification or alternations in proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of the system lifecycle.

**5.2 REQUIREMENTS**

**5.2.1 SOFTWARE REQUIREMENTS**

Windows 7

Matlab R2013b

**5.2.2 HARDWARE REQUIREMENTS**

I3 PROCESSOR

4GB RAM

250 GB HDD

**Chapter-6**

**PROJECT DESIGN**

**6.1 FUNCTIONAL REQUIREMENTS**

**DATA FLOW DIAGRAM**

A **data flow diagram** (**DFD**) is a graphical representation of the “flow” of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then “exploded” to show more detail of the system being modeled.

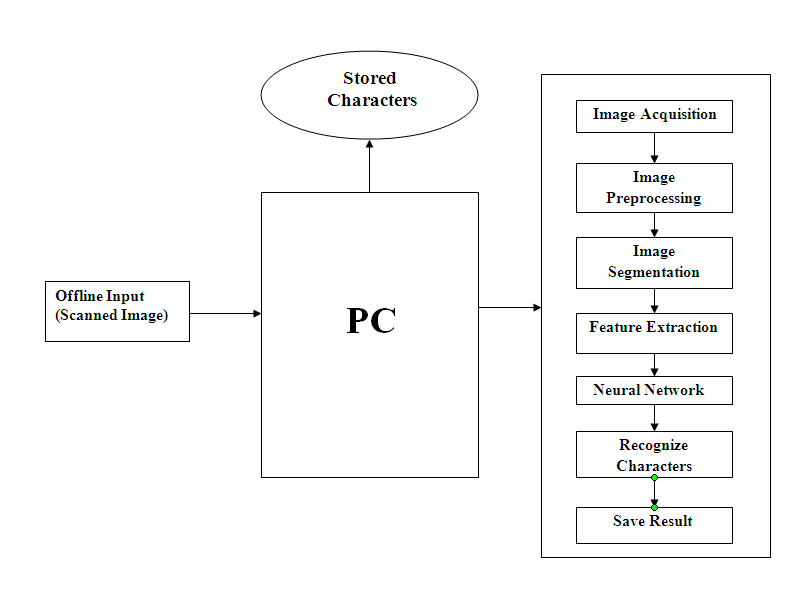
**OCR**

**Application**

Figure(6.1)-DFD

**6.2 PROCEDURAL DESIGN**

**6.2.1 System Architecture**

System Architecture is the overall organization of a system, broken into several components called subsystems.

Figure(6.2 a)-System Architecture

Our system will recognize the Devanagari Script Characters from the scanned digital image. Instead of giving the direct input of digital image to the OCR Engine system will preprocess the image. This Preprocessed image is then segmented. System will extract the features from each and every segmented part of the image. After Feature Extraction the processed data is then passed towards the OCR Engine which will perform the Character Recognition.

The most important principle of automatic pattern recognition is training the machine what kind of pattern may be present and what they look like. In OCR the patterns are letters, numbers and punctuations. Machine is trained to recognize the pattern by showing it all the kind of characters present in the script. This period is referred as the training period. On the basis of these examples the machine built a prototype of all the characters. Then during recognition the machine compares the unknown character to the prototype and assigns the character which is the closest match. The three steps in recognition are as follow.

1. Preprocessing

2. Segmentation/Cropping

3. Recognition

**6.2.1.1 Preprocessing:**

Image processing refers to processing of a 2D picture by a computer. Basic definitions:

An image defined in the “real world” is considered to be a function of two real variables, for example, a(x,y) with a as the amplitude (e.g. brightness) of the image at the real coordinate position (x,y).

Modern digital technology has made it possible to manipulate multi-dimensional signals with systems that range from simple digital circuits to advanced parallel computers. The goal of this manipulation can be divided into three categories:

* Image Processing (image in -> image out)
* Image Analysis (image in -> measurements out)
* Image Understanding (image in -> high-level description out)

An image may be considered to contain sub-images sometimes referred to as regions-of-interest, ROIs, or simply regions. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions. Thus one part of an image (region) might be processed to suppress motion blur while another part might be processed to improve color rendition. Sequence of image processing:

Most usually, image processing systems require that the images be available in digitized form, that is, arrays of finite length binary words. For digitization, the given Image is sampled on a discrete grid and each sample or pixel is quantized using a finite number of bits. The digitized image is processed by a computer. To display a digital image, it is first converted into analog signal, which is scanned onto a display.

It refers to accentuation, or sharpening, of image features such as boundaries, or contrast to make a graphic display more useful for display & analysis. This process does not increase the inherent information content in data. It includes gray level & contrast manipulation, noise reduction, edge crispening and sharpening, filtering, interpolation and magnification, pseudo coloring, and so on.

The text document is generally scanned at 300 or 400 DPI. Preprocessing is also done to improve the accuracy of the recognition algorithm. Main steps in preprocessing are noise removal, binarization and skew correction.

The main sources of noise in the input image are as follows:

•Noise due to the quality of paper on which the printing is done.

•Noise induced due to printing on both sides of paper or the quality of printing

•Noise added due to the scanner source brightness and sensors.

All this noise results in reduction of accuracy of OCR system. As a result of this having a noise   
correction routine in place becomes inevitable. To reduce the amount of noise, image is passed   
through a mean filter; in this filter the intensity of the each pixel is replaced by the average intensity   
of pixels surrounding it. After de-noising the image is subjected to binarization and skew (or tilt)   
correction.

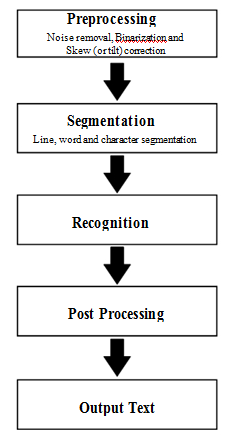


Figure 6.2.b Procedure of Devanagari Recognition

**Binarization**

Printed documents generally are black text on white background. Hence most of the OCR algo-  
rithms are designed to interpret bi-level images (an image that has only two possible value of pixel

i.e. black and white). This process of converting colored or grayscale images to bi-level image is often known as binarization or thresholding



Figure 6.2.c Image before binarization (left); Image after binarization (right)

**6.2.1.2 Image Cropping**

Image cropping is performed to select only the required portion from the image and to use it for underlying processing operations .Cropped image is quiet easy to process as it is smaller in size as well as it contains low level of information.

Here we are using image cropping because it is not possible to process each and every complete word at a time. There is no any kind of algorithm present for cropping the “shirorekha” hence we have to do this work manually. Result of image cropping is as shown below:

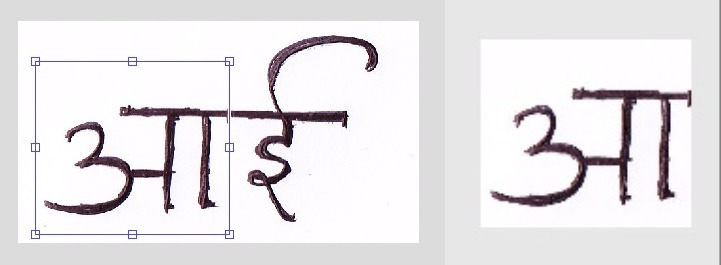


Figure 7.4 Image before and after cropping

**6.2.1.3 Image Recognition**

The cropped image is ready to be recognized. Next step begins with training the neural network.

Neural network software is used to [simulate](http://en.wikipedia.org/wiki/Simulation), [research](http://en.wikipedia.org/wiki/Research), develop and apply artificial neural networks, [biological neural networks](http://en.wikipedia.org/wiki/Biological_neural_network) and, in some cases, a wider array of [adaptive systems](http://en.wikipedia.org/wiki/Adaptive_system).

The instance of a neural network that we used is a feed forward neural network. Following techniques are available for training the neural network

### Supervised Training.

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.

### Unsupervised, or Adaptive Training.

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption. At the present time, unsupervised learning is not well understood. This adaption to the environment is the promise which would enable science fiction types of robots to continually learn on their own as they encounter new situations and new environments. Life is filled with situations where exact training sets do not exist. Some of these situations involve military action where new combat techniques and new weapons might be encountered. Because of this unexpected aspect to life and the human desire to be prepared, there continues to be research into, and hope for, this field. Yet, at the present time, the vast bulk of neural network work is in systems with supervised learning. Supervised learning is achieving results

#### **Reinforcement learning**

In [reinforcement learning](http://en.wikipedia.org/wiki/Reinforcement_learning), data \scriptstyle x are usually not given, but generated by an agent's interactions with the environment. At each point in time \scriptstyle t , the agent performs an action \scriptstyle y_t and the environment generates an observation \scriptstyle x_t and an instantaneous cost \scriptstyle c_t, according to some (usually unknown) dynamics. The aim is to discover a policy for selecting actions that minimizes some measure of a long-term cost; i.e., the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

### Learning algorithms

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a [Bayesian](http://en.wikipedia.org/wiki/Bayesian_probability) framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of [optimization](http://en.wikipedia.org/wiki/Mathematical_optimization) theory and [statistical estimation](http://en.wikipedia.org/wiki/Statistical_estimation).

Most of the algorithms used in training artificial neural networks employ some form of [gradient descent](http://en.wikipedia.org/wiki/Gradient_descent). This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a [gradient-related](http://en.wikipedia.org/wiki/Gradient-related) direction.

[Evolutionary methods](http://en.wikipedia.org/wiki/Evolutionary_methods), [gene expression programming](http://en.wikipedia.org/wiki/Gene_expression_programming), [simulated annealing](http://en.wikipedia.org/wiki/Simulated_annealing), [expectation-maximization](http://en.wikipedia.org/wiki/Expectation-maximization), [non-parametric methods](http://en.wikipedia.org/wiki/Non-parametric_methods) and [particle swarm optimization](http://en.wikipedia.org/wiki/Particle_swarm_optimization)[[40]](http://en.wikipedia.org/wiki/Artificial_neural_network#cite_note-40) are some commonly used methods for training neural networks.

**Types of Artificial neural network**

Artificial neural network types vary from those with only one or two layers of single direction logic, to complicated multi–input many directional feedback loops and layers. On the whole, these systems use algorithms in their programming to determine control and organization of their functions. Some may be as simple as a one-neuron layer with an input and an output, and others can mimic complex systems such as [dANN](http://en.wikipedia.org/wiki/List_of_artificial_intelligence_projects#Software_libraries), which can mimic chromosomal DNA through sizes at the cellular level, into artificial organisms and simulate reproduction, mutation and population sizes.

Most systems use "weights" to change the parameters of the throughput and the varying connections to the neurons. Artificial neural networks can be autonomous and learn by input from outside "teachers" or even self-teaching from written-in rules.

**Character recognition with neural network**

The neural net approach utilized three separate steps. The first step simply translated the binary character data into a friendlier form. The second step took the output of the first and trained a backpropagation network on it, outputting all the resulting weights and general network information. The third step took the output of the second and created a network. It then ran a full character set through the network and output identification information for all the characters the set contained. The reasons for implementing theh neural net OCR as three programs were all practical. By keeping the first step separate, the preprocessing code from the feature extraction OCR program could be used, eliminating this one area of difference between the two algorithms. The second step was separated just because learning was such a slow process. Several machines could thus be dedicated to nothing but learning while a different machine was used to analyze the results.

The network consisted of sixty-four inputs, ninety-six hidden nodes, and seven outputs. It was essentially a flat feedforward network that was fully connected without self-inputs or biases.It was made to train on the same character set that the feature extraction algorithm had used as its dictionary.Learning was achieved through backpropagation without momentum.

**6.2.2 PROCESS MODEL**

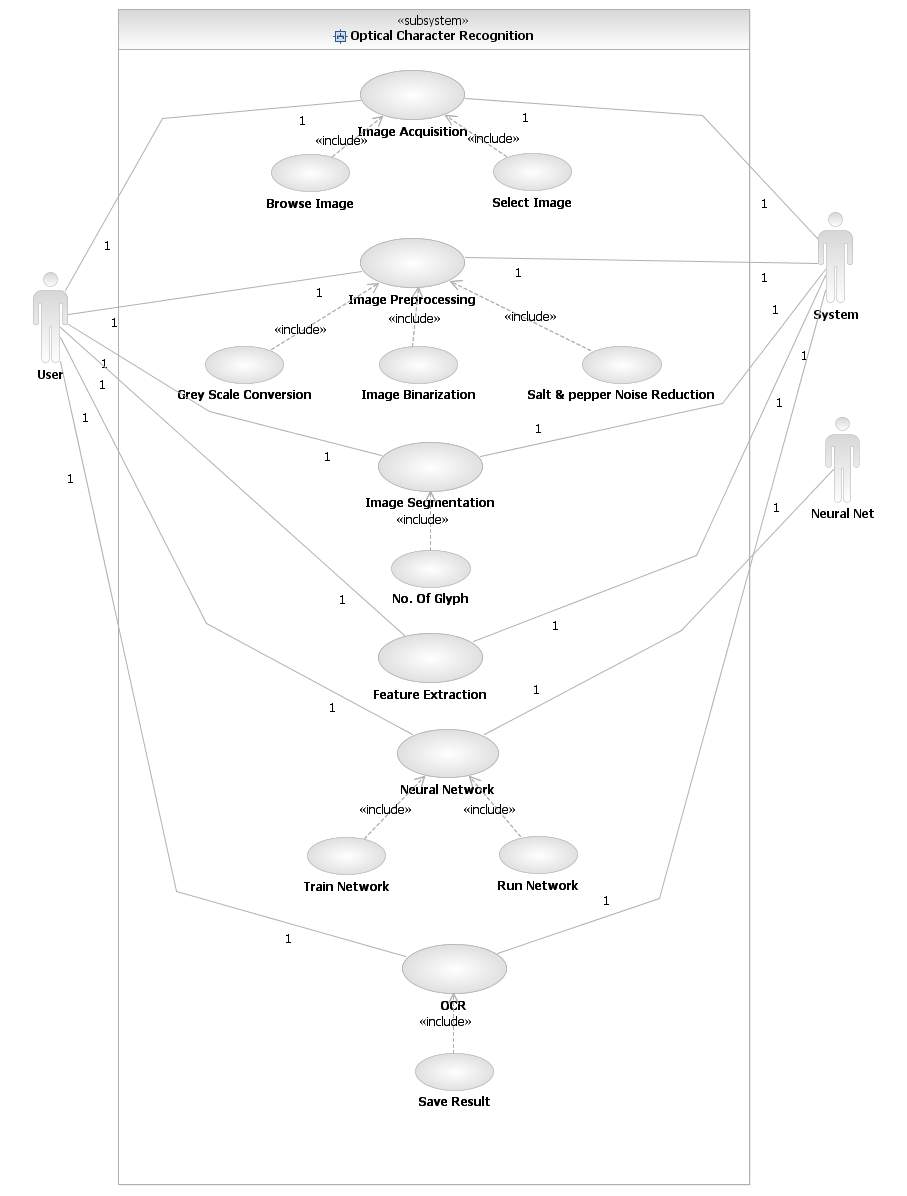
Our project will be built on the waterfall mode. This model suggests work cascading from step to step like a series of waterfalls. It consists of the following steps in the following manner



Figure(6.2.e)-Process Model

**6.3 UML DIAGRAMS**

**6.3.1 USE CASE ANALYSIS**



Figure(6.3 a)-Use Case Diagram

Use Case Analysis identifies the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis Phase of software development to articulate the high-level requirements of the system.

|  |  |  |
| --- | --- | --- |
| **Actor** | An Actor, as mentioned, is a user of the system, and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor. | http://res.dotnetcoders.com/images/uml/usecase-actor.png |
|  | A Use Case is functionality provided by the system, typically described as verb+object (e.g. Register Car, Delete User). Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse. | http://res.dotnetcoders.com/images/uml/usecase-usecase.png |
| **Association** | Associations are used to link Actors with Use Cases, and indicate that an Actor participates in the Use Case in some form. Associations are depicted by a line connecting the Actor and the Use Case. | http://res.dotnetcoders.com/images/uml/usecase-association.png |

**Include**

An include relationship is a relationship between two use cases

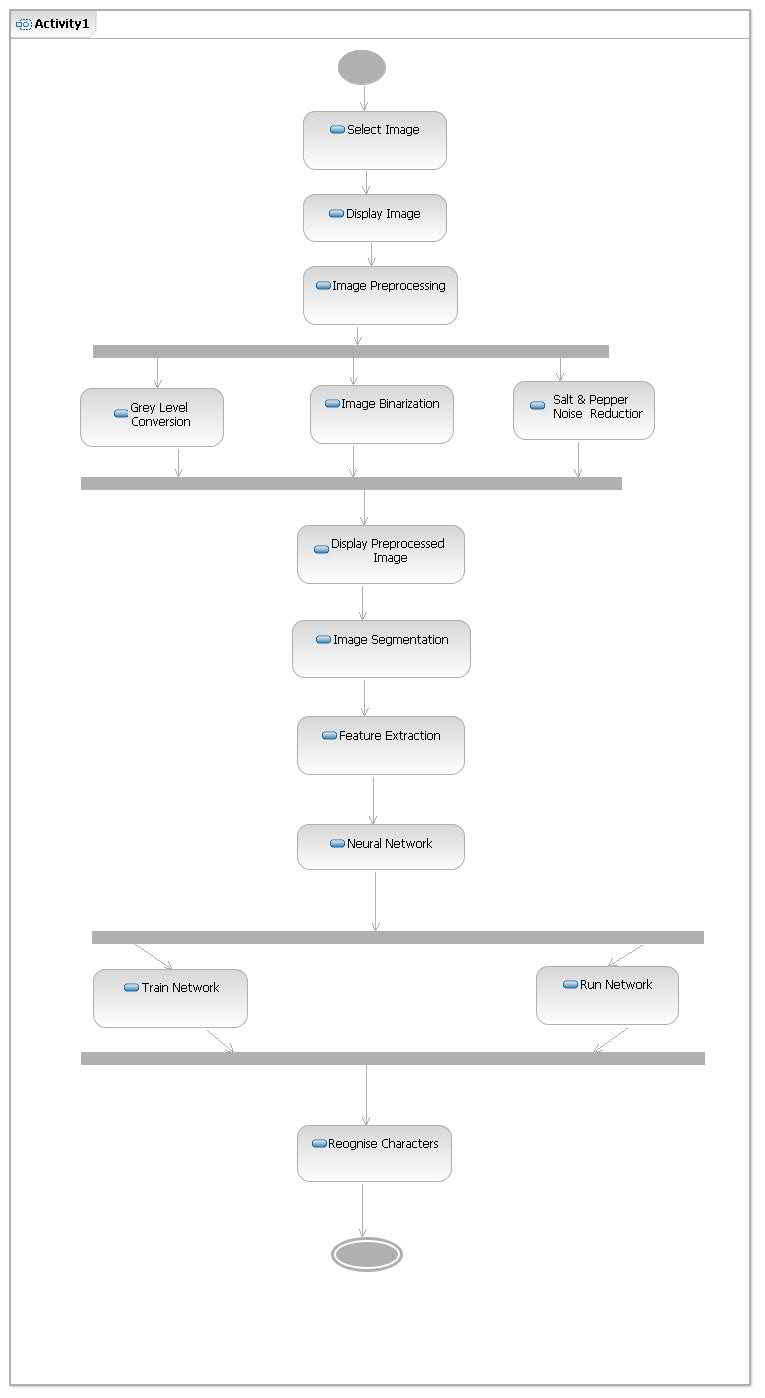
http://sourcemaking.com/files/sm/images/uml/img_95.jpg

**Extend**

An extend relationship is a relationship between two use cases

C:\Users\System Acc\Downloads\24.PNG

**6.3.2 ACTIVITY DIAGRAM**



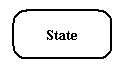
Figure(6.3 b)-Activity Diagram

Activity diagram provides basic idea about various activities through which our system goes.

**COMPONENTS:**

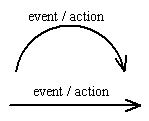
**States**

States represent situations during the life of an object. You can easily illustrate a state in SmartDraw by using a rectangle with rounded corners.



**Transition**

A solid arrow represents the path between different states of an object.Transition with the event that triggered it and the action that results from it.



**Initial State**

A filled circle followed by an arrow represents the object's initial state

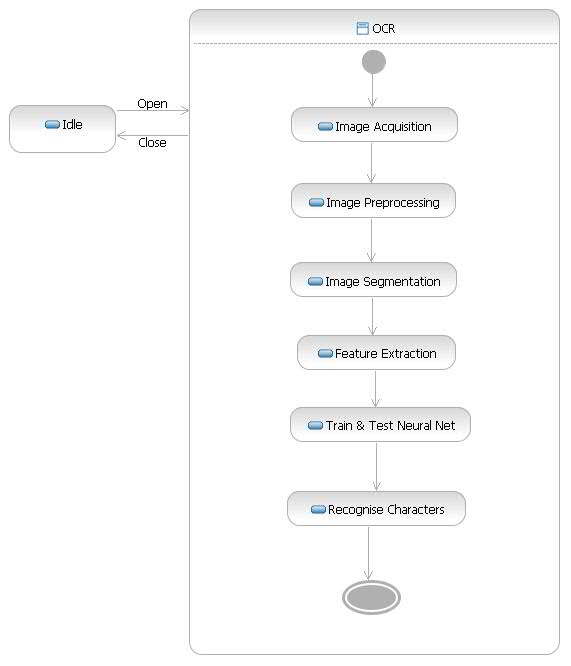
Initial State

**Final State**

An arrow pointing to a filled circle nested inside another circle represents the object's final state.

Final State

**6.3.3 State diagram**

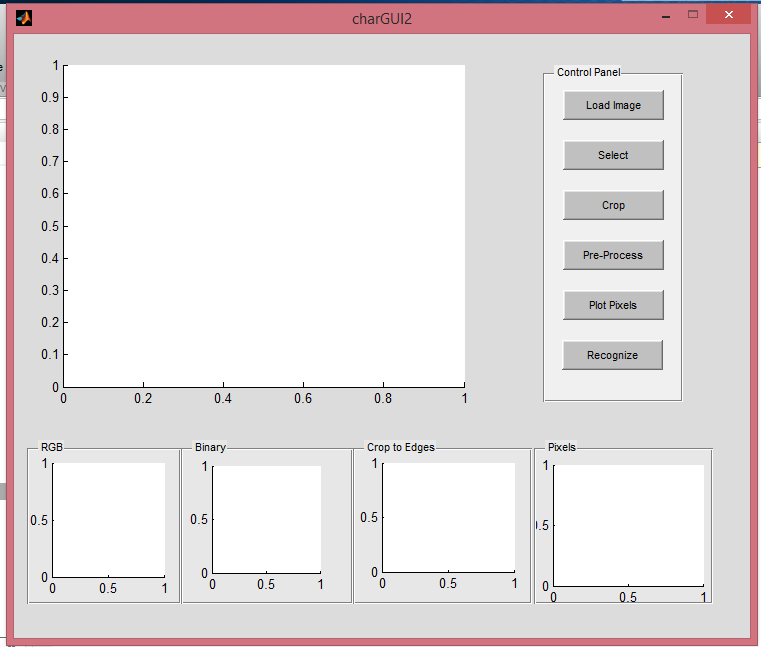
****

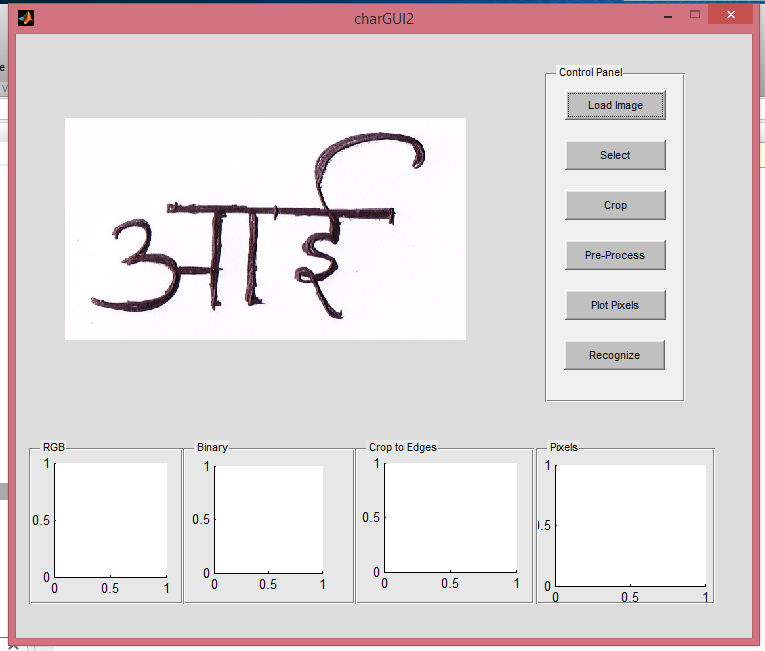
Figure(6.3 c)-State Diagram

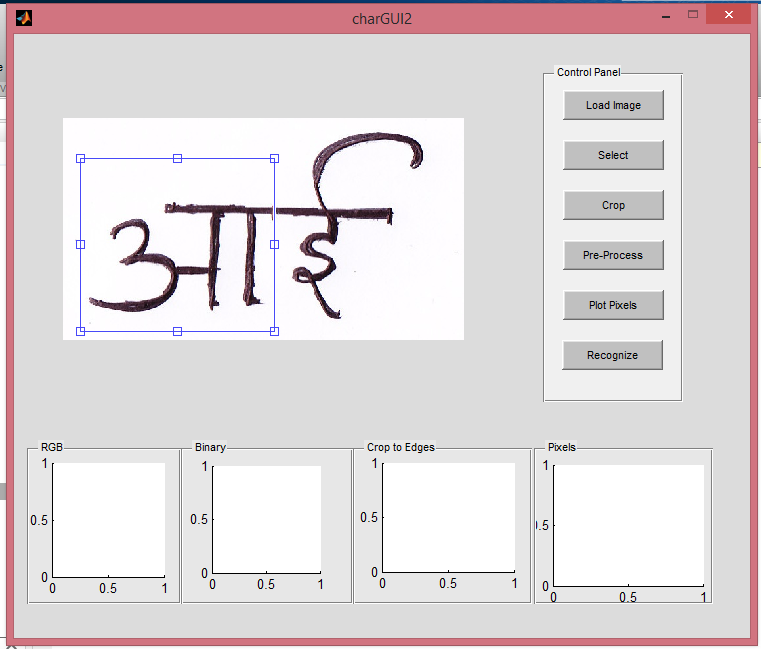
**Chapter-7**

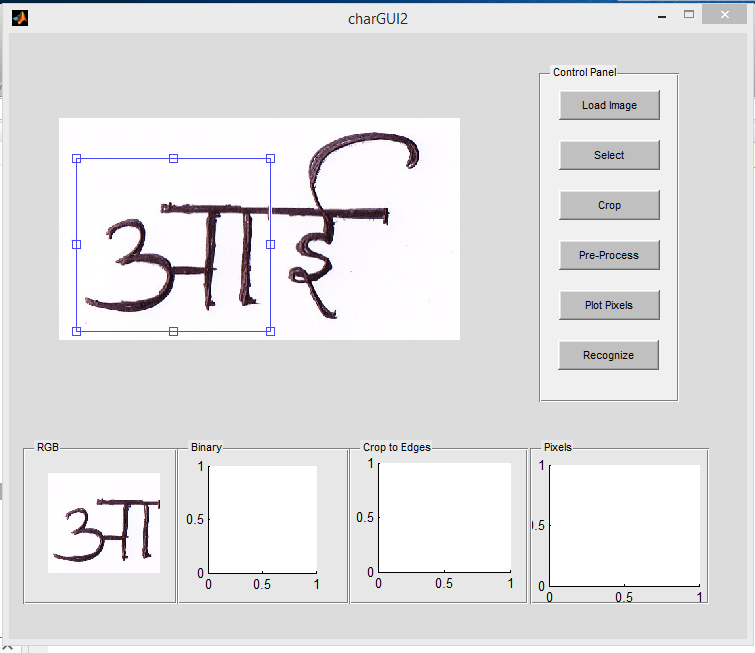
**IMPLEMENTATION DETAILS**

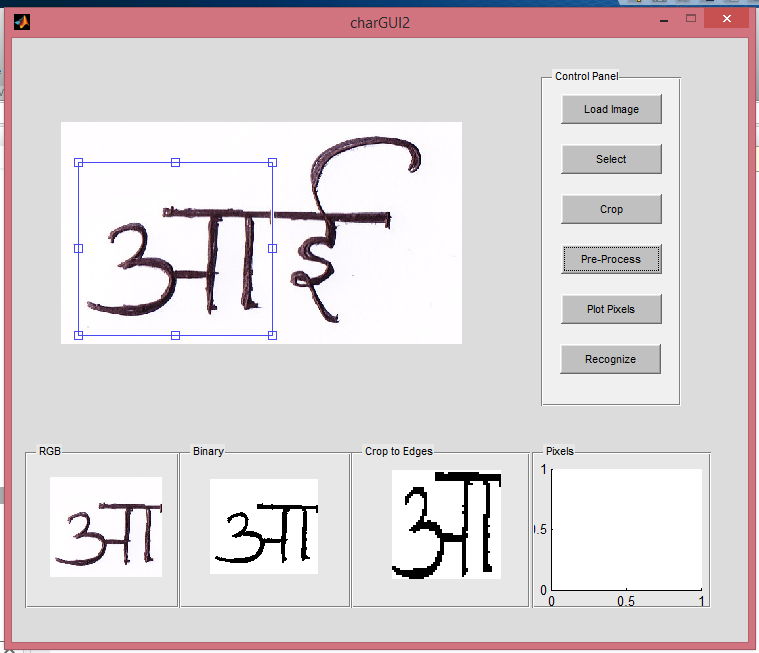
**7.1 INTERFACE DESIGN**

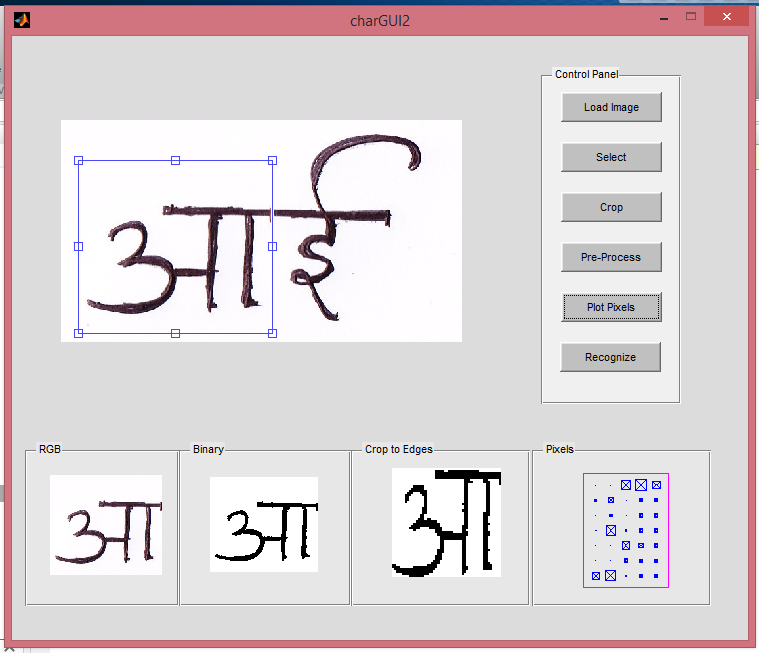
****

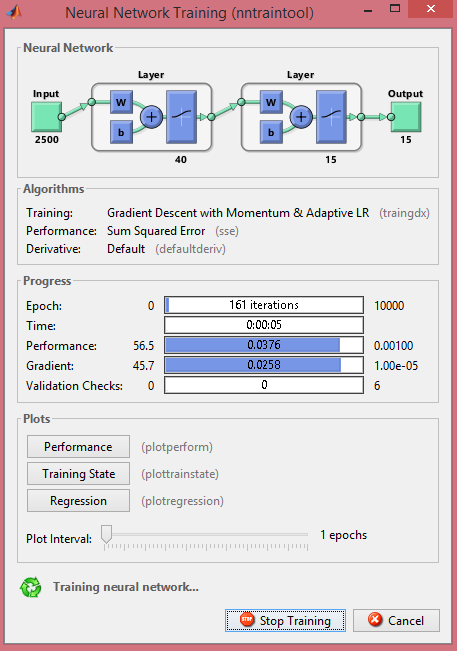
****

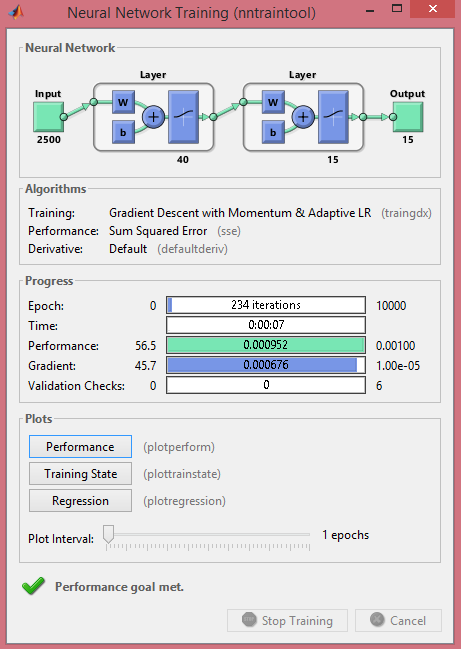
****

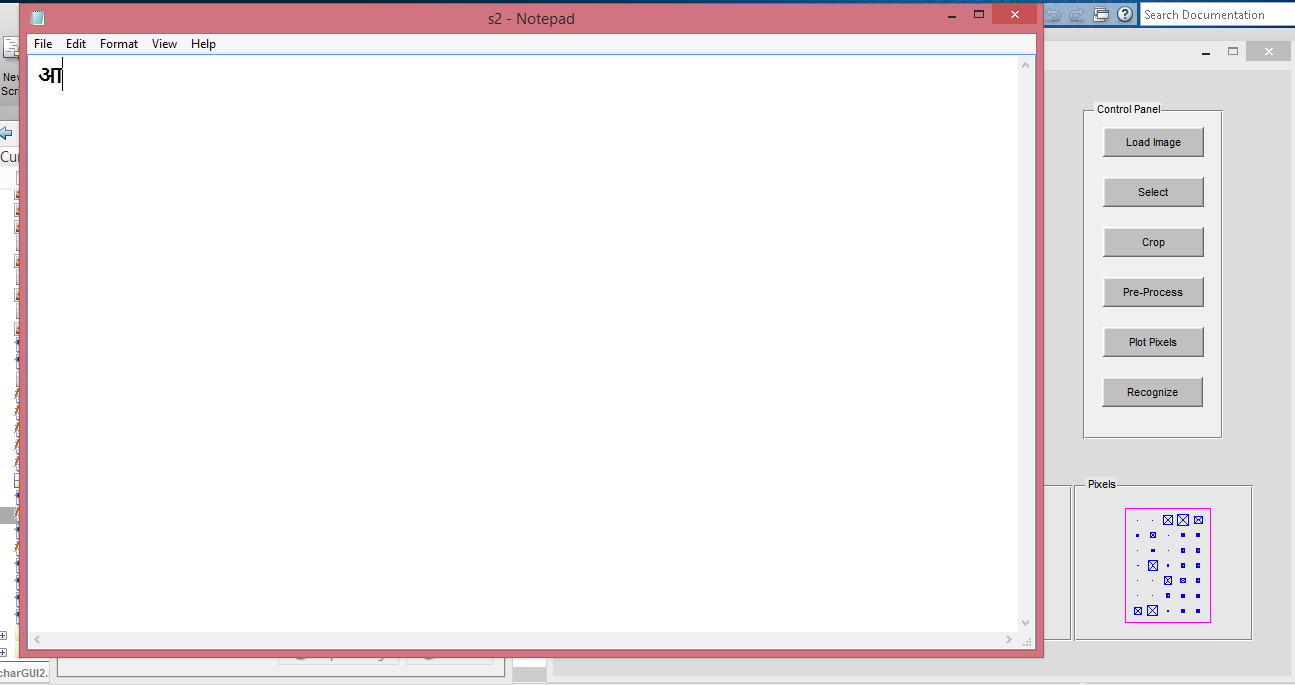
****

****

****

****

****

****

**7.2 SAMPLE CODE**

**7.2.1 Image Input**

[filename, pathname] = uigetfile({'\*.jpg';'\*gif';'\*.bmp';'\*.\*'}, 'Pick an Image File');

S = imread([pathname,filename]);

axes(handles.axes1);

imshow(S);

**7.2.2 Image Cropping**

axes(handles.axes1);

S = handles.S;

handles.img\_crop = imcrop(S,handles.loc);

axes(handles.axes2);

**7.2.3 Image Preprocessing**

img\_crop = handles.img\_crop;

imgGray = rgb2gray(img\_crop);

bwa= im2bw(img\_crop,graythresh(imgGray));

bw2=imresize((edu\_imgcrop(bwa)),[50 50]);

**7.2.4 Plotting pixel positions**

bw2 = handles.bw2;

charvec = edu\_imgresize(bw2);

axes(handles.axes5);

%charvec=imresize(charvec1,[42 42]);

plotchar(charvec);

**7.2.5 Neural Network Simulation & Recognition**

net = newff(minmax(le),[40 15],{'logsig' 'logsig'},'traingdx');

net.LW{2,1} = net.LW{2,1}\*0.01;

net.b{2} = net.b{2}\*0.01;

net.performFcn = 'sse';

net.trainParam.goal = 0.001;

net.trainParam.show = 20;

net.trainParam.epochs = 10000;

net.trainParam.mc = 0.95;

net.trainParam.mc=0.1

net = train(net,le,t);

Y = sim(net,le);

cor=0;

for i=1:15

max=1;

for j=1:15

if Y(j,i)>Y(max,i)

max=i;

end

end

if max==i

cor=cor+1;

end

end

a=bw2

a1=[];

for i=1:50

for j=1:50

a1=[a1;a(i,j)];

end

end

a2=sim(net,a1);

'Output vector for test image'

a2

max=1;

for j=1:15

if a2(j)>a2(max)

max=j;

% x=j;

end

end

**Chapter-8**

**TECHNOLOGIES USED**

**8.1 Matlab R2007b**

**MATLAB** (**mat**rix **lab**oratory) is a [multi-paradigm](http://en.wikipedia.org/wiki/Multi-paradigm_programming_language) [numerical computing](http://en.wikipedia.org/wiki/Numerical_analysis) environment and [fourth-generation programming language](http://en.wikipedia.org/wiki/Fourth-generation_programming_language). Developed by [MathWorks](http://en.wikipedia.org/wiki/MathWorks), MATLAB allows [matrix](http://en.wikipedia.org/wiki/Matrix_(mathematics)) manipulations, plotting of [functions](http://en.wikipedia.org/wiki/Function_(mathematics)) and data, implementation of[algorithms](http://en.wikipedia.org/wiki/Algorithm), creation of [user interfaces](http://en.wikipedia.org/wiki/User_interface), and interfacing with programs written in other languages, including [C](http://en.wikipedia.org/wiki/C_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B), [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), and [Fortran](http://en.wikipedia.org/wiki/Fortran).

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the [MuPAD](http://en.wikipedia.org/wiki/MuPAD) [symbolic engine](http://en.wikipedia.org/wiki/Computer_algebra_system), allowing access to [symbolic computing](http://en.wikipedia.org/wiki/Symbolic_computing) capabilities. An additional package, [Simulink](http://en.wikipedia.org/wiki/Simulink), adds graphical multi-domain simulation and [Model-Based Design](http://en.wikipedia.org/wiki/Model_based_design) for [dynamic](http://en.wikipedia.org/wiki/Dynamical_system) and [embedded systems](http://en.wikipedia.org/wiki/Embedded_systems).

**Features of MATLAB**

* High-level language for numerical computation, visualization, and application development
* Interactive environment for iterative exploration, design, and problem solving
* Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
* Built-in graphics for visualizing data and tools for creating custom plots
* Development tools for improving code quality and maintainability and maximizing performance
* Tools for building applications with custom graphical interfaces
* Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft® Excel®

**Functions Supported by MATLAB**

**Variables:** Variables are defined using the assignment operator, =. MATLAB is a weakly typed programming language because types are implicitly converted. It is a dynamically typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change.

**Vectors/matrices:** A simple array is defined using the colon syntax: *init*:*increment*:*terminator*.

**Structures:** MATLAB has structure data types. Since all variables in MATLAB are arrays, a more adequate name is "structure array", where each element of the array has the same field names. In addition, MATLAB supports dynamic field names.

### Function handles: MATLAB supports elements of [lambda calculus](http://en.wikipedia.org/wiki/Lambda_calculus) by introducing function handles or function references, which are implemented either in .m files or anonymousnested functions.

**Classes:** Although MATLAB has classes, the syntax and calling conventions are significantly different from other languages. MATLAB has value classes and reference classes, depending on whether the class has *handle* as a super-class (for reference classes) or not (for value classes).

**Graphical user interface and programming:** MATLAB supports developing applications with [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) features. MATLAB includes GUIDE (GUI development environment) for graphically designing GUIs. It also has tightly integrated graph-plotting features. For example the function *plot* can be used to produce a graph from two vectors *x* and *y*

**8.2 Artificial Neural Network:**

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.  
Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.
5. Training Algorithms for Neural Network
6. Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins.
7. There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help.
8. The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs. However, in the full blown sense of being truly self learning, it is still just a shining promise that is not fully understood, does not completely work, and thus is relegated to the lab.

### Supervised Training.

1. In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.
2. The current commercial network development packages provide tools to monitor how well an artificial neural network is converging on the ability to predict the right answer. These tools allow the training process to go on for days, stopping only when the system reaches some statistically desired point, or accuracy. However, some networks never learn. This could be because the input data does not contain the specific information from which the desired output is derived. Networks also don't converge if there is not enough data to enable complete learning. Ideally, there should be enough data so that part of the data can be held back as a test. Many layered networks with multiple nodes are capable of memorizing data. To monitor the network to determine if the system is simply memorizing its data in some nonsignificant way, supervised training needs to hold back a set of data to be used to test the system after it has undergone its training. (Note: memorization is avoided by not having too many processing elements.)
3. If a network simply can't solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs.
4. Another part of the designer's creativity governs the rules of training. There are many laws (algorithms) used to implement the adaptive feedback required to adjust the weights during training. The most common technique is backward-error propagation, more commonly known as back-propagation. These various learning techniques are explored in greater depth later in this report.
5. Yet, training is not just a technique. It involves a "feel," and conscious analysis, to insure that the network is not overtrained. Initially, an artificial neural network configures itself with the general statistical trends of the data. Later, it continues to "learn" about other aspects of the data which may be spurious from a general viewpoint.
6. When finally the system has been correctly trained, and no further learning is needed, the weights can, if desired, be "frozen." In some systems this finalized network is then turned into hardware so that it can be fast. Other systems don't lock themselves in but continue to learn while in production use.

### Unsupervised, or Adaptive Training.

1. The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption.
2. At the present time, unsupervised learning is not well understood. This adaption to the environment is the promise, which would enable science fiction types of robots to continually learn on their own as they encounter new situations and new environments. Life is filled with situations where exact training sets do not exist. Some of these situations involve military action where new combat techniques and new weapons might be encountered. Because of this unexpected aspect to life and the human desire to be prepared, there continues to be research into, and hope for, this field. Yet, at the present time, the vast bulk of neural network work is in systems with supervised learning. Supervised learning is achieving results.
3. One of the leading researchers into unsupervised learning is Tuevo Kohonen, an electrical engineer at the Helsinki University of Technology. He has developed a self-organizing network, sometimes called an auto-associator, that learns without the benefit of knowing the right answer. It is an unusual looking network in that it contains one single layer with many connections. The weights for those connections have to be initialized and the inputs have to be normalized. The neurons are set up to compete in a winner-take-all fashion.
4. Kohonen continues his research into networks that are structured differently than standard, feedforward, back-propagation approaches. Kohonen's work deals with the grouping of neurons into fields. Neurons within a field are "topologically ordered." Topology is a branch of mathematics that studies how to map from one space to another without changing the geometric configuration. The three-dimensional groupings often found in mammalian brains are an example of topological ordering.
5. Kohonen has pointed out that the lack of topology in neural network models make today's neural networks just simple abstractions of the real neural networks within the brain. As this research continues, more powerful self-learning networks may become possible.

## Backpropagation Learning Algorithm

1. The backpropagation algorithm trains a given feed-forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted. The backpropagation algorithm is based on Widrow-Hoff delta learning rule in which the weight adjustment is done through mean square error of the output response to the sample input. The set of these sample patterns are repeatedly presented to the network until the error value is minimized.
2. Refer to the figure 2 below that illustrates the backpropagation multilayer network with $ M$layers. $ N_j$represents the number of neurons in $ j$th layer. Here, the network is presented the $ p$th pattern of training sample set with $ N_0$-dimensional input $ X_{p1}, X_{p2},
   ... , X_{pN_0}$and $ N_M$-dimensional known output response $ T_{p1}, T_{p2},
   ... , T_{pN_M}$. The actual response to the input pattern by the network is represented as $ O_{p1}, O_{p2},$$ ... , O_{pN_M}$. Let $ Y_{ji}$be the output from the $ i$th neuron in layer $ j$for $ p$th pattern; $ W_{jik}$be the connection weight from $ k$th neuron in layer $ (j-1)$to $ i$th neuron in layer $ j$; and $ \delta_{ji}$be the error value associated with the $ i$th neuron in layer $ j$.

|  |
| --- |
| \begin{figure} \centerline {\epsfysize=4.0in \epsfbox{./figures/figBackprop.epsi}}\end{figure} |
| **Figure (8.2)-** Backpropagation Neural Network |

1. Steps to follow until error is suitably small
2. Step 1: Input training vector.  
   Step 2: Hidden nodes calculate their outputs.  
   Step 3: Output nodes calculate their outputs on the basis of Step 2.  
   Step 4: Calculate the differences between the results of Step 3 and targets.  
   Step 5: Apply the first part of the training rule using the results of Step 4.  
   Step 6: For each hidden node, n, calculate d(n).  
   Step 7: Apply the second part of the training rule using the results of Step 6.
3. Steps 1 through 3 are often called the *forward pass*, and steps 4 through 7 are often called the *backward pass*. Hence, the name: back-propagation.

**Chapter-9**

**TEST CASES**

**Testing Methods**

System testing is a critical phase implementation. Testing of the system involves hardware devise and debugging of the computer programs and testing information processing procedures. Testing can be done with text data, which attempts to stimulate all possible conditions that may arise during processing. If structured programming Methodologies have been adopted during coding the testing proceeds from higher level to lower level of program module until the entire program is tested as unit. The testing methods adopted during the testing of the system were unit testing and integrated testing.

**Unit testing:*-***

Unit testing focuses first on the modules, independently of one another, to locate errors. This enables the tester to detect errors in coding and logical errors that is contained within that module alone. Those resulting from the interaction between modules are initially avoided.

**Integration testing:***-*

Integration testing is a systematic technique for constructing the program structure while at the same time to uncover the errors associated with interfacing. The objective is to take unit-tested module and build a program structure that has been detected by designing. It also tests to find the discrepancies between the system and its original objectives. Subordinate stubs are replaced one at time actual module. Tests were conducted at each module was integrated. On completion of each set another stub was replaced with the real module.

**System testing:-**

**System testing** of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified [requirements](http://en.wikipedia.org/wiki/Requirements). System testing falls within the scope of [black box testing](http://en.wikipedia.org/wiki/Black_box_testing), and as such, should require no knowledge of the inner design of the code or logic.

The following are the test cases in Optical Character Recognition

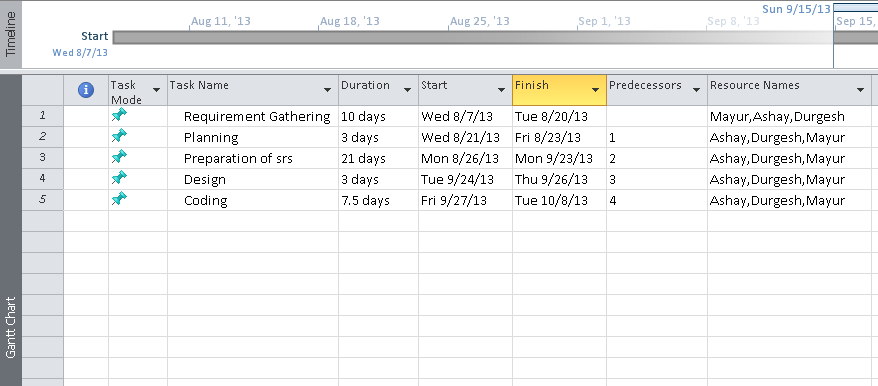
**Unit testing**

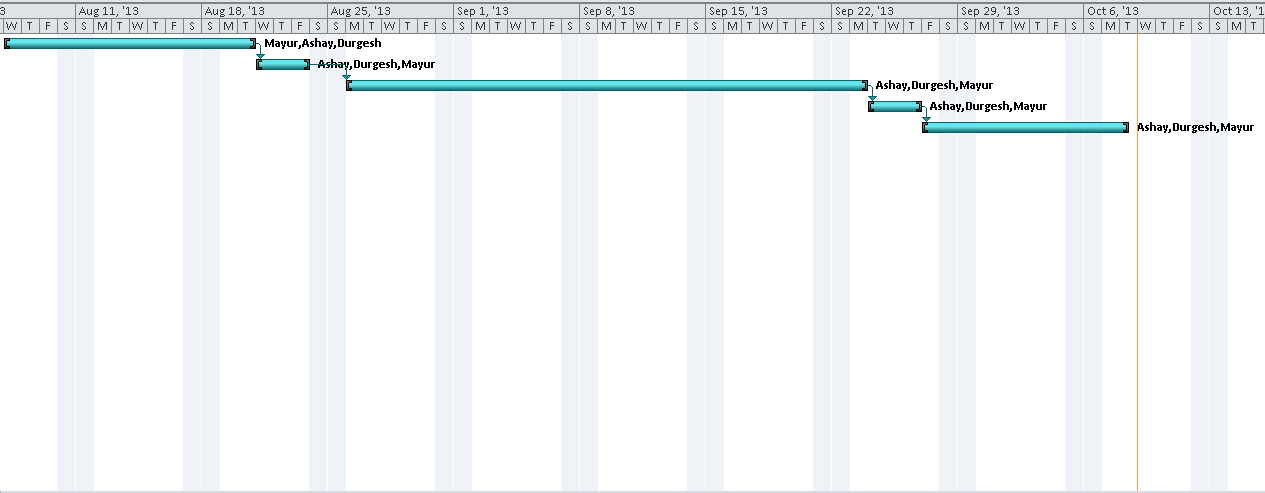
**Test Objective:** To test the recognition capability.

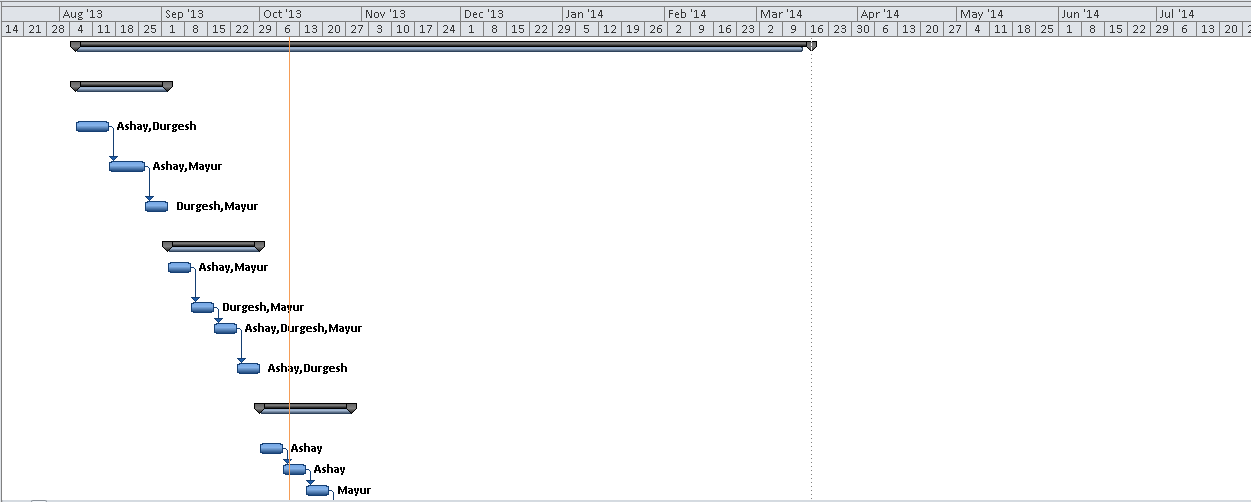
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item No** | **Test Condition** | **Description** | **Input Specification** | **Output Specification (Expected Results)** | **Pass or Fail** | |
| 1 | Generation of training data set | Collecting the training data for neural network | Number of training images in bitmap format | If all the images i.e. training data is in monochrome bitmap format then it can be directly used for training the neural network | Pass | |
| 2 | Preprocessing of training data | Training data is preprocessed to remove some addictive noise from it as well as to crop it in to the specified predefined size | Appropriate amount of training data for better recognition accuracy | Training data is now noise free and all the training characters are cropped to the edges | Pass | |
| 3 | Neural network simulation | Training data needs to be passed through 3 basic layers of feedforword backpropogation network | An instance of neural network and stored collection of training data | Successful simulation of neural network with all specified training data | Pass | |
| 4 | Successful recognition of characters | Characters from the images are recognized accurately | Training data as well as test data | Successfully recognition of character by comparing it with all the stored training data by using feedforword  Backpropogation  Neural network | Pass |

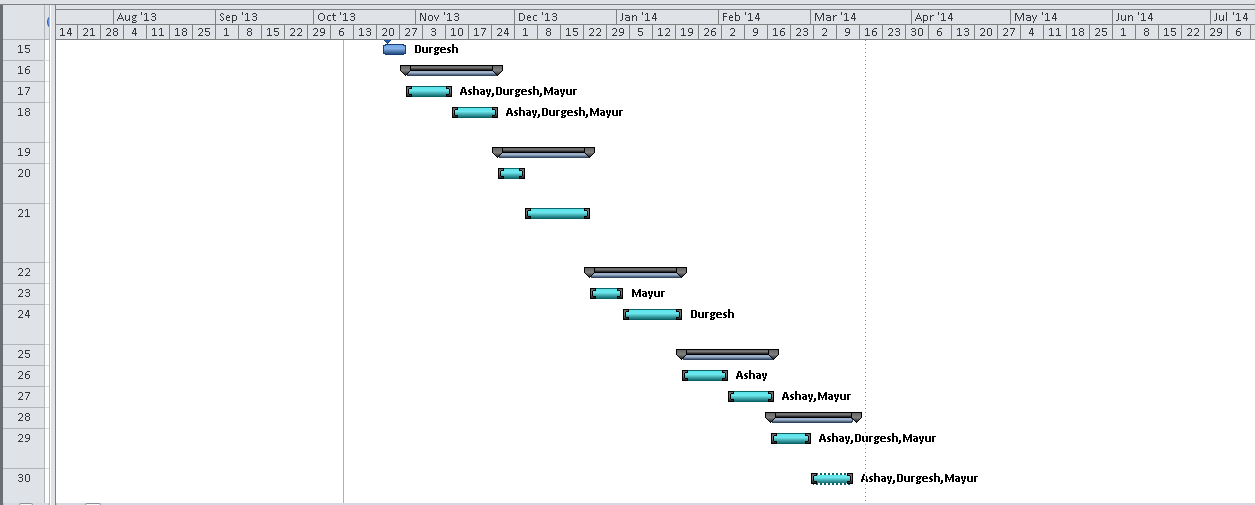
**Chapter-10**

**PROJECT TIME LINE**

****

****

****

****

**Chapter-11**

**TASK DISTRIBUTION**

**W.B.S. CHART**

1.5.6 Black Box Testing

1.5.5 White Box Testing

1.5.4 Test Execution

1.4.5 Code Character Recognition Module

1.4.4 Code Feature Extraction Module

1.1.8 Specify Project Goals

1.1.7 UML Diagrams

1.1.6 Character Recognition

1.1.5 Feature Extraction

1.3.4 Development Methodology

1.1.4 Segment Image

1.2.3 Black Book

1.3.3 Design Module in Detail

1.4.3 Code Segment Module

1.5.3 Setup Test Environment

1.1.3 Preprocess Image

1.2.2 Software Requirement Specification

1.3.2 Identify Design Modules

1.4.2 Code Preprocess Module

1.5.2 Create Test Data

1.1.2 Load Image

1.2.1 Synopsis

1.5.1 Preparation of Test Plan

1.4.1 Frontend and Backend

1.3.1 Identify Design Pattern

1.1 Analysis

1.2 Documentation

1.5 Testing

1.4 Coding

1.3 Design

Optical Character Recognition

**Chapter-12**

1.5.7 Implementation and Maintainence

**CONCLUSION AND FUTURE WORK**

The “Optical Character Recognition” system can able to recognize both handwritten as well as printed “Devanagari script” characters from the digital image. The recognized characters are automatically get copied into the text file so user can able to perform some basic text editing operations like cut, copy, paste etc. The performance of system goes on increasing as we increases the system training. Implementation of FeedForword Backpropogation Neural Network algorithm makes our system more accurate and faster. Results are mostly accurate and in case of low recognition accuracy the overall performance can be handled by training system repeatedly.

Various techniques that will make our system more functional and reliable,which can be considered as a future work for our system are as follows.

* Voice Synthesis – This technology can also be used along with Optical Character Recognition System so that recognized characters will be available in an audio format to the user which will be more efficient and useful for physically disabled users.
* Language Translation – Recognized characters can be easily translated from one language to another by using language translator engine like “Google Translator”.
* Multilingual OCR – It is possible to build a system which is capable of recognizing characters from almost all languages available in world. However such system is very complex to build and requires so many efforts to have a good recognition accuracy.

**Chapter-13**

**REFERENCES**

[1] Optical Character Recognition (OCR) for Printed Devnagari Script Using Artificial Neural Network - International Journal of Computer Science & CommunicationVol. 1, No. 1, January-June 2010, pp. 91-95

[2] OCR for Script Identification of Hindi (Devnagari) Numerals using Feature Sub Selection by Means of End-Point with Neuro-Memetic Model by Banashree N. P., and R. Vasanta - International Journal of Electrical and Computer Engineering 2:7 2007

[3] Recognition of Handwritten Devnagari Characters through Segmentation and Artificial neural networks - International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 6, August - 2012 ISSN: 2278-0181

[4] Text book on Neural Network for Pattern Recognition by – Christopher M. Bishop

[5] E. W. Brown, ["Character Recognition by Feature Point Extraction"](http://www.ccs.neu.edu/home/feneric/charrec.html), unpublished paper written at Northeastern University, 1992

[6] The Commodore 128 Computer, Commodore Business Machines, 1984

[7] Stephen I. Gallant, Chapters 11 & 12; Neural Network Learning and Expert Systems, 1993, The MIT Press, Cambridge, MA

[8] D. S. Levine, pp. 58-62; Introduction to Neural & Cognitive Modeling, 1991, Lawrence Erlbaum Associates, Publishers; Hillsdale, NJ

[9] Patrick K. Simpson, pp. 112-123, Artificial Neural Systems, 1990, Pergamon Press, New York, NY

[10] Handwritten Bangla Character Recognition Using Neural Network- International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 11, November 2012,ISSN: 2277 128X

**Chapter-14**

**APPENDIX**

|  |  |
| --- | --- |
| ANN | Artificial Neural Networks |
| ANSI | American National Standards Institute |
| BPM  DPI | Bitmap  Dots Per Inch |
| ECMA | European Computer Manufacturers Association |
| FFNN | Feedforward Neural Network |
| Gb | Giga byte |
| GHZ | Giga Hertz |
| GIF  ISO | Graphic Interchange Format  International Standards Organization |
| JPEG/JPG  MATLAB | Joint Photographic Expert Group  Matrix Laboratory |
| M-files | Matlab Files |
| NN | Neural Networks |
| NNT | Neural Networks Toolbox |
| OCR  OS | Optical Character Recognition  Operating System |
| PDF | Portable Document  Format |
| RTF | Rich Text Format |
| TIFF | Tagged Image File Format |
|  |  |
|  |  |