

**TRIBHUWAN UNIVERSITY**

**Institute of Science and Technology**

**“Nepali Handwritten Character Recognition using CNN”**

A Project Report

**Submitted to:**

**Department of Computer Science and Information Technology**

**Asian School of Management and Technology**

**Kathmandu, Nepal**

***In the partial fulfillment of the requirement for the bachelor’s Degree in computer***

***Science and information technology***

**Submitted By:**

Amir Maharjan (T.U. Roll No. 7922/072)

Anisha Maharjan (T.U. Roll No. 7924/072)

Manish Tuladhar (T.U. Roll No.7943/072)

Prajwal Shrestha (T.U. Roll No. 7948/072)

**ACKNOWLEDGEMENT**

We owe our most profound appreciation to the Asian School of Management and Technology for furnishing us with a chance to work on project as part of our syllabus. We are generously obligated to our supervisor Mr. Surya Bam for his immense support and the guidance through the entire project. It was his significant recommendation and suggestion that helped us to adapt up to the rising impediments amid the improvement of this project.

We are additionally thankful to Mr. Chakra Rawal for providing the flexible working environment and support as much as possible from his side. We are likewise thankful to every one of our respected educators for their motivation and encouragement that made ready towards the completion of this project.

We would like to extend our sincere regards to our respected principal Er. Anil Lal Amatya and all the faculty members who have helped us directly or indirectly for completing this project. We would also like to thank all of our friends, senior and colleague for their valuable suggestion, comments and support.

Thanking You,

Amir Maharjan (T.U. Roll No. 7922/072)

Anisha Maharjan (T.U. Roll No. 7924/072)

Manish Tuladhar (T.U. Roll No.7943/072)

Prajwal Shrestha (T.U. Roll No. 7948/072)

**ABSTRACT**

The constant development of computer tools leads to a requirement of easier interfaces between the man and the computer one of the example is handwritten character recognition. Handwritten character recognition is the ability of computer to receive and interpret the handwritten input images from sources such as document, photograph or touch-screen and transform it to machine readable and editable format. In this project the system implemented using the python programming language. There are several steps involved in this system. The first step is preprocessing. The input images (datasets) which consists of the handwritten character will be loaded and will go through some process like conversion to csv format, reshaping them into 4D array (batch, column, row, channel) and normalization. Next, the datasets will go through operations like convolution, max-pooling and dropout in order to extract the features. Then they are flattened to 1D array and passed through the dense layers to perform the classification being based on the extracted features. A simple and interactive HTML page is created in order to allow user to input their character through html canvas and see the result.

**LIST OF FIGURES**

2.1 Literature Review

2.2 Use Case Diagram for Handwritten Character Recognition

2.3 Sequence Diagram

3.1 Max Pooling Layer

3.2 Flattening Layer

3.3 Layers used in the CNN

**ABBREVIATIONS**

**2D** Two-Dimensional

**3D** Three-Dimensional

**4D** Four-Dimensional

**CNN** Convolutional Neural Network

**CSV** Comma Separated Values

**MLP** Multilayer Perceptron

**PCA** Principal Component Analysis

**ReLU** Rectified Linear Units

**SVM** Support vector machine

# CHAPTER ONE: INTRODUCTION

## Introduction

In modern society, we rely heavily on computers to process huge volumes of data. For economic reasons or business requirements, there is a great demand to quickly input the mountains of printed and handwritten information into the computer [1]. Handwritten character recognition is a field of research in artificial intelligence, computer vision, and pattern recognition. A computer performing handwriting recognition is said to be able to acquire and detect characters in paper documents, pictures, touch-screen devices and other sources and convert them into machine-encoded form. Its application is found in optical character recognition, transcription of handwritten documents into digital documents and more advanced intelligent character recognition systems [2].

Handwritten Recognition can be separated into two main types which are offline handwritten recognition and online handwritten recognition [3]. They use different input signals. For offline handwritten recognition, the input signal is generally from document of characters by using scanner. The input signal for online handwritten recognition is a signal while writer is writing. It converts the tip movements (strokes) of digital pen to a list of coordinates [4].

In twenty first Century handwritten digit communication has its own standard and most of the times in daily life are being used as means of conversation and recording the information to be shared with individuals. One of the challenges in handwritten characters recognition wholly lies in the variation and distortion of handwritten character set because distinct community may use diverse style of handwriting, and control to draw the similar pattern of the characters of their recognized script [5]. In next days, character recognition system might serve as a cornerstone to initiate paperless surroundings by digitizing and processing existing paper documents [4].

## Problem Definition

Nowadays, there is a great demand to quickly input the heaps of printed and handwritten information into the computer [1] and in this scenario there have been several problems for recognizing the data. When the data are misunderstood like during number recognition in cheque then amount withdrawn is affected. Likewise, it affects the communication too. Thus, it is the need of time to create a good system which can recognize the characters well.

## OBJECTIVES

The objective of character recognition is to recognize the input handwritten character and display the respective character.

## Scopes and Limitations

The scopes of handwritten character recognition are as follows:

* This work can be further extended to the character recognition for other languages.
* It can be used for automatic reading.
* It can be extended as a word recognition which in terms can be helpful for physically handicapped people for the processing of documents.

Every system has demerits along with the merits. So, like every other system handwritten character recognition system also has so limitations. The limitations are as follows:

* The handwritten characters are not always of same size and shape. It differs from person to person. So, the general problem arises during the classification of the characters because there might be some similarity between different characters written by different person.
* It gives the correct output only when the dataset provided is correct and sufficient.
* It takes a little time to generate the result.

## Report Organization

* Chapter 1 explains the introduction, problem statement and proposed system, and scopes and limitations.
* Chapter 2 reviews related works, requirement analysis and feasibility study of our project including use-case diagram and sequence diagram
* Chapter 3 specifies the system design of the project and describes the process model.
* Chapter 4 explains the system implementation and testing
* Chapter 5 discusses the conclusion

# CHAPTER TWO: REQUIREMENT ANALYSIS AND FEASIBILITY STUDY

## Literature Review

Due to the fact that each writer will have different style of writing and all scripts have their own character set and complexities to write text. Many researchers have worked in the area of handwriting recognition, and numerous techniques and models have been developed to recognize handwritten text both online and offline.

For recognition of isolated handwritten Devanagari numerals Principal component analysis (PCA) along with edge direction histogram and spines are used.[6][7]. A comparative study of Devanagari handwritten character recognition using twelve different classifiers and four sets of features was presented by Pal [7].

B.V. Dhandra et.al. [8] have proposed neural network approach along with four feature selection methods i.e. maximum profile distances, fill hole density, water reservoir and directional density features for recognizing Devanagari handwritten numerals and have achieved 98.4% recognition rate.

Further in [9] authors have implemented density and background directional distribution features for various zones where every zone is of size 8 x 8 pixels and each zone contains 9 features consisting of one density feature and 8 background directional distribution features. For classification Support vector machine (SVM) is used with RBF kernel.

To recognize non compound devanagari handwritten characters Sandhya arora et.al.[10] have proposed approach in which two approaches are combined i.e. Multilayer Perceptron (MLP) and Minimum edit distance. Shadow features and chain code histogram features are used with MLP.

Prachi Mukherji and Priti Rege [11] used shape features and fuzzy logic to recognize offline Devnagari character recognition. They segmented the thinned character into strokes using structural features like endpoint, cross-point, junction points, and thinning. They classified the segmented shapes or strokes as left curve, right curve, horizontal stroke, vertical stroke, slanted lines etc. They used tree and fuzzy classifiers and obtained average 86.4% accuracy.

Neha Sahu, Nitin Kali Raman, [12] describes the development and implementation system consisting combination of various stages. Artificial Neural Network technique is used to designed to preprocess, segment and recognize Devanagari characters. The system was designed, implemented, trained and found to exhibit an accuracy of 75.6 % on noisy characters.

Sonika Dogra, Chandra Prakash, [13] describes a recognition system proposed system Support Vector Machine is used as classifier and Diagonal feature extraction approach is used to extract features. From the results combination of SVM classifier and diagonal feature extraction approach is better method for the recognition of handwritten characters.

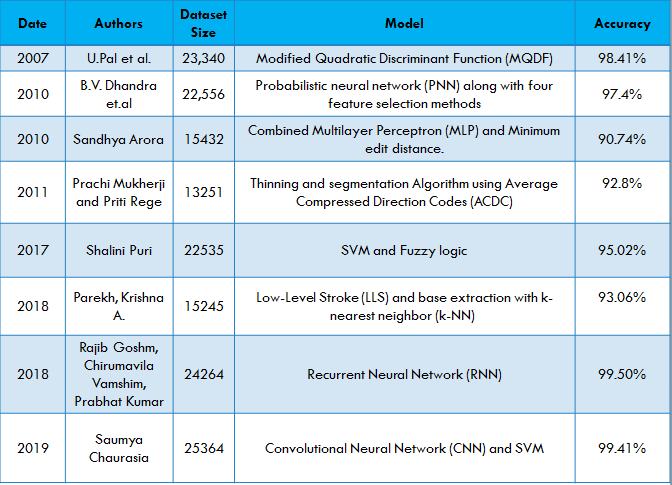


Fig 2.1 Literature Review

## Requirement analysis

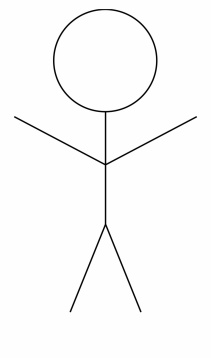
Requirement analysis results in the specification of software’s operational characteristics indicates software’s interface with other system elements and establish constraints that soft- ware must meets. The requirements analysis task is a process of discovery, refinement, modeling and specification. The scope, initially established by us and refined during project planning, is refined in details. Model of the required data, information and control flow and operations behavior are created.

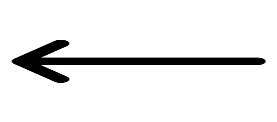
### Functional requirement

The functional requirements for a system describe what system does.

* The developed system should recognize handwritten Nepali character present in the image.
* System shall show the error message to the user when given input is not in the required format.
* System must provide the quality of service to user.
* System must provide accuracy for character recognition.

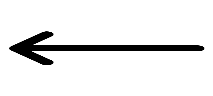
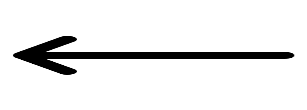
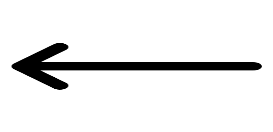
#### Use case diagram





**Draw Image**

**Clear/Reset Image**

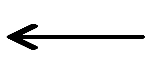


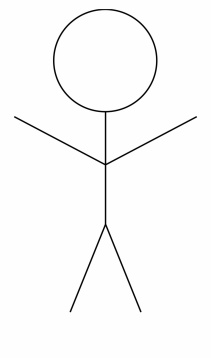
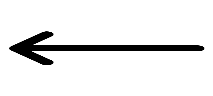
**Extract Feature**

**Read Input**

**Match Feature**

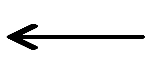
**Generate Output**





**User**

**Recognize Character**



**System**

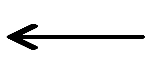


Fig: 2.2 Use Case Diagram for Handwritten Character Recognition

#### Sequence Diagram

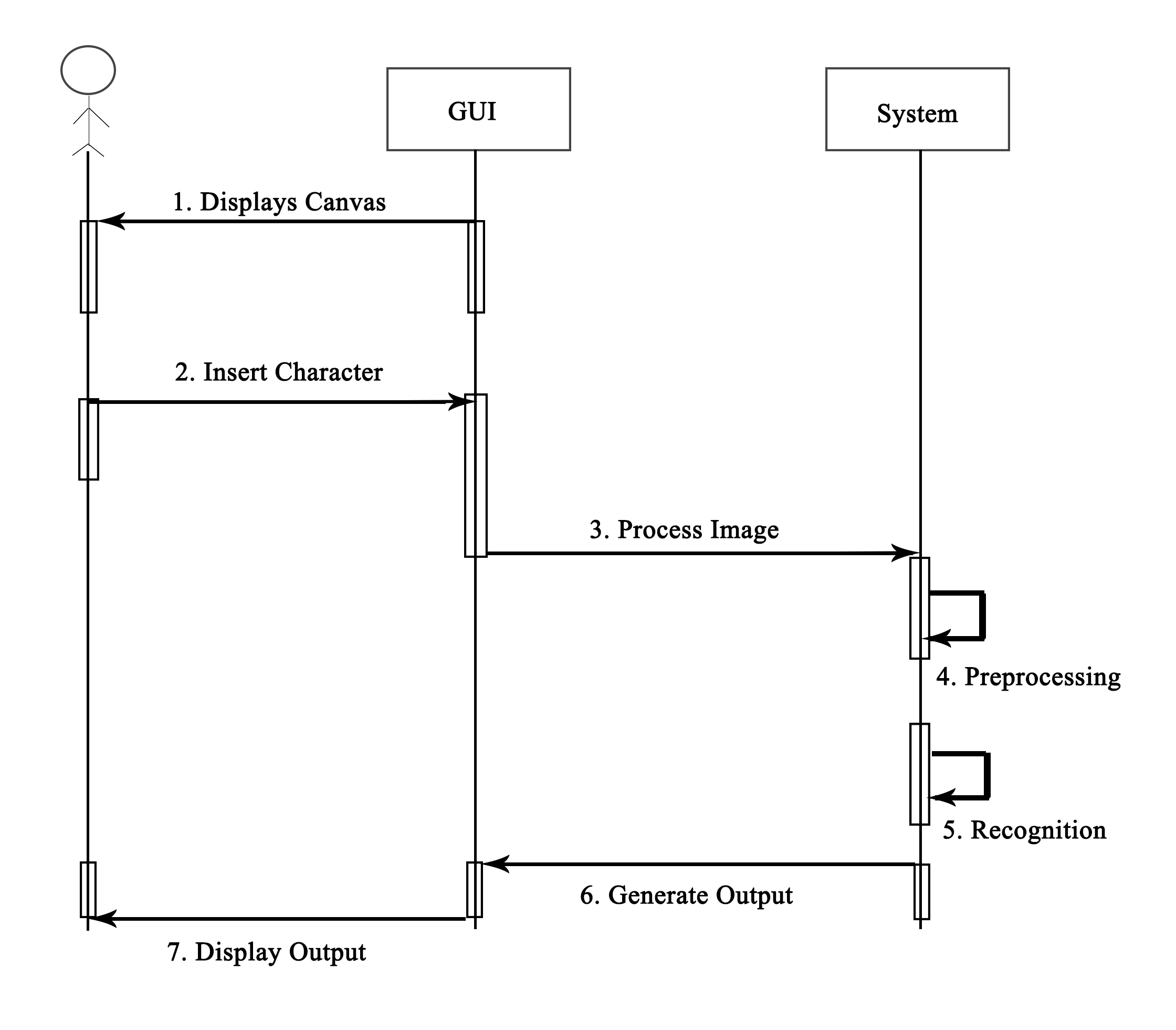


Fig 2.3 Sequence Diagram

### Non-Functional requirement

As the name suggests these are the requirements that are not directly interacted with specific functions delivered by the system.

* **Performance:** Handwrittencharacters in the input image will be recognized.
* **Functionality:** Thissoftware will deliver on the functional requirements.
* **Availability:** Thissystem will retrieve the handwritten text regions only if the image contains written text in it.
* **Flexibility:** It provides the users to load the image easily.
* **Learnability:** The software is very easy to use and reduces the learning work.
* **Reliability:** This software will work reliably for low resolution images and not for graphical images.

## Feasibility Study

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it’s worth doing.

### Technical Feasibility

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis. Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization?

### Operational Feasibility

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management from users?
* Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
* Have the user been involved in the planning and development of the project?
* Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project.
* Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

### Economic Feasibility

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale. Possible questions raised in economic analysis are:

* Is the system cost effective?
* Do benefits outweigh costs?
* Estimated cost of hardware
* Estimated cost of software/software development

### Schedule Feasibility

It is the measure of how reasonable the project time table is or the deadline is reasonable or not. During the lack of time or the time become mandatory, we must finish the project within a given time period. It mainly addresses:

Can the project really be completed in given period of time?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SN | Task Name | Duration | 2019 | | | | |
| March | April | May | June | July |
| 1. | Study and Analysis | 7 weeks |  | | | | |
| 2. | Data Collection | 2 weeks |  | | | | |
| 3. | Implementation | 5 weeks |  | | | | |
| 4. | Testing and Analysis | 3 weeks |  | | | | |
| 5. | Documentation | 5 weeks |  | | | | |
| 6. | Review | 1 weeks |  | | | | |
| 7. | Presentation and Submission | 1 day |  | | | | |

Fig: 2.3 Project Schedule

## Hardware and software requirement

### Hardware Requirement

* Intel i5 Processor
* 8 GB RAM
* 10 GB Hard Disk Space

### Software Requirement

* Python v3
* Django Framework
* Numpy (to do basic array manipulations)
* Pandas ( to do basic file related operations)
* Pickle ( to save our model )
* imread (to read dataset image)

# CHAPTER THREE: SYSTEM DESIGN

## System Architecture

### Pre -Processing

We will be converting our entire dataset to a csv (comma-separated values) file so that it would be easier to feed to our model. The csv would contain the image data along with the image label, which we will be separating.

The image data in our csv file would be in 3D arrays. The input shape that our CNN expects is a 4D array (batch, height, width, channels). Channels signify whether the image is grayscale or colored (In our case, we are using grayscale images so we give 1 for channels). So, we will be reshaping our image data accordingly. It’s always good to normalize data. Our image data will have data in each pixel in between 0–255. We will be scaling it to 0–1.

Character Recognition is a multi-class classification problem. All values(output) are equal to us so, we’ll be using one-hot encoding for image labels. One-hot encoding transforms integer to a binary matrix where the array contains only one ‘1’ and the rest elements are ‘0’.

### Recognition Model

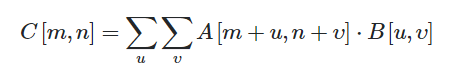
We’ll be using Convolutional Neural Network to develop a handwritten Nepali character recognition system in Python language. Preprocessed datasets will be fed to Convolutional Neural Network which will include feature extraction and classification.

We will be designing our model by using different layers. They are as follows:

#### Feature Extraction

* The first hidden layer will be a convolutional layer called Convolution2D. The layer has 32 filters/output channels, with the kernel size of 5×5 and an activation function. This layer is able to detect the pattern (edges, shapes, textures, objects, etc.) in the images.

Here a filter with a kernel size 5x5 is just a 5x5 matrix initialized with random numbers. After receiving an input channel, this filter will slide over every 5x5 pixels of the input and compute the sum of element wise multiplication of itself and the 5x5 input block and store it, until it’s slid over every 5x5 block of pixels from the entire image.



A = 5x5 input block

B = 5x5 filter

C = result matrix

* The Second layer will be the Max Pooling layer with pool size 2x2. Max pooling reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer.

It takes the first 2x2 region of the input (output of above convolutional layer) and calculate the max value from each value in the 2x2 block. This value is stored in the output channel, which makes up the full output from this max pooling operation. Then it just slides over by 2 and again calculate the max value in the next 2x2 block and store it in output and the process continues. Once it reaches the edge, it moves down by 2 and repeats the process. This process is carried out for the entire image, and when it’s finished, it outputs the new representation of the image in the output channel

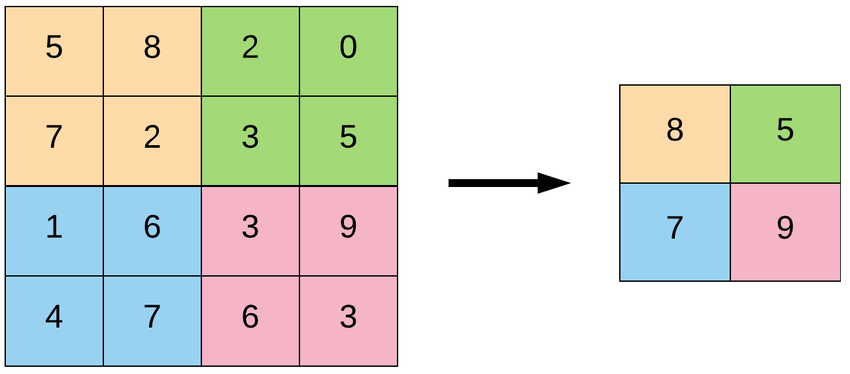
Example:

Fig 3.1 Max Pooling Layer

* The third layer will be another convolutional layer with 32 filters/output channels with the kernel size of 3×3 and an activation function. (just like first layer, but kernel size of 3x3 instead of 5x5)
* The fourth layer will be another Max Pooling layer, exactly like second layer.
* The next layer will be a regularization layer using dropout called Dropout. It is configured to randomly exclude 20% of neurons in the layer in order to reduce overfitting. Overfitting occurs when our model becomes really good at being able to classify or predict on data that was included in the training set, but is not as good at classifying data that it wasn’t trained on. Dropping out 20% neurons randomly helps reducing the overfitting.

#### Classification

* Next layer will convert the 2D matrix data to a vector called Flatten. It allows the output to be processed by standard fully connected layers which will be the next layer in the model. Outputs that are passed to fully connected layers must be flattened out before the fully connected layer will accept the input.

Flattening the 2D matrix input from the above layer simply means converting that 2D input data matrix to 1D. For this, it takes every row of the 2D input matrix in order, and put them serially in a 1D array.

So, if the input matrix is 4x4, then the flattened array will be a 1D array of length 16.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

Fig 3.2 Flattening Layer

* Next layer will be a fully connected layer (dense layer) with 128 neurons (“fully connected” means that every node in the first layer is connected to every node in the second layer). It performs classification based on the features extracted by the previous layers.

Each connection between two nodes has an associated weight, which is just a number. Each weight represents the strength of the connection between the two nodes. When the network receives an input at a given node in the input layer, this input is passed to the next node via a connection, and the input will be multiplied by the weight assigned to that connection.

For each node in the second layer, a weighted sum is then computed with each of the incoming connections. This sum is then passed to an activation function, which performs some type of transformation on the given sum.

node output = activation (weighted sum of inputs)

* Next (last) layer will be output layer with 46 neurons (46 because there are 10 digits, 36 alphabets 'ka' to 'gya') and it uses softmax activation function. Each neuron will give the probability of that class. It’s a multi-class classification that’s why softmax activation function is used instead of usual sigmoid activation function.

Fig 3.3 Layers used in the CNN

Output CNN Model

Convolutional2D Layer

5 x 5 Kernels

32 Filters (output channel)

MaxPooling Layer

2 x 2 Pool Size

Convolutional2D Layer

3 x 3 Kernels

32 Filters (output channel)

Dropout Layer

Flatten Layer

Flatten to continuous linear vector

Dense Layer (128 units)

Dense Layer

46 units (no of outputs)

Preprocessed datasets

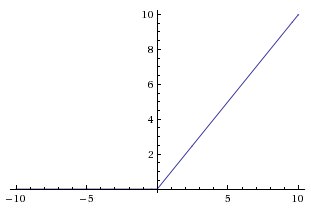
MaxPooling Layer

2 x 2 Pool Size

### Activation Function

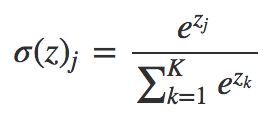
We will be using **Rectified Linear Units (ReLU)** activation function in our layers except in last output layer. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. It is the simplest non-linear activation function you can use, obviously. When you get the input is positive, the derivative is just 1, so there isn't the squeezing effect It can be written as

f( x ) = max( 0, x )



In the last output layer, we used **softmax** activation function. The softmax function squashes the outputs of each unit to be between 0 and 1, just like a sigmoid function. But it also divides each output such that the total sum of the outputs is equal to 1.

The output of the softmax function is equivalent to a categorical probability distribution, it tells you the probability that any of the classes are true. Mathematically the softmax function is shown below, where z is a vector of the inputs to the output layer (if you have 10 output units, then there are 10 elements in z). And again, j indexes the output units, so j = 1, 2, ..., K.



## Process Design

Preprocessing

Training Dataset

Feature Extraction

Classification

Trained Model

Preprocessing

User Input

Feature Extraction

Feed to Model

Make Prediction

Result

# CHAPTER FOUR: IMPLEMENTATION AND TESTING

## Implementation

User is allowed to draw a character in the HTML canvas. The canvas data is taken and converted into an image file of size 32 by 32. That image is loaded and converted into an array of (1, 32, 32 ,1). That array signifies a single image with width and height of 32, with 1 channel as it is greyscale. Then the model we created from the above processes is loaded and is given the array input. The model then gives us character it predicted. After that, the image that was saved from the canvas earlier is compared with one of the images of the predicted character (loaded form the training dataset) in order to compute the similarity. The result is then displayed if the comparison gives a satisfactory similarity value. Else a message is displayed saying it failed to predict.

### Tools Used

We have used following tools in order to create our project.

* Python

Python is a modern high-level programming language. It is consistently ranked as one of the easiest programming languages to learn, and is known for its high reliability and simple syntax. Python is the most popular programming language in the world and is also known as the most suitable language for data science tools and applications.

* VS Code

Visual Studio Code or VSC for short is one of the most popular source code editors used by the programmers. It’s fast, it’s lightweight and it’s powerful too. It has everything which any programmer expects from any code editor with some additional and useful features. It’s lightweight, fast, open source and cross-platform nature along with other cool features gives it an extra edge over any other editor.

* Python API
  + NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object, sophisticated functions, useful linear algebra, Fourier transform, and random number capabilities

* + Pandas

The pandas module is invaluable for manipulating all manner of data sets. Most of the module’s functionality revolves around a “DataFrame” which is a Python version of a spreadsheet, stored in memory. A pandas DataFrame provides a plethora of handy utilities which make manipulating the data more intuitive.

* + Pickle

Pickle is used for serializing and de-serializing a Python object structure. Any object in python can be pickled so that it can be saved on disk. It “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream.

## Testing

### Unit Testing

For unit testing, each of the system’s components were tested, beginning with preprocessing part. A single image was taken from the training dataset and was feed to preprocessing layer. It successfully processed the image into desired format i.e into 4D array (batch, height, width, channels). Then that preprocessed array was sent to Convolutional function and the result image was plotted. It showed that the feature of data was successfully extracted. Then the result was sent to MaxPooling function and the

result was plotted again. It successfully decreased the size of the data and preserved all the extracted features while doing it. Then likewise flatten layer was also tested. It successfully serialized the data.

The HTML Canvas that is used to take input from the user was also tested. A character was drawn in the canvas. After drawing it successfully generated the image of that user drawn character which is then later feed into the model.

### System Testing

After testing all the individual components of our system, the whole system was also tested to see if it can successfully predict the handwritten character or not. At first a very clear form of character was drawn in the HTML canvas and prediction was done. The result was very accurate. It successfully predicted the character almost every time. After that a little distorted character was drawn. The model was still able to predict them but with low probability. But there were also some cases where the model was completely wrong. Then a completely invalid character was drawn. Model was able to detect that the character was invalid.

So, there were some errors where the model failed to successfully predict. But overall model can successfully predict the Nepali handwritten character.

# CHAPTER FIVE: CONCLUSION

## CONCLUSION

An implementation of Handwritten Digit Recognition using Convolutional Neural Network has been implemented in this project. At first, we reviewed the approaches that are nowadays used in similar applications. After that, we delved into the inner workings of CNN. With the knowledge we had described, we specified the requirements of the project and planned the solution. We mainly focused on the users input character, processing it and returning the accurate result as possible. Finally, our project met the expected outcome; it is able to recognize 36 alphabets of Nepali Language. The results have proven that this approach have fetched good performance when compared to the traditional methods.

# **References**

[1] Ivor Uhliarik,"Handwritten Character Recognition Using Machine Learning Methods"

[2] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu and Ching Y. Suen, "Character Recognition Systems"

[3] Kwanlekha Sae-Tang and Jumpol Polvichai, "An Effective Handwritten Digit Recognition with Convolutional Neural Network"

[4] Wikipedia contributors, "Handwriting recognition," *Wikipedia, The Free Encyclopedia,* <https://en.wikipedia.org/w/index.php?title=Handwriting_recognition&oldid=866980498> (accessed April 3, 2019)

[5] S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana and Abdullah Al Jobair, "Handwritten Digit Recognition using Machine Learning Algorithms"

[6] Baheti M.J., Kale K.V., “Gujarati Numeral Recognition Affine Invariants Moment Approach”, 14th International Conference on Recent Trends in Engineering and Technology, May-2012, IJCECSCSE

[7] Pal, U., Wakabayashi, T., Kimura, F. 2009. “Comparative Study of Devanagari Handwritten Character Recognition Using Different Feature and Classifiers” 10th International Conference on Document Analysis and Recognition, ICDAR '09 : 11114

[8] B. V Dhandra, “Kannada, Telugu and Devanagari Handwritten Numeral Recognition with Probabilistic Neural Network? : A Novel Approach,” Architecture, pp. 83–88, 2010

[9] M. Jangid, R. Dhir, R. Rani, and K. Singh, “SVM Classifier for Recognition of Handwritten Devanagari Numeral,” Processing, no. Iciip, 2011.

[10] S. Arora, “Recognition of Non-Compound Handwritten Devnagari Characters using a Combination of MLP and Minimum Edit Distance,” J. Comput. Sci., no. 4, pp. 107–120.

[11] Prachi Mukherji, Priti Rege, “Shape Feature and Fuzzy Logic Based Offline Devanagari Handwritten Optical Character Recognition”, Journal of Pattern Recognition Research 4,pp. 52-68, 2009

[12] Neha Sahu, Nitin Kali Raman “An Efficient Handwritten Devnagari Character Recognition System Using Neural Network” IEEE, 2013

[13] Sonika Dogra, Chandra Prakash, “PEHCHAAN: Devnagari Handwritten Character Recognition System Based On SVM”, IJCSE, ISSN : 0975-3397 Vol. 4 No. 05 May 2012

Vyas, M. Verma, K.A "Comprehensive survey of handwritten character segmentation" IEEE, 2014