# ABSTRACT

Handwritten Recognition is the process of text extraction from images of handwritten text. Devnagari numeral string is the collection of devnagari digits forming a number. In this project handwritten Devnagari numeral string recognition is done using Multi-layer Perceptron Neural Network (MLP). Handwriting Recognizer a system that locates and recognizes devnagari numeral strings written in a white paper.

The classifier used to recognize the Devnagari digit is Multi-Layer Perceptron Neural Network that has 1028(32\*32) input nodes, 300 hidden nodes and 10 output nodes. First, the Neural Network is trained with 1700 data sets of each digit (0-9). The images of handwritten text is preprocessed and then fed to neural networks for recognition.

The obtained result gives 96.3 % accuracy with a learning rate of 0.3.

Keywords**:** Multi-layer Perceptron Neural Network, Back Propagation algorithm, Nepali digits recognition, devnagari digit recognition, handwritten digit recognition

**TABLE OF CONTENTS**

[LETTER OF APPROVAL i](#_Toc489479714)

[ACKNOWLEDGEMENT ii](#_Toc489479715)

[STUDENT’S DECLARATION iii](#_Toc489479716)

[ABSTRACT iv](#_Toc489479717)

[LIST OF FIGURES vii](#_Toc489479718)

[LIST OF TABLES viii](#_Toc489479719)

[LIST OF ABBREVIATIONS ix](#_Toc489479720)

[CHAPTER 1: INTRODUCTION 1](#_Toc489479721)

[1.1 Background 1](#_Toc489479722)

[1.2 Problem Statement 2](#_Toc489479723)

[1.3 Objectives 2](#_Toc489479724)

[1.4 Scope 2](#_Toc489479725)

[1.5 Limitation 2](#_Toc489479726)

[1.6 Outline of Document 3](#_Toc489479727)

[CHAPTER 2: REQUIREMENT AND FEASIBILITY ANALYSIS 4](#_Toc489479728)

[2.1 Literature Review 4](#_Toc489479729)

[2.1.1 Devnagari Numeral String Recognition 4](#_Toc489479730)

[2.1.2 Different approaches for numeral string recognition 4](#_Toc489479731)

[2.1.3 Use of Multilayer Percepton Neural Network 5](#_Toc489479732)

[2.2 Requirement Analysis 6](#_Toc489479733)

[Functional requirement 6](#_Toc489479734)

[Non-functional requirement 6](#_Toc489479735)

[2.3 Feasibility Analysis 6](#_Toc489479736)

[2.3.1 Technical feasibility 6](#_Toc489479737)

[2.3.2 Operational feasibility 6](#_Toc489479738)

[2.3.3 Schedule feasibility 7](#_Toc489479739)

[CHAPTER 3: SYSTEM DESIGN 8](#_Toc489479740)

[3.1 Methodology 8](#_Toc489479741)

[3.1.1 Data collection 8](#_Toc489479742)

[3.1.2 Data Information 9](#_Toc489479743)

[3.1.3 Data preprocessing 10](#_Toc489479744)

[3.1.4 Training Neural Network 11](#_Toc489479745)

[3.2 Learning Algorithm 12](#_Toc489479746)

[3.3 System Design 14](#_Toc489479747)

[3.3.1 Class diagram 14](#_Toc489479748)

[3.3.2 State diagram 15](#_Toc489479749)

[3.3.3 Sequence diagram 16](#_Toc489479750)

[CHAPTER 4: IMPLEMENATION AND TESTING 17](#_Toc489479751)

[4.1 Implementation 17](#_Toc489479752)

[4.1.1 Tools used 17](#_Toc489479753)

[4.2 Description of Major Classes 18](#_Toc489479754)

[4.2.1 Segment 18](#_Toc489479755)

[4.2.2 NeuralNetwork 18](#_Toc489479756)

[4.2.3 Main 18](#_Toc489479757)

[4.3 Steps 19](#_Toc489479758)

[4.3.1 Thresholding 19](#_Toc489479759)

[4.3.2 Connected Component Analysis 20](#_Toc489479760)

[4.4 Testing 22](#_Toc489479761)

[CHAPTER 5: MAINTENANCE AND SUPPORT PLAN 24](#_Toc489479762)

[5.1 Maintenance Plan 24](#_Toc489479763)

[5.2 Support Plan 24](#_Toc489479764)

[CHAPTER 6: CONCLUSION AND RECOMMENDATION 25](#_Toc489479765)

[6.1 Conclusion 25](#_Toc489479766)

[6.2 Recommendation 25](#_Toc489479767)

[APPENDIX 26](#_Toc489479768)

[References 27](#_Toc489479769)

# 

# LIST OF FIGURES

[Figure 2.1 Activity network diagram of the project 7](#_Toc489478845)

[Figure 3.1 Methodology 8](#_Toc489478846)

[Figure 3.2 Sample of data set images 9](#_Toc489478847)

[Figure 3.3 Sample of data set images 11](#_Toc489478848)

[Figure 3.4 Class Diagram of Handwritten Digits Recognizer 14](#_Toc489478849)

[Figure 3.5 State Diagram of the project 15](#_Toc489478850)

[Figure 3.6 Sequence Diagram of the project 16](#_Toc489478851)

[Figure 4.1 an image of handwritten numeral string 19](#_Toc489478852)

[Figure 4.2 Thresholding with value = 80(left) Vs Adaptive Thresholding(right) 19](#_Toc489478853)

[Figure 4.3 Negative image 20](#_Toc489478854)

[Figure 4.4 shows the removal of noise using median filter. 20](#_Toc489478855)

[Figure 4.5 Median blurred image 21](#_Toc489478856)

[Figure 4.6 Dilation 21](#_Toc489478857)

[Figure 4.7 Connected Component Analysis 21](#_Toc489478858)

[Figure 4.8 Segmentation of strings 22](#_Toc489478859)

[Figure 4.9 Segmentation of digits 22](#_Toc489478860)

[Figure 4.10 Resize to 32\* 32 size image 22](#_Toc489478861)

[Figure 4.11 Efficiency against learning rate 23](#_Toc489478862)

[Figure 4.12 Efficiency against number of hidden layers 23](#_Toc489478863)

# 

# LIST OF TABLES

[Table 2.1- Functional and non-functional requirements 6](#_Toc489422453)

# LIST OF ABBREVIATIONS

ANN: Artificial Neural Network

CSS: Cascading Style Sheet

MLP: Multi-Layer Perceptron

HTML: Hyper Text Markup Language

URL: Uniform Resource Locator

# CHAPTER 1: INTRODUCTION

Handwriting is a person’s individual style of writing. Handwritten digit recognition is the System’s ability to understand intelligible handwritten input from various sources such as paper documents, photographs, touch-screens and other devices. Handwritten digits recognition is carried out for English digits but there is very less work done in recognizing Nepali handwritten digits and characters.

Handwritten digits recognition is an automated system that assists in recognizing human written digits. This could be a lot of help in bank cheque verification, number plate recognition, ancient document digitization and many more.

Recognizing handwritten numeral string is a difficult problem for a computer due to unlimited number of styles, sizes and variation of digit patterns. Unlike machine printed document, the free-style handwritten numerical string of digits is often curved, have various skew angles, no uniform direction, connect or overlap with each other.

Neural Networks do not require any feature to be explicitly defined, instead they work on the raw pixel data generating the best features and using them to classify the inputs into different classes [1]. In this work, Multi-layer Perceptron Neural Network is used to recognize the image. Pre-processing includes the steps that are required to shape the input image into a form suitable for segmentation [2]. The image that to be feed in the neural network should be preprocessed and normalized to the data similar to trained data.

## Background

Written form of communication is the basic form of communication nowadays. Handwritten numeral strings are the combination of handwritten digits. Handwritten digits are free-style digits that can be cursive or separated. The writing style varies according to writer. The variation can be in size, inclination, style etc.

There are works carried out in isolated Nepali digit recognition that has 94.44 % of accuracy. But the work on recognizing strings of Nepali digit is not carried out till now. Character segmentation from cursive handwritten documents is a difficult task. So, in literature most of the researches were conducted on separated characters.

## Problem Statement

Document digitization gives us advantage of storing and retrieving documents. But digitization of document is lengthy-process by manual typing. It is difficult to digitize handwritten documents. The recognition of handwritten numeral string can be used in part of recognizing numbers and their string from the document.

Similarly, handwritten digit recognition can be used in bank verification, vehicle’s number plate recognition, postal code recognition etc. The use of this system can be beneficial to the users that are manually recognizing lot of numbers in a day.

## 1.3 Objectives

The objective of this project is to implement back propagation learning algorithm in multi-layer perceptron neural network to recognize strings of devnagari digits.

## 1.4 Scope

This system can be used as a part in document digitization, cheque verification, and number-plate recognition. This can be used by people who works with documented Nepali numbers.

## 1.5 Limitation

a) Strings with joined or overlapped digits are not recognized correctly

b) The background should be white

c) Only .png and .jpeg image format are supported

## 

## 1.6 Outline of Document

The document started from a preliminary section. The preliminary section of the document consists of a title page, an abstract, table of contents and the list of figures and tables used throughout the document.

The flow of the document in one; Introduction section discusses the basic overview of handwritten digit recognition, problem statement, objective of recognizing handwritten digits are stated.

Chapter two; Requirement and Feasibility Analysis Section contain the background and general information of handwritten Nepali numeral string. This chapter also reviews basic algorithms of image pre-processing and discusses different methods by previous researcher developing a recognizer for handwritten numeral string.

The third chapter; System Design Section discusses the overall system methodology process, which includes the pre-processing algorithms. Besides that, this chapter covers the algorithm used in the process and their thorough explanation followed by basic terms and theories of the system design consideration.

The fourth chapter; Implementation and Testing present the flow of the application and every major task implemented in the user interface. This portion also covers the major classes covered in the system and the tools used. It presents the results of the project, which includes training of the data in the neural network. And later testing phase is carried out by importing images of handwritten numbers and applying pre-processing stages on the imported images and then fitting those into the neural network in order to get accurate results.

The fifth chapter; Maintenance and Support Plan consist of future plans in order to keep the application up and running. This portion also talks about plans for sustainability of the application if it ever launches publicly.

The sixth chapter discussed the conclusion of this project and recommendation on further works and upgrades of the system.

# 

# CHAPTER 2: REQUIREMENT AND FEASIBILITY ANALYSIS

## 2.1 Literature Review

### 2.1.1 Devnagari Numeral String Recognition

Although handwritten recognition systems are quite stabilized, Devnagari script based handwritten recognition system has not completely reached the level of higher performance. In the year 1976, Sethi and Chatterjee has initiated the necessity of Devnagari based handwritten recognition system, many researchers are following the footsteps [3]. Devnagari script based Nepali isolated-numbers have been recognized by Ashok Kumar Pant from IOE with an accuracy of 94.44 percentage [1]. However, the work on recognizing Devnagari numeral strings is not carried out so far.

### 2.1.2 Different approaches for numeral string recognition

There are two approaches for handwritten numeral string recognition. One is that a numeral string is considered as a sequence of images of individual digits. From this point of view, recognizing a numeral string is similar to partitioning the image into those of individual digits. In the other approach, a digit is treated as a set of strokes and then basic operations are conducted at stroke level [4]. This project approach can be categorized as the first one because the digits can be grouped and segmented with image processing which is convenient and reduces pixels clustering. For recognizing the individual or segmented digits, either of the following methods can be used. They are using Multilayer perceptron neural, Gradient Features and SVM (Support vector machines) etc.

### 2.1.3 Use of Multilayer Percepton Neural Network

Neural Networks have been very useful in finding the patterns.

Neural Networks do not require any feature to be explicitly defined, instead they work on the raw pixel data generating the best features and using them to classify the inputs into different classes [1].

Subhashini, P. P. S., and V. V. K. D. V. Prasad et al [5] has proposed, a method based on Radial basis function (RBF). Neural network plays an important role in pattern classification problems. Training neural network was a challenging nonlinear optimization problem. Multiple algorithms have been premeditated for choosing the RBF neural network prototypes and used to train the network. The efficiency of the proposed methodology was tested on the handwritten digits of different fonts and found to be successful in recognizing the digits. This method is tested on handwritten digits of 0 to 9 of 25 different fonts. The success rate of this method for recognizing handwritten digit is viable.

Bottou, Léon, et al [6] this paper proposed the comparison of performance between several classifier algorithms on standard database of handwritten numerals. Here consider accuracy, training time, recognition time, and memory requirements. Here, used the LeNet 4 classifier which eliminates the redundant training examples which also reduce the size requirements of the memory-based classifiers that tested at the cost of increase run time. LeNet 4 is produce the feature vector which is significant variation in the training time and it is performs recognition at 1000 characters. The recognizer is designed to train for find out the correct digit and also correct segmentation. Neural Network is used for recognition and it has advantage over memory-based technique.

## 2.2 Requirement Analysis

Table 2.1- Functional and non-functional requirements

|  |  |
| --- | --- |
| Functional requirement | Non-functional requirement |
| Train() | Train the neural net with the dataset |
| Upload() | Let the user upload the image containing strings of number. The supported image format is .png, .jpeg only. |
| Preprocess() | Apply thresholding, inversion and resize the image to feed to the neural network. |
| Identify() | Recognize all strings |

Table 1 describes the basic functionality of Recognizer which includes training the neural network. The user is provided with the interface from where user can upload the image containing the strings of image. The image is preprocessed and normalized to the similar image as image in dataset. Then, the image is identified.

The user can upload image and recognize it as many times as he/she wants.

## 2.3 Feasibility Analysis

### 2.3.1 Technical feasibility

This is a web application that uses Flask Framework. It uses Jinja2, HTML and CSS as front end and Python as the back end. It requires a server, client, and internet connection to function properly.

It supports both Windows and Linux platform for its operation. All of the technology required by this project are available and can be accessed freely, hence it was determined technically feasible.

### 2.3.2 Operational feasibility

This project has a simple design and is easy to use. It uses two-tier architecture (i.e. Client and Server). It can be easily accessed from anywhere having the internet connection and can be used in recognition of Nepali digits.

### 2.3.3 Schedule feasibility

Let us consider the activity as follows:

A as Collect Datasets, B as Literature Review, C as Design User Interface, D as Documentation, E as Segmentation , F as Train Neural Network, G as Recognize. The activity diagram is shown in figure 1.

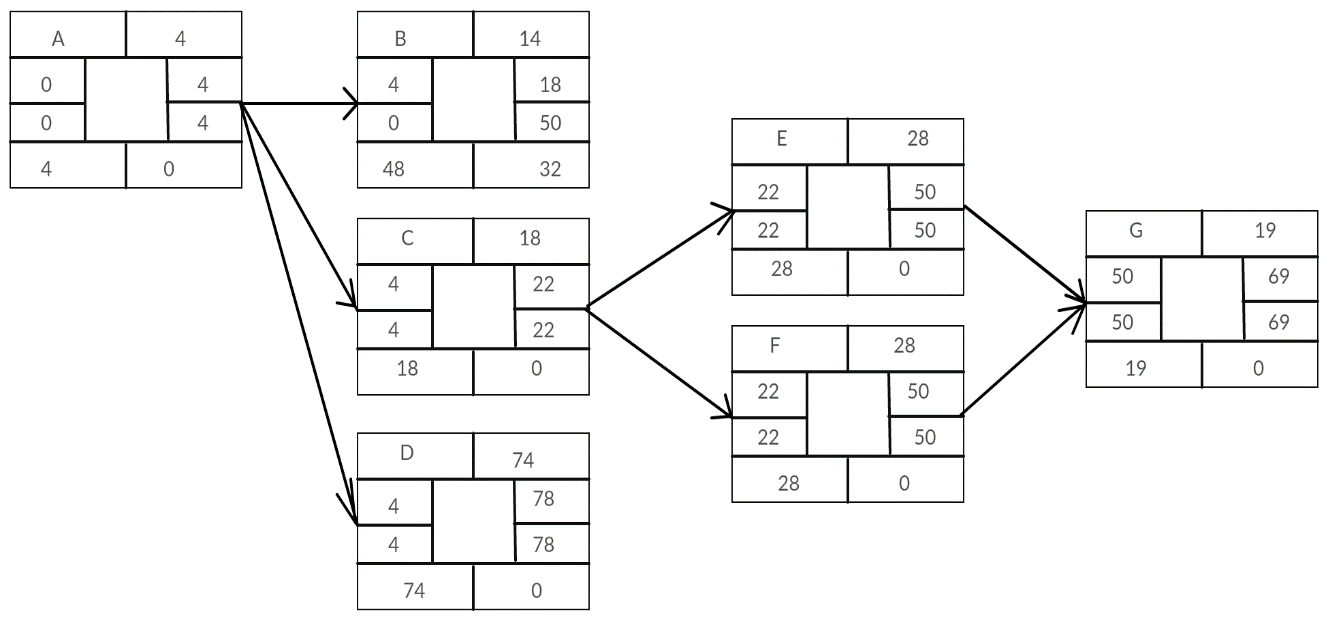


Figure 2.1 Activity network diagram of the project

Figure 1 shows Activity Network Diagram of developing this project. The early finish time for the project is 77 days and the late finish time for the project is 90 days. Three tasks “Literature Review”, “User Interface Design ” and “Documentation” had the slack time of 31, 7 and 12 days respectively. The above we can see that this project was completed in 13 weeks which is within 15 weeks of a semester. Hence, Recognizer was determined to be feasible in terms of schedule.

# 

# CHAPTER 3: SYSTEM DESIGN

## 3.1 Methodology

The recognizer is based on neural network classifier that feeds on datasets. The overview of methodology is as shown in the figure below:

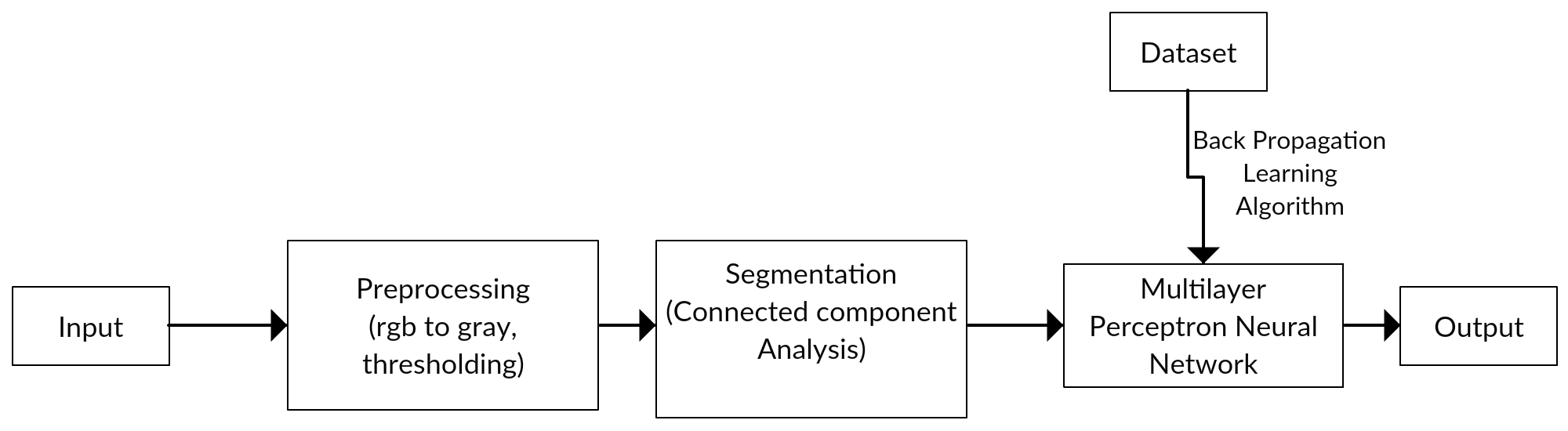


Figure 3.1 Methodology

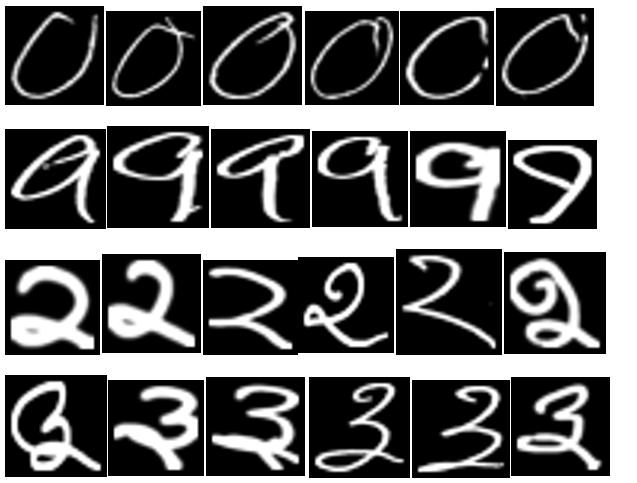
Figure 3.1 shows the methodology that is followed in recognition of handwritten Devnagari numeral strings.

### 3.1.1 Data collection

The data set were collected from Computer Vision Research Group, Nepal. The link for the dataset is:

[https://web.archive.org/web/20160105225948/http://cvresearchnepal.com:80/wordpress/](https://web.archive.org/web/20160105225948/http:/cvresearchnepal.com:80/wordpress/)

### 3.1.2 Data Information



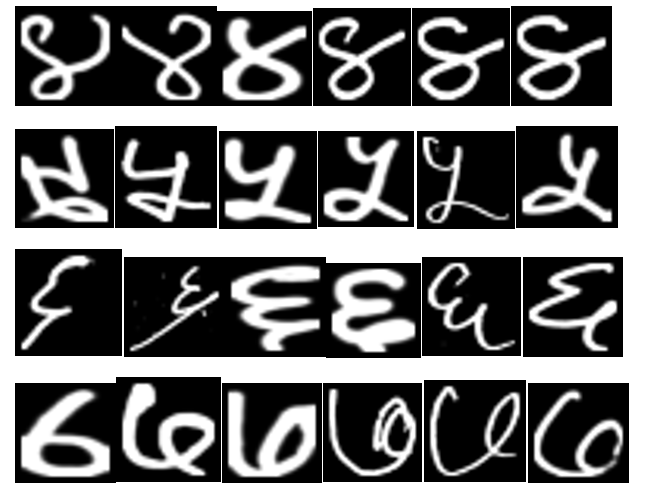




Figure 3.2 Sample of data set images

Figure 3.2 shows a sample of training data sets of digits 0-9 in Devnagari script. The data sets are prepared by Computer Vision Research Group. This consists of 2000 distinct images of a single digits. These are the digits written by students of grade 6 and 7 of Mount Everest Higher Secondary School.

This images are image of size 32\*32. From 2000 images, I selected 17000 images as the train set and 300 images as the test set.

### 3.1.3 Data preprocessing

The images are time consuming to feed in the neural network directly. Therefore the images are turned in csv file that contains 1025 values being the first value as label and 1024 values the gray-scale value of the pixel. The label is to denote the exact digits for training the neural network.

The preprocessing is also done while detecting and recognizing the image uploaded by the user. The steps in preprocessing is as follows:

* Adaptive thresholding

The image uploaded by the user may not have constant lightening throughout the picture. Therefore, the picture is passed through adaptive thresholding that changes the image to black background and white digits.

* Segmentation

The segmentation is done in two phase. The first segmentation is to locate the string of the numbers. The image processing can be a great help to recognize the string.

Basically, dilation in the image is done in x-axis, which thickens the image in x-axis. The images near to each other are joined and act as the single segment.

Connected Component Analysis is done to extract the strings that is to be segmented. Then, another segmentation phase is segmenting the digits in the string. In this phase, there is no need to apply dilation.

### 3.1.4 Training Neural Network

The neural network architecture used in this project for training purpose is as follows:

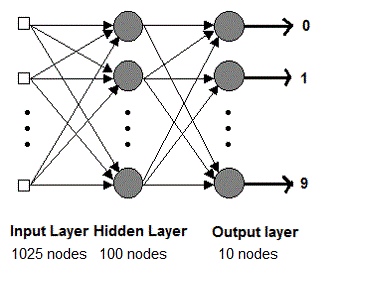


Figure 3.3 Sample of data set images

Figure 3.3 shows the architecture of neural network for recognizing handwritten digit. The architecture consists of multi-layer perceptron neural network that has three layers. They are input layer, hidden layer and output layer. The input layer has 1025 nodes. The image size is 32\* 32. Therefore it requires 1024 nodes for each pixel. The one extra node is the Label node that guides the neural network by mentioning the correct digit of the image. There are 100 nodes for the hidden layer. The number of nodes for hidden layer is difficult to determine as there are no fixed way of getting the numbers. The output layer has 10 nodes. Each node is the representation of digits from 0-9. Here, Neural Network is basically a classifier that classifies the image or csv file fed to it into ten classes.

## 3.2 Learning Algorithm

The training algorithm used to train the neural network is back-propagation learning algorithm. Neural Network is a supervised learning classifier that has input set, output set and target value. If the output value is not equal to target value, the error is propagated through each layers in backward direction. Therefore, this is called Back-propagation learning algorithm.

Every nodes fire when they reach the activation function. The activation function used here is the sigmoid activation function shown in equation (3.1).

g(x) = 1/ (1 + e-x ) ----- (3.1)

The back propagation algorithm is as follows:

1. Initialize the weights to small random values
2. Feed the training sample through the network and determine the final output
3. Compute the error for each output unit, for unit k which is given in equation (3.2).

δk = (tk – yk)f’(y\_ink) ----------- (3.2)

where

δk = error

tk = targeted output

yk =Actual Output

f’ =derivative of function, in this case sigmoid function

1. Calculate the weight correction term for each output unit, for unit k which is given in equation (3):

ΔWjk = α δk zj -------------------------(3.3)

where ΔWjk = difference in weight

α = learning rate

δk = error

zj = hidden layer signal

1. Propagate the delta terms (errors) back through the weights of the hidden units where the delta input for the jth hidden unit is given in equation (3.4):

δin\_j = ∑ δk Wjk for k = 1 to m -------------(3.4)

1. Calculate the weight correction term for the hidden units by equation (5):

ΔWij = α δk xi -----------------------------------(3.5)

1. Update the weights as the equation (6).

Wik(new) = Wik(old) + ΔWik -------------------(3.6)

1. Test for stopping (maximum cylces, small changes, etc)

## 3.3 System Design

### 3.3.1 Class diagram

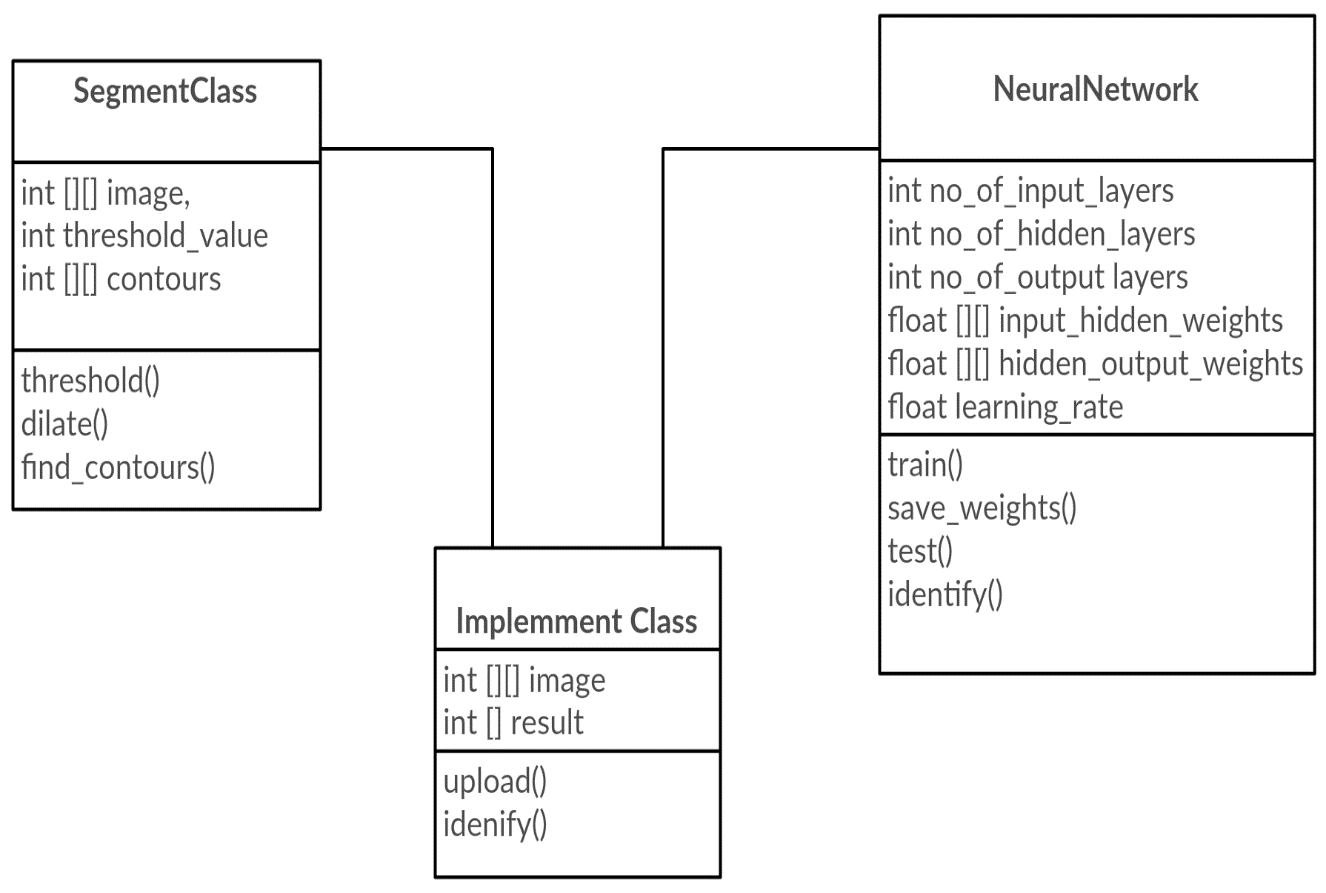


Figure 3.4 Class Diagram of Handwritten Digits Recognizer

Figure 3.4 explain the classes used in this project. There are three classes used in total, Segment class, Neural Network class and Implement class.

Implement class is the main class from which the function of segmentation class and Neural Network class is called. Implement class also is responsible for file upload. The main class calls segmentation class which finds the contours and segment the digits. Thereafter, the Neural Networks class is called which identifies the digits. The implement class also arranges the letters to make a string.

### 

### 3.3.2 State diagram

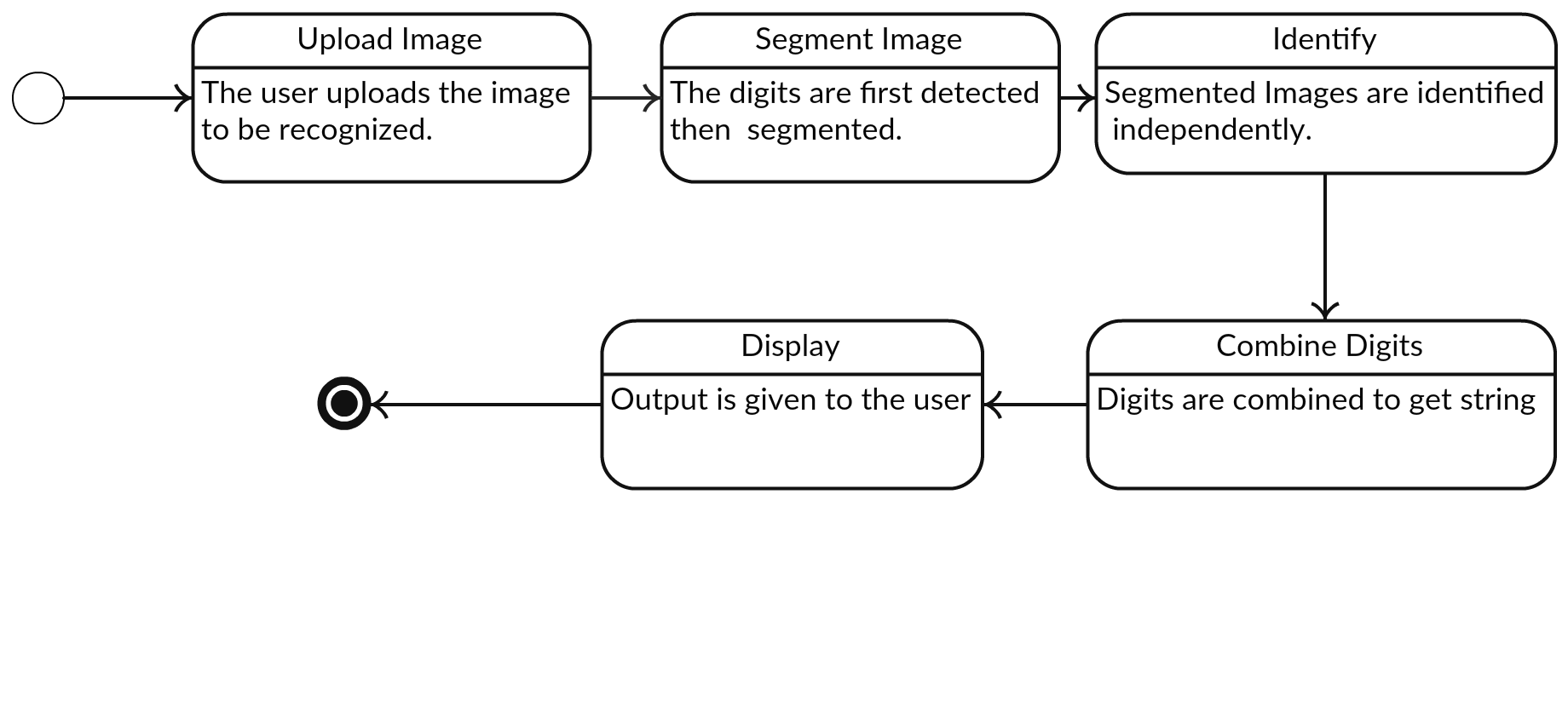


Figure 3.5 State Diagram of the project

Figure 3.5 explains the different state of the system is shown. First, the user opens the system. The system is in the state to receive image for recognition. When a user uploads image to be recognized, the system transits to the state where it segments the image. Then the segmented images are passed to neural net classifier.

After the segmented images are classified then the digits are combined to get the strings. Finally, the display is show to the user.

### 3.3.3 Sequence diagram

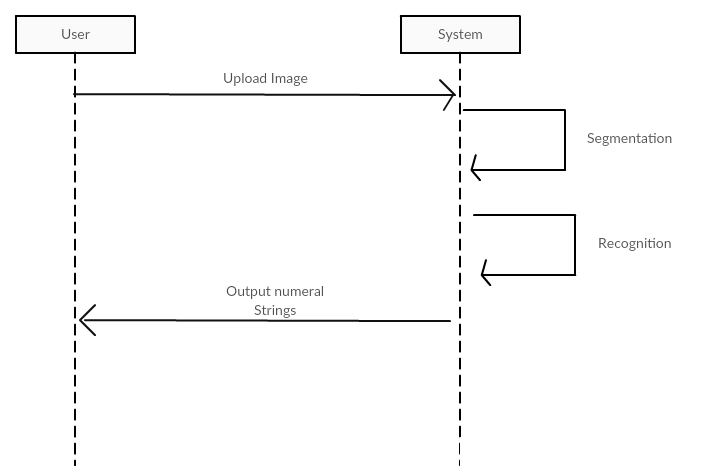


Figure 3.6 Sequence Diagram of the project

Figure 3.6 explains the sequence of the system. Initially, a user opens a browser and uploads the image of the handwritten digits. Then the system segments the images with in the system and recognizes the images with in the system. The final output is the numeral string that is displayed to the user.

# CHAPTER 4: IMPLEMENATION AND TESTING

## 4.1 Implementation

User can access the application through a browser and see the interface. The interface is simple and consists of only a form that has input type image and submit button. The user needs to select the input image and press submit. If no image is selected, it gives warning that the input is empty. Similarly, the input image should be of the format .png or .jpeg.

After the user inputs the image, the system locates the digit in the image. Then it groups the digits into strings by dilation. The digits that are nearer are separate and then each digits are segmented. Then the segmented images are then fed to the neural network.

The training was done in a computer having following specifications:

Intel® Core™ i7-5500U CPU @ 2.40 GHz

RAM: 8 GB

OS: 64 bit

It took 48.09 seconds for training the neural network that trains on 17,000 data.

### 4.1.1 Tools used

CASE tools:

1. Creately.com for drawing diagrams

Client side:

a) HTML

b) Twitter Bootstrap CSS

c) Jinja2

Server side:

1. Python

This section describes the technologies used in this project. This project is a web application that uses Flask framework of python language. It uses HTML and CSS for front end, Jinja 2 for scripting and Python as the backend language.

All the algorithms for the application are written in python classes and the groovy classes. OpenCv library is used for image processing whereas no library is used for neural network. The neural network is simple class coded in python.

## 4.2 Description of Major Classes

The major classes in the application are:

### 4.2.1 Segment

Segment is the class which methods are called from the main class. Segment class imports OpenCv framework for image handling. The segment class is called twice. First to segment strings and then the second is to segment the individual strings. The segment class calls function of openCv such as adaptive threshold, resize etc. to handle the images.

### 4.2.2 NeuralNetwork

Neural Network is the classifier class of this project. The NeuralNetwork class consists of input nodes, hidden nodes and output nodes. The NeuralNetwork class is randomly initialized with small random values as weight while training. However, during the recognition or testing phase, the Neural Network class is initialized with updated weights. This class contains function such as train, test, save weights.

### **4.2.3 Main**

This is the main class which assists in uploading images and calling the function of above classes to apply on that image. This also renders the view and design when the application loads.

## 4.3 Steps

To recognize the strings of digits written in the paper, the first step is to locate the digit. This is done with the help of image processing. The steps in forming the numeral strings after each digit is verified is as below:

### **4.3.1 Thresholding**

The image uploaded by the user may consists noise, multiple strings etc. The first step to convert the image uploaded to image similar to dataset is thresholding. Thresholding can be done with a fixed thresholding value or adaptive thresholding method. The difference in fixed thresholding values and adaptive thresholding value is shown in figure.

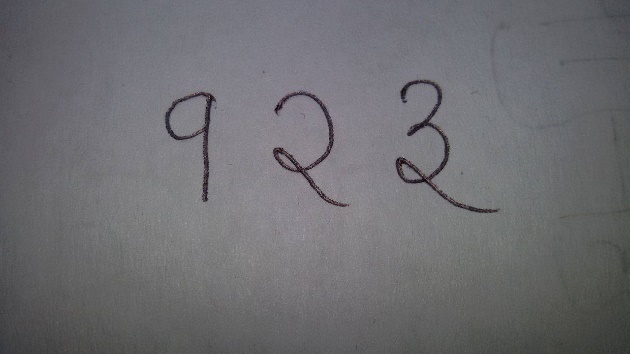


Figure 4.1 an image of handwritten numeral string

Figure 4.1 shows the image uploaded by user.

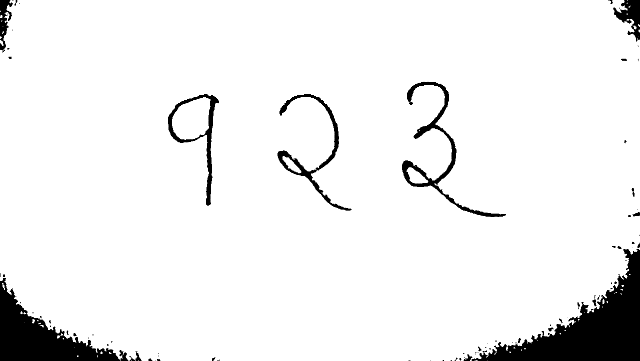
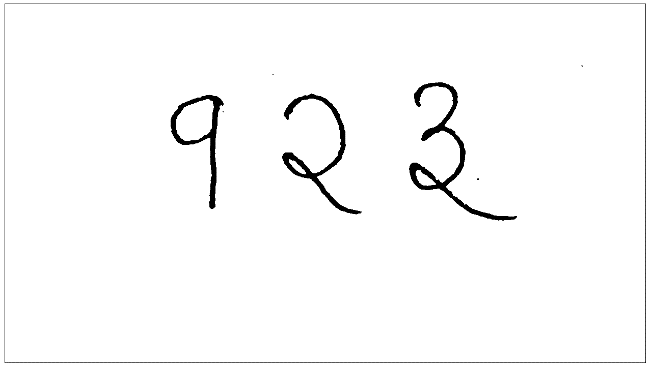


Figure 4.2 Thresholding with value = 80(left) Vs Adaptive Thresholding(right)

Figure 4.2 shows the different approach for thresholding. Adaptive thresholding threshold image using local thresholding values.

Then the negative of the image is taken. This gives image shown in Figure 4.3.

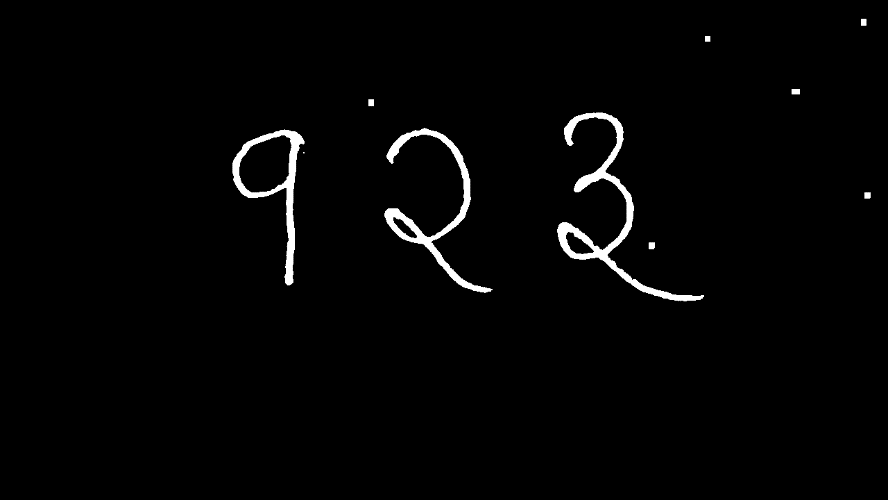


Figure 4.3 Negative image

Figure 4.3 shows the negative image. It still contains noise.

This image still contains small dots known as salt noise. This noise can be reduced by median filtering. The image after median filtering is shown in figure 4.4:

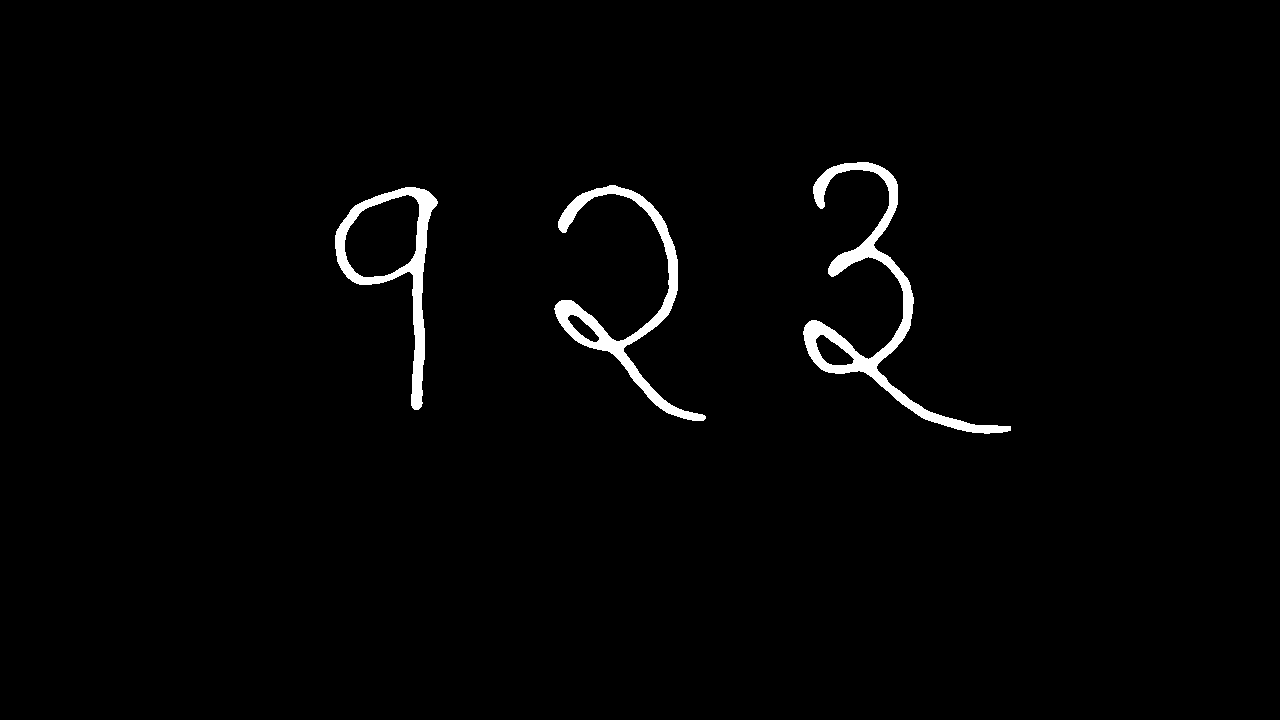


Figure 4.4 Median filter

Figure 4.4 shows the removal of noise using median filter.

### **4.3.2 Connected Component Analysis**

This is the way of segmenting the digits. For the image that contains multiple strings of digit, the program first dilates the image with 50 pixels. If the pixels are touched, then the digits are considered to be as a single string.

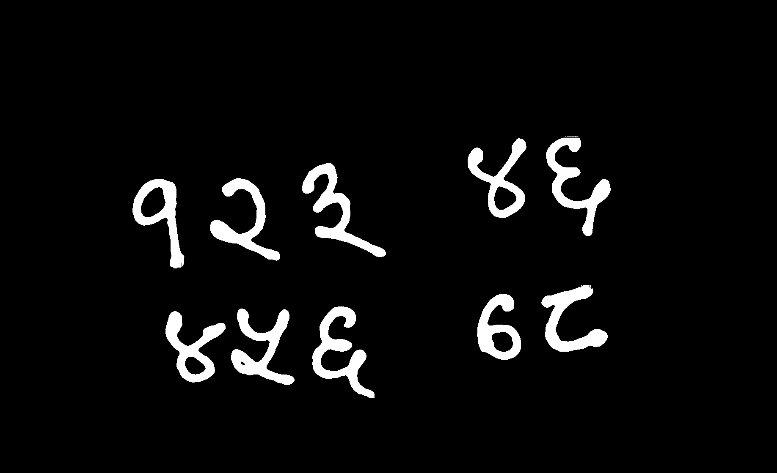


Figure 4.5 Median blurred image

Figure 4.5 shows the strings to be recognized.

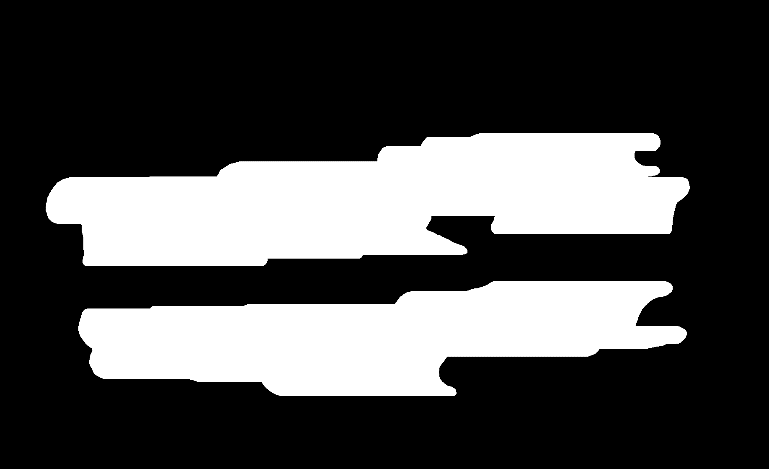


Figure 4.6 Dilation

Figure 4.6 shows dilation of the strings.

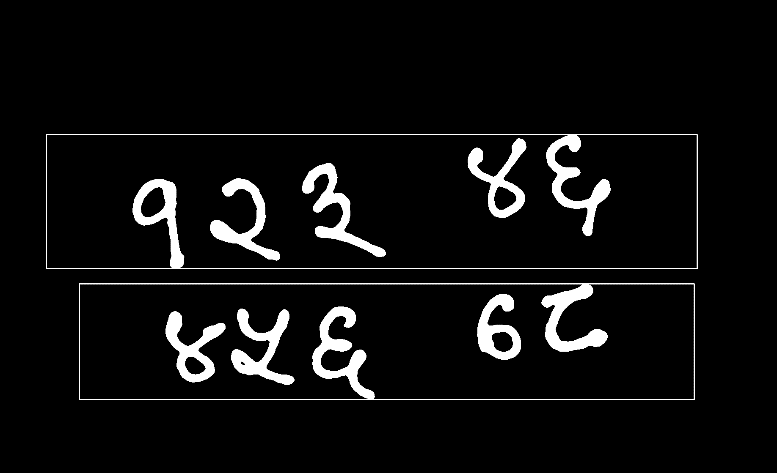


Figure 4.7 Connected Component Analysis

Figure 4.7 shows the segmentation of string using connected component analysis.

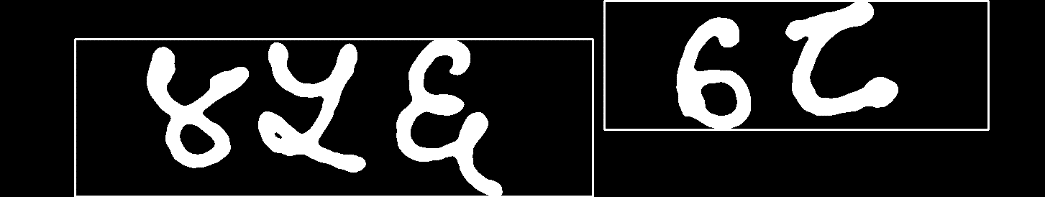


Figure 4.8 Segmentation of strings

Figure 4.8 shoes segmentation of strings inside the first segmented string.

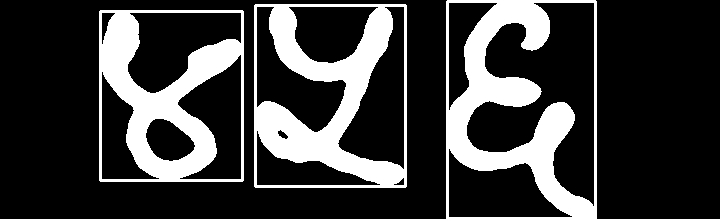
 

Figure 4.9 Segmentation of digits

Figure 4.9 shows the segmentation of digits.

E:\7thSem\FinalYearProject\docu photo\segmentation\resized.png E:\7thSem\FinalYearProject\docu photo\segmentation\f.pngE:\7thSem\FinalYearProject\docu photo\segmentation\3.png

Figure 4.10 Resize to 32\* 32 size image

Figure 4.10 shows the resized image of single digits. The 32\* 32 pixel image is the converted to csv format and send to nueral network for recognition.

## 4.4 Testing

The 300 data for each digits where kept as the test set. The above architecture give 96.3 % accuracy after the neural net is trained.

The efficiency was tested in terms of learning rate and number of nodes in hidden network. The variation is show in graph below:

Figure 4.11 Efficiency against learning rate

Figure 4.11 shows learning rate plotted against efficiency. The highest efficiency is at 0.3 learning rate which is 96.3%.

Figure 4.12 Efficiency against number of hidden layers

Figure 4.12 shows number of hidden nodes plotted against efficiency.

# CHAPTER 5: MAINTENANCE AND SUPPORT PLAN

## 5.1 Maintenance Plan

This project will implement corrective maintenance for resolving different bugs and errors that may occur when this project is made live. Perfective maintenance will be implemented for increasing efficiency of the project by optimizing various implementation methods. Preventive maintenance will be implemented to make sure that this project will not be harmed by hackers and security mechanism will be added.

## 5.2 Support Plan

This project will be presented to the respective authority of the Government for investment so that the Nepali language can be promoted. For self-sustenance of this project, the project can be hosted in any website.

# CHAPTER 6: CONCLUSION AND RECOMMENDATION

## 6.1 Conclusion

Recognition of handwritten Devnagari Numeral String using Multi-layer perceptron is carried out with 96.3 % accuracy. The recognition of the digits in string is dependent on the preprocessing of the image and normalization of image toward the train image sets.

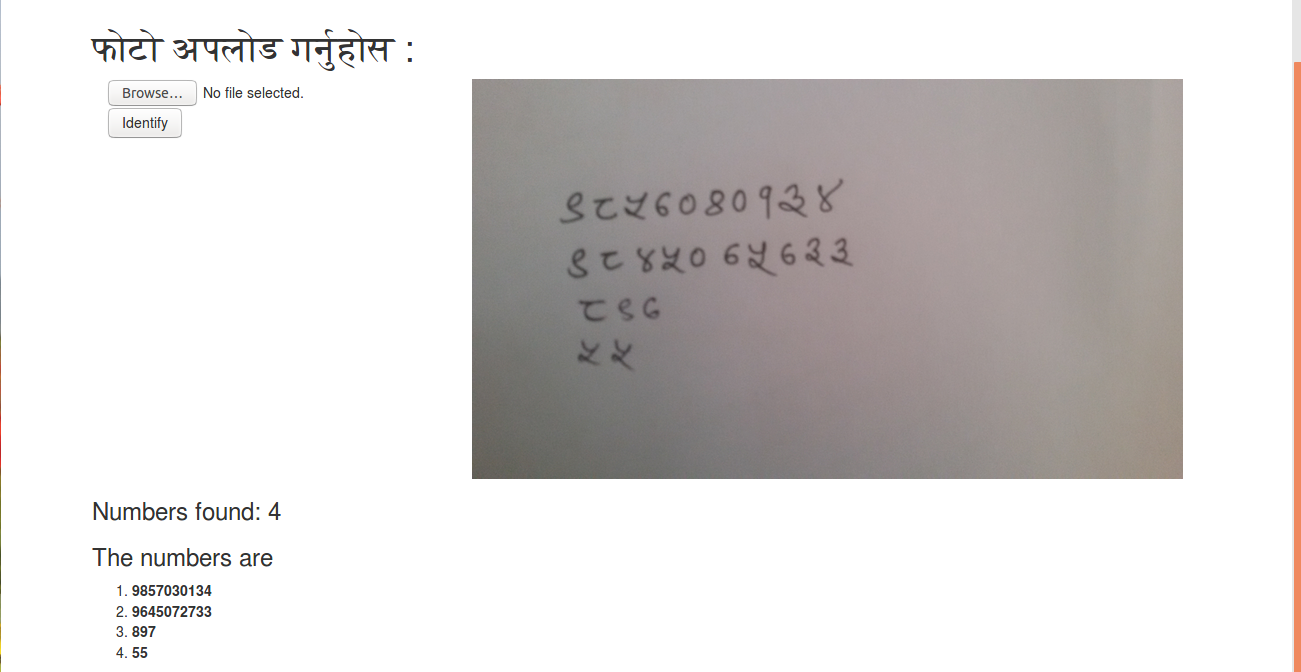
## 6.2 Recommendation

This project feeds on the raw pixel values that doesn’t extract any features. However, the features of the digits such as number of holes, inclination to axis etc. also could be taken in account to increase the feature set. This may give more accurate result.

# APPENDIX



APP. Figure 1 Landing Page of the project



APP. Figure 2 Landing Page of the project

# References

|  |  |
| --- | --- |
| [1] | A. K. P. K. G. Shailesh Acharya, "Deep learning based large scale handwritten Devanagari character recognition". |
| [2] | V. P. Agnihotri, "Offline Handwritten Devanagari Script," 2012. |
| [3] | B. C. I. K. Sethi, "Machine Recognition of Hand-printed Devnagri Numerals," 2015. |
| [4] | H.-Y. K. J.-W. S. J. H. K. Chan E. Cheong, "Handwritten Numeral String Recognition with Stroke Grouping". |
| [5] | G. H. k. P. N. T. Vasudev, "Transformation of Arc-Form-Text to Linear Form Text Suitable". |
| [6] | G. B. O. L. Leon Bottou, "Comparison of learning algorithms for handwritten digit recogniton". |