NATIONAL COLLEGE OF ENGINEERING

(Affiliated to Tribhuwan University)

Talchhikhel, Lalitpur

[Subject Code: CT 654]

A MINOR PROJECT REPORT ON

"NEPALI CHARACTER RECOGNITION BY USING CNN"

Submitted by:

Saday Malla	**074/79130**
Saurav Shakya	**074/79135**
Vishma Shrestha	**074/79140**
Yalab Guragain	**074/79146**

A MINOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

March, 2021

NEPALI CHARACTER RECOGNITION BY USING CNN

Submitted by:

Saday Malla [074-79130]

Saurav Shakya [074-79135]

Vishma Shrestha [074-79140]

Yalab Guragain [074-79146]

Supervised by:

Er. Pradip Adhikari

(Department of Computer and Electronic Engineering)

A MINOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTFOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to

Department of Computer and Electronics Engineering

National College of Engineering

Talchhikhel, Lalitpur.

March, 2021

COPYRIGHT

NATIONAL COLLEGE OF ENGINEERING

NEPALI CHARACTER RECOGNITION BY USING CNN

APPROVAL LETTER

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a Project repent entitled "**NEPALI CHARACTER RECOGNITION BY USING CNN**" submitted by Saday Malla, Saurav Shakya, Vishma Shrestha and Yalab Guragain in partial fulfillment for the degree of Bachelor of Engineering in Computer Engineering.

_____

Supervisor
Er. Pradip Adhikari
National College of Engineering
Talchhikhel, Lalitpur

_____

Project Co-ordinator
Er. Mohan Maharjan
National College of Engineering
Talchhikhel, Lalitpur

_____

External Examiner
Ganesh Gautam
Asst. Professor,
Institute of Engineering,
Tribhuwan University

_____

Head of Department
Er. Pradip Adhikari
National College of Engineering
Talchhikhel, Lalitpur

DATE OF APPROVAL:

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our supervisor **Er. Pradip Adhikari** for his guidance on our project study, research and accomplishment of project. His supervision helped us in improvising and the successful completion of our project. We feel blessed for having a supervisor like **Er. Pradip Adhikari.**

We would also like to express our special thanks to our teachers: Er. Mohan Maharjan and Er. Rebanta Aryal, Er. Anup Shrestha and Er. Sharmila Bista for managing their time to provide us with their helping hands to clear our confusions on various topics that kept going with the project despite of the difficulties that we faced on the way. We thank our senior students of **National College of Engineering** for the immense help with the project and providing us with the encouragement.

Last but not least, we would like to thank **NCE Administration**, under which we took this project and our friends and family who became our main inspiration to carry out this project.

ABSTRACT

Nepali Character Recognition (NCR) is a system for converting hand written text to notational characters. It is a special problem in the domain of Pattern Recognition. It deals with the recognition of optically processed characters that are being transformed into an electronic version that can be manipulated by a computer. In this Project, the system is implemented using CNN algorithm. There are various steps involved in this system. The first Process is preprocessing. The input dataset which consists of handwritten character will be loaded and go through some process like reshaping them in suitable array and are normalized. Next, the dataset 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 dense layers to perform the classification being based on the extracted features. A simple and interactive page is created as GUI in order to allow user to draw the character and see the result. This system tries to predict best possible result. This system uses python programming language and different hardware platforms, software platforms to make the system as a whole.

Keyword:

NCR, Neural Network, Image Processing, Devanagari, CNN

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

- ADAM: Adaptive Momentum
- AI: Artificial Intelligence
- ANN: Artificial Neural Network
- CNN: Convolution Neural Network
- GUI: Graphical User Interface
- MLPN: Multi-Layer Perceptron network
- MSE: Mean Squared Error
- NCR: Nepali Character Recognition
- NN: Neural Network
- OCR: Optical Character Recognition
- ReLU: Rectified Linear Unit
- SGD: Stochastic Gradient Descent

1. INTRODUCTION

1.1. Background

1.1.1. Nepali Characters

Devanagari is the national font of Nepal and also used throughout the various part of India also. It has 10 numeral characters(०, १, २, ३, ४, ५, ६, ७, ८, ९) and 36 consonants (क, ख, ग, घ, ङ, च, छ, ज, झ, ञ, ट, ठ, ड, ढ, ण, त, थ, द, ध, न, प, फ, ब, भ, म, य, र, ल, व, श, ष, स, ह, क्ष, त्र, ज्ञ) with 13 vowels. Devanagari Characters have some characters with similar structures. The 'ब' and 'व' are different in only the cross inside circle. Similarly 'ङ' and 'ड' have only difference is dot. Also the character 'प',' म ', and 'य' are almost same. The character '२' and 'र' are almost same. Some characters like 'क्ष', 'त्र', 'ज्ञ' are derived from other previous characters like 'ग', 'य' etc.

1.1.2. Optical Character Recognition

Extracting textual information from the natural image is one of the tedious tasks to perform as it has many practical applications. Unlike human brain which has the capability to very easily recognize the characters given, machines are not intelligent enough to perceive the information available in image. Therefore, a large number of research efforts have been put forward that attempts to transform a document image to format understandable for machine. In modern society, we rely heavily on Computers to process huge volumes of data. For economic reasons to business requirements, there is a great demand to quickly input the mountain of information to the computer. Character Recognition is the field of research in Artificial Intelligence, Computer Vision and Pattern Recognition.

Optical Character Recognition (OCR) is the process of text extraction from images of handwritten or typewritten characters. It deals with the recognition of optically processed characters that are being transformed into an electronic version that can be manipulated by a computer. Optical Character Recognition has drawn extensive attention and attracted research in recent times in various academic and productive fields. The optical recognition is necessary in reading number plates, scanning documents etc. Various research has been done to scan documents using different techniques. There have been significant researches and attempts in case of English language texts. Some programs can also accurately extract the characters drawn. But there have been no noteworthy attempts in the case of Nepali Characters.

As far as the recent trend is concerned, there is a strong resurging interest in the neural-network-based learning because of its superior performance in many image/video understanding applications nowadays. The recent success of deep neural networks is due to the availability of large amount labeled training data and more efficient computing hardware. It is called deep learning because we observe better performance improvement when adding more layers. There are two common neural-network architectures. The one which we have used in this project is Convolutional Neural Networks (CNNs). [9]

1.1.3. Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Some Machine Learning Terminologies are:

a. **Supervised Learning:**

Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

b. Unsupervised Learning:

Unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

c. Optimization:

It is the most essential ingredient in the recipe of machine learning algorithms. It starts with defining some kind of loss function/cost function and ends with minimizing the loss using one or the other optimization routine. The choice of optimization algorithm can make a difference between getting a good accuracy in hours or days. The applications of optimization are limitless and is widely researched topic in industry as well as academia. Examples are SGD (Stochastic Gradient Descent), ADAM, Gradient Descent etc.

d. Loss:

It is a non-negative value, where the robustness of model increases along with the decrease of the value of loss function. Common loss functions are MSE (Mean Square Error), Categorical Cross Entropy etc.

1.1.4. Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Convolutional Neural Networks along with other Neural Networks are used on Deep Learning techniques.

1.1.4.1. Convolutional neural networks

Convolutional Neural Network (CNN) is a special type of multilayer perceptron; a feed-forward neural network trained in supervised mode using a gradient descent propagation learning algorithm [8] that minimizes a loss function. It is one of the most successive machine learning architectures in computer vision and has achieve state-of-the-art results in such tasks as optical character recognition, generic objects recognition, real-time face detection and pose estimation, speech recognition, license plate recognition etc. Its strategy is to extract simple features at higher resolution and transform them to complex features at lower resolution.

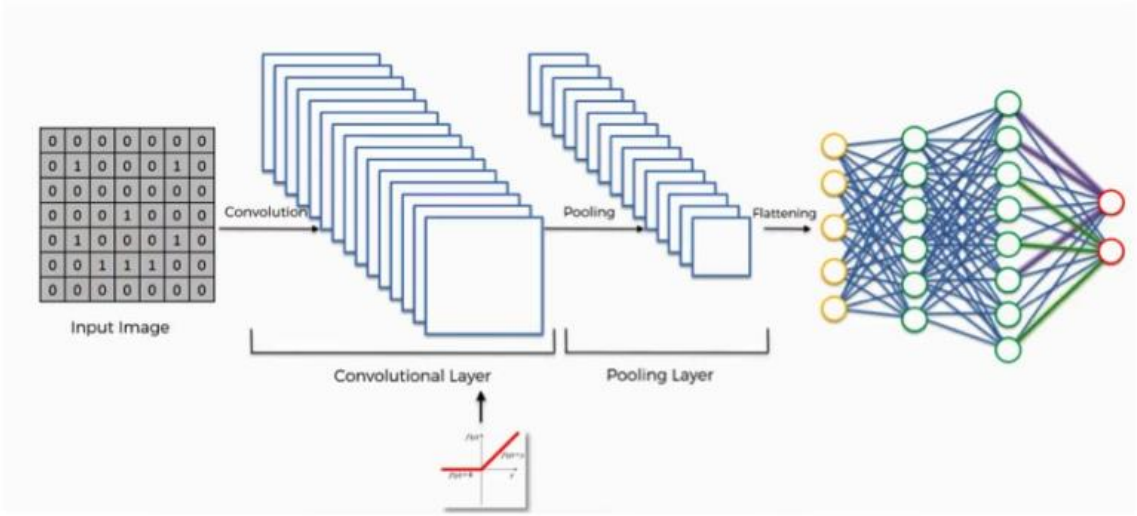

Figure 1.1-1 CNN

1.2. Problem Statement

There are a number of problems that exist in Nepali scenario which are needed to be solved through the Nepali Character recognition system. A lot of research is required for the development of sophisticated systems which can recognize Nepali characters. Building an optical character recognition in Nepali text is even challenging than that of English texts. First, there are much more characters in Nepali than in English – including all attachments and combinations, there are more than 400 distinct characters that need to be recognized and classified correctly. Second, the characters are joined.so to successfully attempt the project, we have to go through deep learning technologies in detail and many more.

The systems which are made till now are not properly implemented although it has a lot of prospects in various fields like banking systems, postal offices, government offices etc. In government offices, using Nepali script in all documents is mandatory; however, editing these

documents is a tedious task since a stable optical recognition system has not been implemented yet effectively because of which people operating these organizations a hard time processing and recording their documents have. Similarly, Nepali-written braille books are not as common as their English version because of lack of good Nepali character recognition systems.

1.2 Aim and Objectives

The aim of this project is to build a complete Automatic Recognition system for handwritten Nepali Character and learn about image processing technique, machine learning algorithm and how to train the machine by self-generating or existing datasets.

The main objectives of this project are:

- To do possible research on Devanagari Characters for Computer Vision.
- To deliver an application which is able to correctly identify every character of a handwritten Nepali character.
- To complete the project in specified time frame.

1.3. Scope and Limitation

This project work considers following scopes and limitations:

- It is only for single language script, that is, Nepali Characters.
- It only considers upright characters.
- Nepali consonants and numeric digits are only included for character recognition.

1.4. Significance of Study

In today's world with the advancement of technology, digital transformation is happening at a much faster rate. So there is a huge significance of Nepali Character Recognition System. The significance of the study can be listed below as:

- Used in areas like Postal Address reading, ancient document digitization, airports to scan passports, tickets etc.
- Keep records in various departments like hospitals, government offices and companies efficiently.
- Can be used in License plate Recognition System.

2. LITERATURE REVIEW

Different papers related to Devanagari OCR, image enhancement, character segmentation and machine learning classifications were studied. Apart from this feasibility study of programming languages like Java, Python, C, C++, etc. to implement OCR was conducted and Python was found to be quite popular and easy to implement.

There have been previous attempts to classify Devanagari text (Nepali, Hindi etc.) for optical character recognition. It is found in those papers that although characters of Devanagari scripts are different from Standard English and European characters, similar algorithms can be applied for optical character recognition.

One of the most useful papers in the field of Devanagari Character Recognition is" Deep Learning Based Large-Scale Handwritten Devanagari Character Recognition" suggesting the approaches for segmentation and recognition of Devanagari characters. [1] In this project, the author used their own dataset consisting of 92 thousand images of 46 different classes of characters of Devanagari script. Along with the dataset, the Author proposed the technique of Deep Convolutional Neural Network (CNN) that have shown superior results to traditional shallow networks in many recognition tasks. Keeping distance with the regular approach of character recognition by Deep CNN, the author highlights the focus on use of dropout and dataset increment approach to improve test accuracy by a margin of nearly 1%. The proposed deep learning architecture scored highest test accuracy of 98.47% on their own dataset.

Another paper with the topic "Optical Character Recognition for Nepali, English Character and Simple Sketch Using Neural Network" deals with the recognition of optically processed characters, with the advent of digital optical scanners. [2] This paper suggests the use of Artificial Neural Network which improved its efficiency and accuracy. This project uses back propagation with two hidden layers for classification of Nepali string with adaptive learning rate. Adaptive learning rate with Gradient descent algorithm was implemented in Neural net with 2 hidden layers and MSE was used as the performance criteria. De-noised test sheets were carefully segmented and inputted in trained neural net which resulted higher accuracy.

Another prominent approach for Images for varied background is described on "Shape Feature and Fuzzy Logic Based Offline Devanagari Handwritten Optical Character Recognition" for the recognition of handwritten Devanagari characters free from normalization thereby giving flexibility and allowing size variation. [3] In this proposed work, the researchers attempt to recognize 45 isolated handwritten characters in Devanagari script. They used database

collected from writers of varied backgrounds where 60% of the database is used for feature extraction. They scan the documents and the digitized images are subjected to preprocessing techniques like filtering, binarization, skeletonization and pruning. The feature extraction module segments the character in segment strokes using modified thinning algorithm. However, errors were observed when the characters were not segmented accurately because of overwriting and different ways of writing the characters in which case this paper would lose some of its effectiveness.

Another approach for text extraction is proposed which has simple and accurate approach for text extraction without using many parameters where the researchers used a very simple FAST algorithm approach i.e., the corner point approach. [4] Firstly, they divided the image into blocks and their density in each block was checked. The denser blocks were labeled as text blocks and the less dense blocks were recognized as the image region or noise. Then they checked the connectivity of the blocks to group the blocks so that the text part can be isolated from the image. They stated that this method is very fast, less complex and versatile and in addition it could be used to detect various languages, handwriting and even images with a lot of noise and blur. However, this method loses a little bit of its effectiveness when it comes to big fonts and for specific pictures for which the corners are responding too much. Through this method they were able to get an average accuracy higher than 90%.

Mohammed Z. Khedher describes in his paper that the recognition of character greatly depends upon evaluation of only selected important features. [5] They used a database of handwritten samples of the same text from 48 different persons. Firstly, they preprocessed unknown texts using grid removal, splitting, thinning, etc. to be presented in later stages in a manageable form. Then the system extracts only the important features carried by the characters. They used 18 features of unknown character and compared them with the features of possible characters from the database and reported most similar character as match which was sufficient to achieve good recognition rate letter shapes of Arabic languages. The authors were able to get more accuracy by using Categorized weights rather than Equal weights. Through the selection of only important selected features, they gained accuracies of 88% and 70% for numbers and letters, respectively. The paper concludes that due to large variation in writing styles in Arabic language, each of the variation should be treated separately.

Lastly, the paper “Handwritten English Character Recognition Using Neural Network” describes the use of Multilayer perceptron for recognizing Handwritten English characters.

Firstly, they acquired the sample by scanning the characters. Then, those scanned characters are converted to binary pixels and the skeletonization process is used to remove the extra pixels not belonging to the character's part. After that, normalization is performed so that all characters could become in equal dimensions of matrix. In this paper, to extract the information from the boundary of a handwritten character, the eight-neighbor adjacent method has been adopted which scans the binary image until it finds the boundary. In this paper, for character recognition in neural Feed Forward Multi-Layer Perceptron network (MLPN) with one hidden layer and one output layer has been used. [6] At last, they developed a system for recognizing handwritten English characters showing the Fourier descriptors with back propagation network yielding recognition accuracy of 94%.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1. Overall Description

3.1.1. Product Perspective

The Nepali Character Recognition System that is mentioned in this document aims to provide an effective way to convert the handwritten characters into a digital format that can be edited and put to use elsewhere.

3.1.2. Product Functions

The Character Recognition system provides capability to correctly identify the Devanagari characters using Convolution Neural Network. The system should convert the given character into digital format. The character drawn is compared to the trained model and then output is generated and displayed according to the highest probability thus obtained.

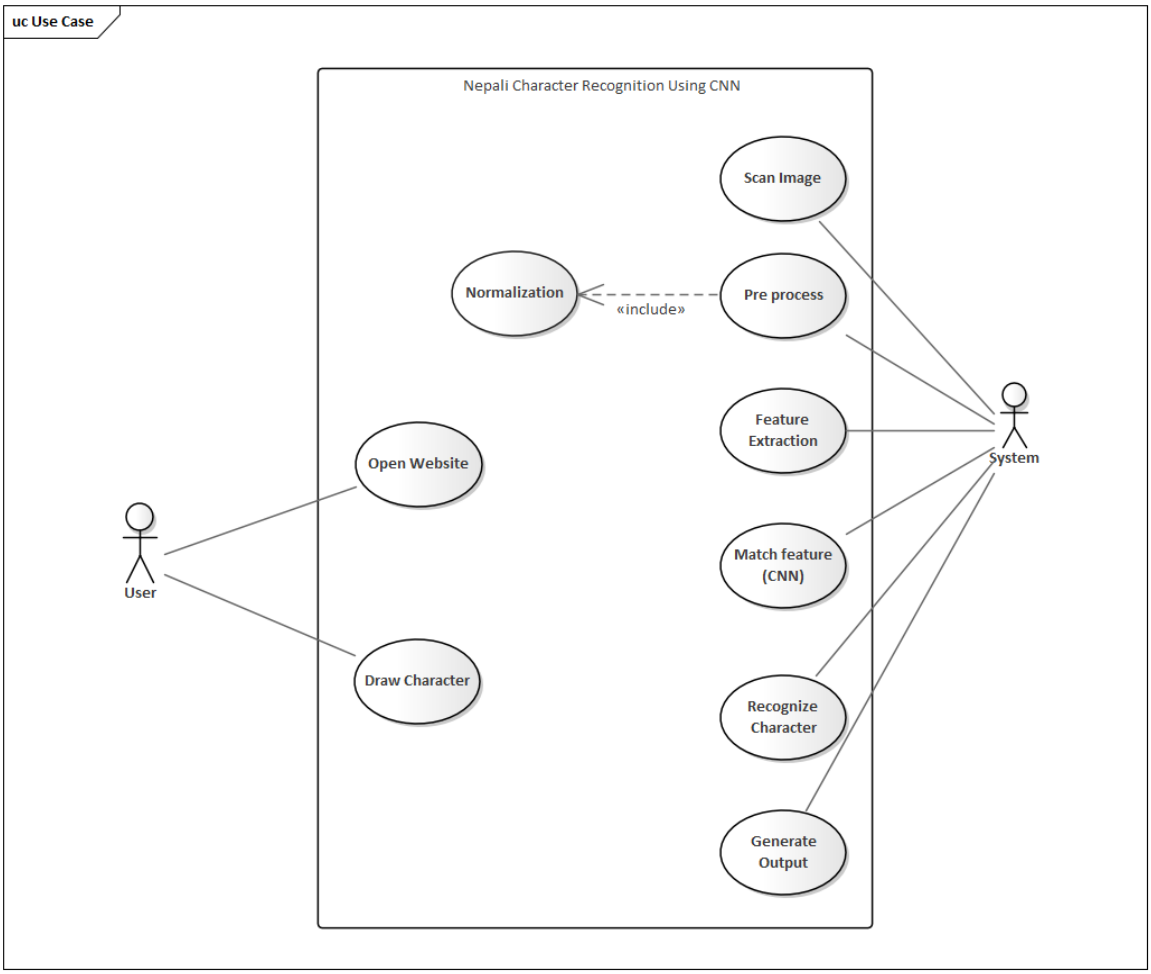

Figure 3.1-1 Use case Diagram for NCR

3.1.3. Operating Environment

The website is python based with the UI made in flask that shall operate in all famous browsers.

3.1.4. User Documentation

The user documentation will be provided in the Web Application which includes Terms & Conditions, User Policy and Software policies.

3.2. Requirement Analysis

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.

3.2.1. Functional Requirements

Functional requirement explains what has to be done by identifying the necessary task, action or activity that must be accomplished. Simply, the functional requirements for a system describe what system does. The following are the functional requirement that we hope to accomplish in our project:

Table 3.2-1 Functional Requirements

R.id	**Functional Requirements**
1	The given system should recognize Handwritten Nepali character from the character drawn.
2	System should display the predicted character using a graphical UI
3	System must provide desired accuracy for character recognition.

3.2.2. Non-functional Requirements

As the name suggests these are the requirements that are not directly interacted with specific functions delivered by the system. The non-functional requirements are:

Table 3.2-2 Non Functional Requirements

Performance requirement:	• Should be able to predict the results smoothly. • Should be reliable and efficient.
Flexibility:	• Model should be flexible since the data sets can be changed while training.
Maintainability:	• The source code should be simple so can be maintained easily in the case of any errors.

Scalability	• The model should be scalable since it works for small as well as large data sets.
Testability	• The model should be testable for various test data sets

3.2.3. System Design Requirements

3.2.3.1. Software Requirements

Table 3.2-3 Software Requirements

Name of Component	**Specifications (Minimum Requirement)**
Operating System	Windows 7, Linux Distros
Language	Python 3.7 With Anaconda Runtime Environment
Software Development Kit	Python
GUI	Flask

3.2.3.2. Hardware Requirements

Table 3.2-4 Hardware Requirements

Name of Component	**Specifications (Minimum Requirement)**
Processor	2.50 GHz
RAM	8 GB
GPU	4 GB
Hard disk	1 TB
Display	1024 x 768
Keyboard	104 keys

3.3. Model Used

Agile model was adopted in our project. The agile development model is a type of incremental model which results in small incremental release building on previous functionality. Agile is a

term used to describe software development approaches that employ continual planning, learning, improvement, team collaboration, evolutionary development, and early delivery.

The reason we decided using agile model is because this model allows changes to the system even in the late stages of development. Thus, this approach allows us to work with our requirements while also allowing us to carry on with our coding process. This allowed us to make changes to our approach and select a suitable method to approach any issue while leaving room for error. In the case of our system, at initial stage we had decided to do recognition only on printed Nepali Characters but later on we decided to go further out by recognizing hand-written characters. So, agile model allowed us to present late changes in requirements during our development. Another reason for using this model is because, later on in higher iteration of our development it will be easier to increase the range of implement of our project if desired, say: To scan a handwritten book and convert it to a digital copy.

3.4. Feasibility Analysis

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.

3.4.1. Economic Feasibility

Economic Feasibility refers to an economic benefits and income being greater than the expenditure. A project is economically feasible if economic benefits are outweighed by the project's cost. The economic benefits of our project does surpass the cost and is considered to be economically feasible. Use of our OCR can replace the paying staffs needed at offices for file maintenance at various offices also.

3.4.2. Technical Feasibility

The technical feasibility captures the required hardware and software components for operating the system. We used Python as our programming language and CNN for image recognition, which we had to learn during the course of development. The necessary resources like python programming language, anaconda distribution, datasets, environment, software libraries and flask were made available for developing the project and hence, made technically feasible. The application was made to be able to recognize all the Devanagari characters from an image.

3.4.3. Operational Feasibility

The aspect of study is to check the level of acceptance of the system by the user. The system that we have designed is easy to operate with necessary User Interface provided, since all the user has to do is provide an input by simply drawing it in the application. The application has already been trained for the specified character detection, so that the end user doesn't have to go through the training part.

4. SYSTEM ANALYSIS

4.1. System Sequential Diagram

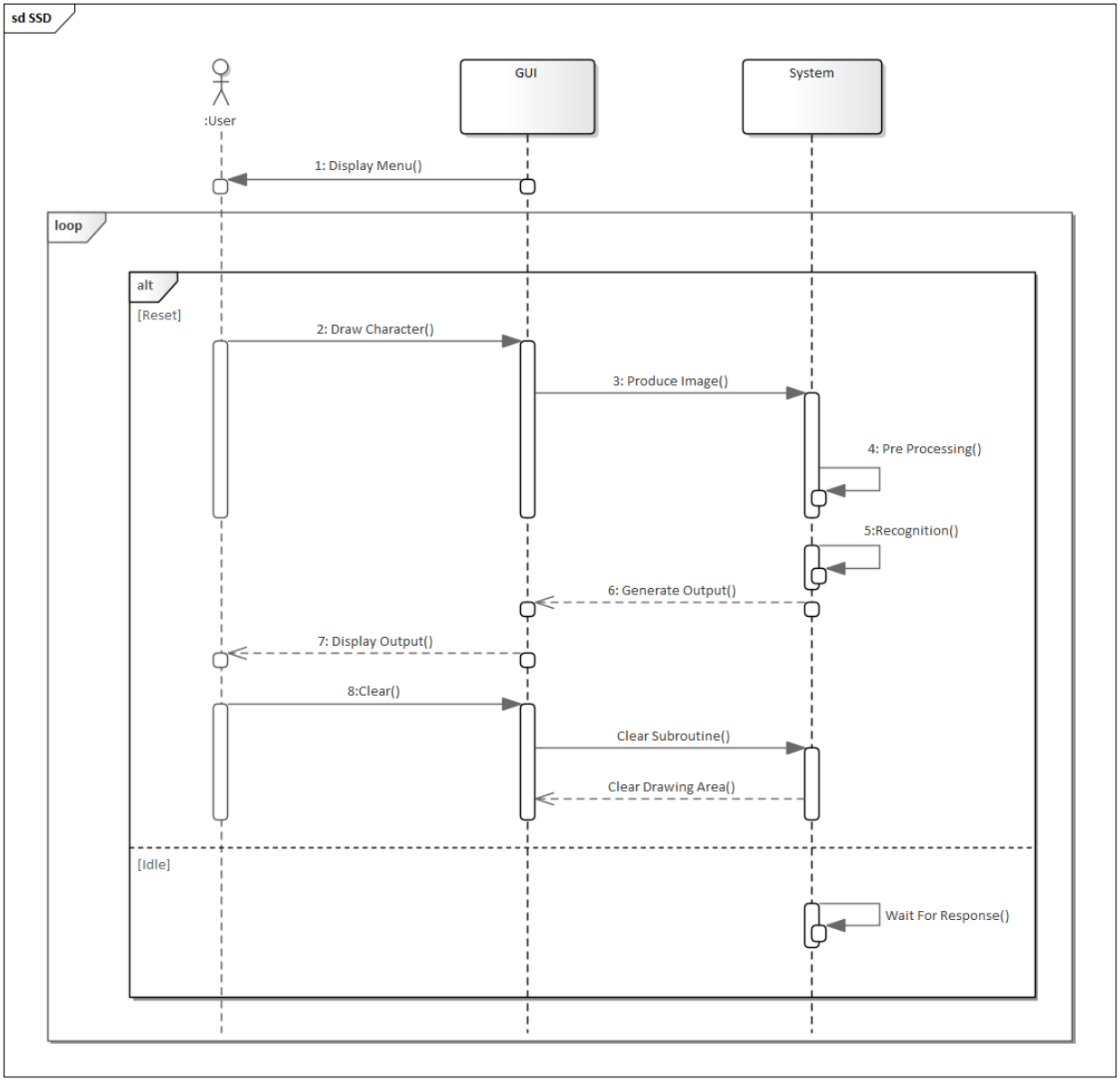

Figure 4.1-1 System Sequential Diagram for NCR

4.2. Activity Diagram

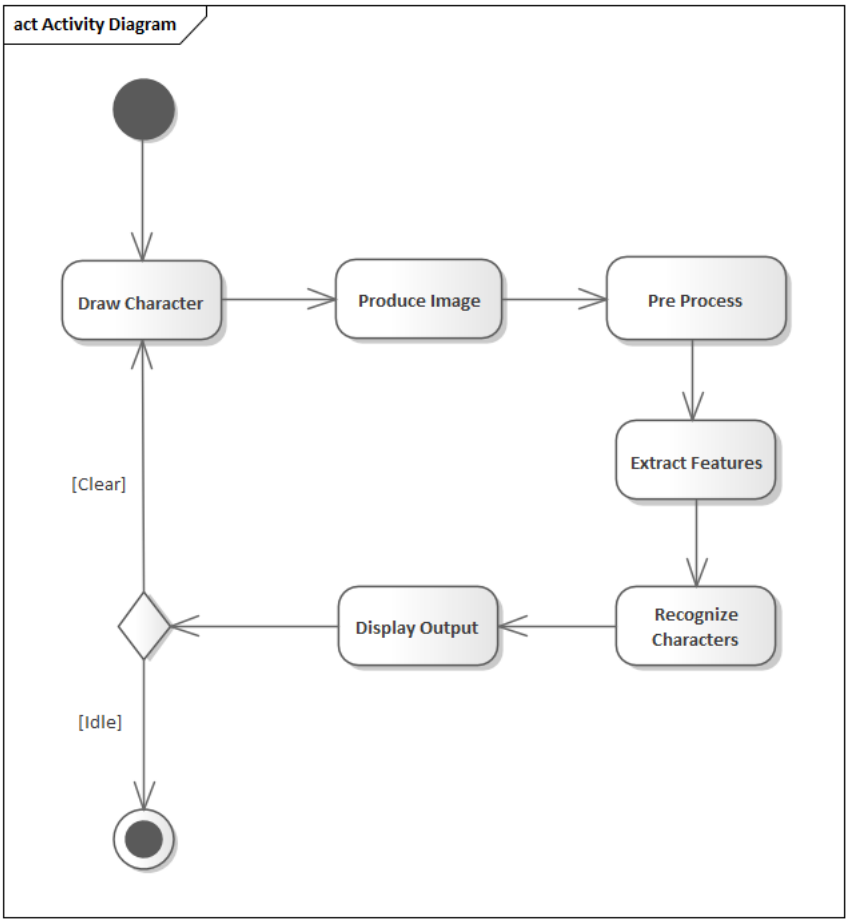

Figure 4.2-1 Activity Diagram for NCR

5. SYSTEM DESIGN

5.1. User-Interface Design

The UI of any application is supposed to be clear and easy to use. We have developed a UI using ‘flask’ that is easy to understand and simple to execute through the use of interactive buttons. The UI enables the user to draw a character with the use of a mouse and then press the predict button to predict the drawn character. Similarly pressing the clear button will clear the drawing window.

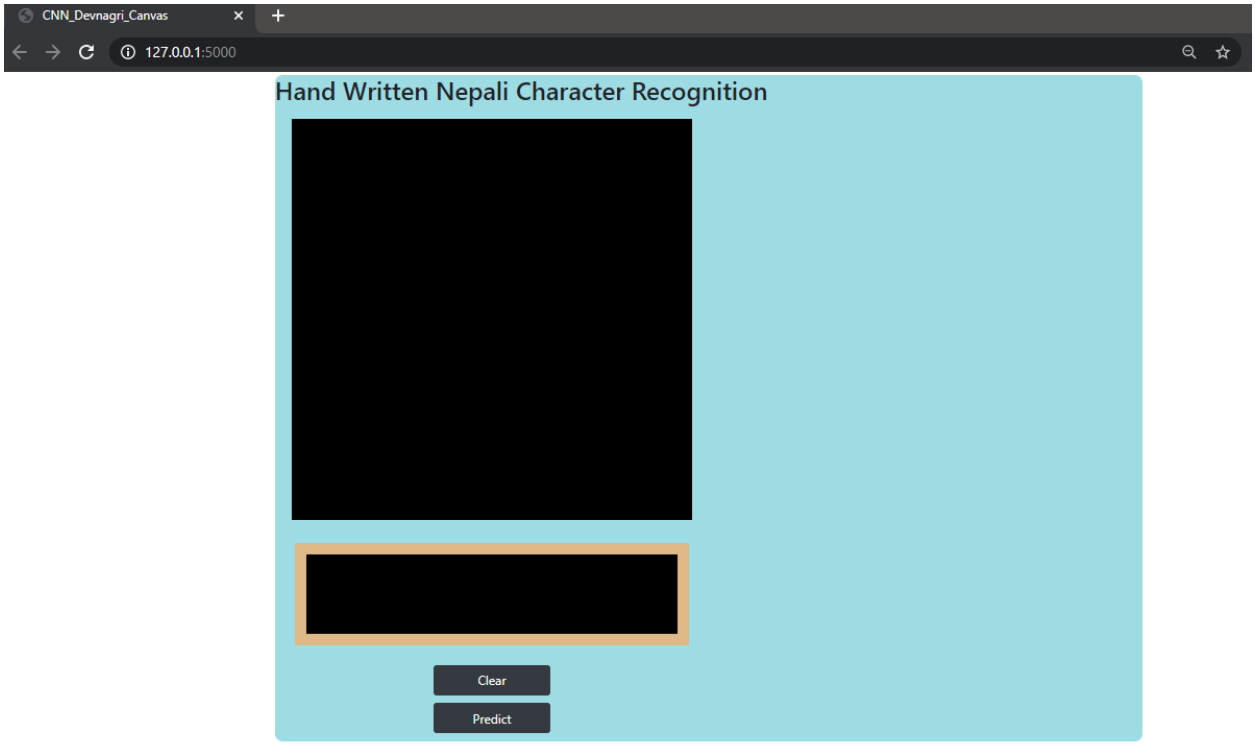

Figure 5.1-1 User Interface

6. METHODOLOGY

6.1. Implementation

6.1.1. Dataset Selection and Preparation

In supervised Learning Applications, we need a large amount of labeled data for the purpose of training. A labeled data is a data that is similar to the expected input and we know the expected output for it. In our Project, the image of the character and output of the character and output is machine encoded form of the character.

In machine learning applications, dataset can be obtained by getting the samples having the close resemblance to the real-world data. Thus, in this project we have created dataset in the following ways:

The dataset already exists in English Format. We have existing dataset found from Kaggle.com having total datasets= 92,000 and train sets=73600 and test sets = 18400 with having 46 classes.

Figure 6.1-1 Dataset visualization

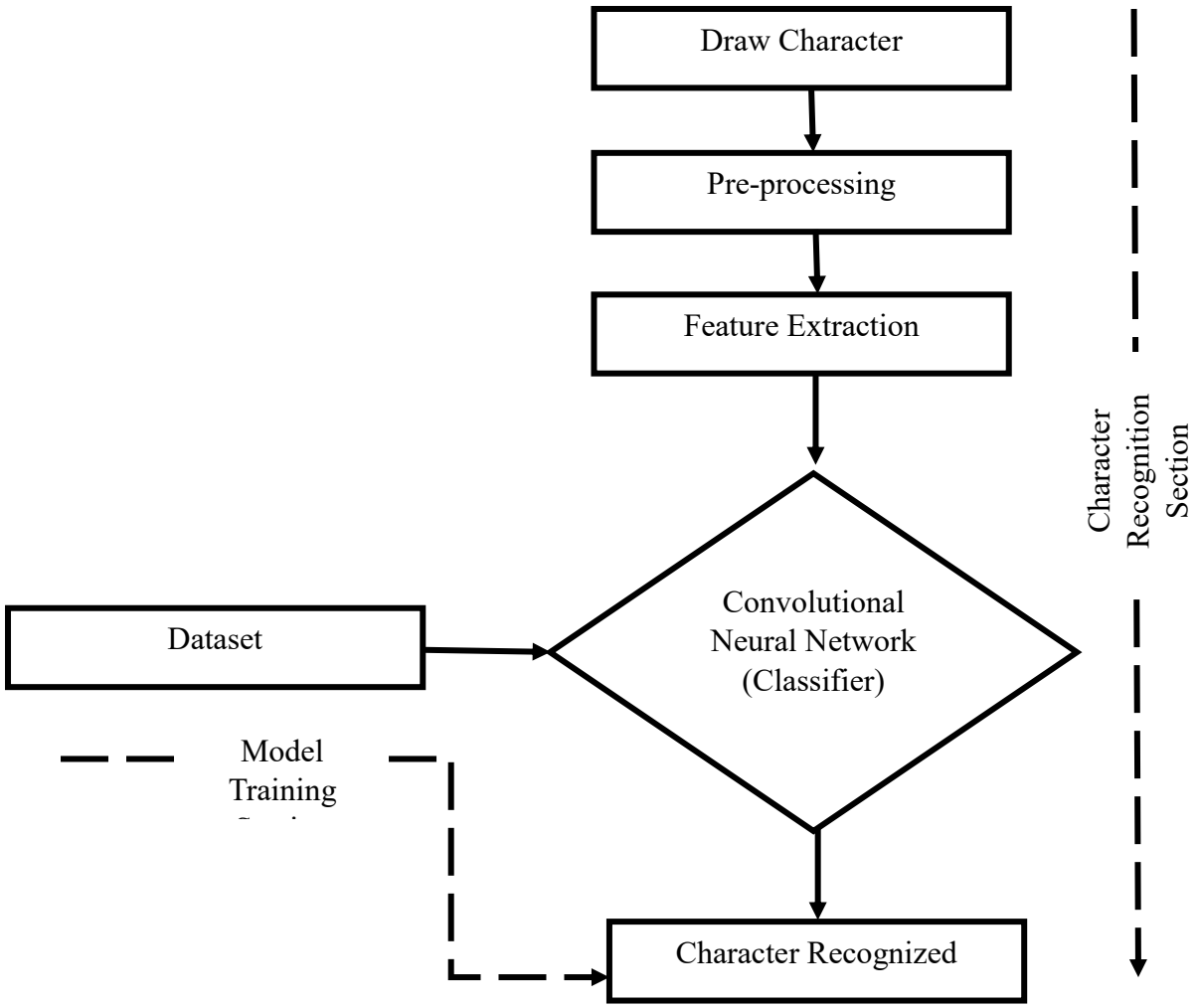

Figure 6.1-2 Block Diagram

6.1.2. Input Image

We use a GUI where image can be drawn and then fed to the system.

6.1.3. Pre-processing

First, the image is given as handwritten character to the system. The image can be of any size but the input to the system must be of fixed size, i.e., the size of canvas (512*512) so that the image resize takes place without distortion. Then the image is resized to the size that the model expects (32, 32).

6.1.4. Character Recognition

The character obtained is received by this module with proper preprocessing. Model is saved after training (Training portion is discussed in Algorithm) in some format. The preprocessed image is passed to the model where prediction occurs. The predicted output is shown in the Interface.

6.2. Architecture

1. **Convolution**: The 32*32 image was convolved using 3*3 filter and 32 feature maps was obtained.
2. **MaxPooling2D**: The maximum value from the 2*2 matrix's portion were selected and a matrix was obtained.
3. **ReLU**: The obtained image was then passed through a ReLU activation and a non-negative was obtained. It helps to break linearity of the convolved feature map.
4. **Convolution**: It is applied and another 64 features map obtained.
5. **ReLU**: It is applied once again.
6. **MaxPooling2D**: It is applied once again.
7. **Convolution**: It is applied twice and another 64 features map obtained
8. **Flatten**: It flattens the input.
9. **Dense**: It does the operation – output =activation (dot (input, kernel) +bias), dimensionality of outer space being 128.
10. **Dropout**: Dropout is used to reduce the overfitting of the data. Units used= 0.4.
11. **Dense**: Applied once again with the dimensionality of 64.
12. **Dropout**: Units used= 0.2.

13. **Dense**: Applied once again with the dimensionality of 46 for output classes and softmax function.

6.3. Interface Design

For the interface, we have used basic HTML, CSS and JavaScript. With the help of flask as backend, we are able to upload the image. Once the predict button is pressed, the input of the image is sent to the flask backend for prediction where model is already loaded. Once the prediction is done, the result is sent from backend to the same html document. The result is displayed on the homepage of the local website.

6.4. Convolutional Neural Network

Convolutional Neural Network (CNN or ConvoNet) is a class of deep Neural Networks, most commonly applied to analyzing visual imagery. Basically, CNN are used to classify images, cluster them by similarity, and perform object recognition with scenes. They are regularized version of Multilayer Perceptron.

A tremendous amount of interest has emerged for CNN in recent years as it has been dominant method in computer vision tasks since the astonishing results were obtained. Medical Research is no exception, as CNN has achieved expert-level performances in various fields. Likewise CNN is used to perform Optical Character Recognition to digitize text.

The efficiency of Convolutional nets in image recognition is one of the main reasons why the works has woken up to the efficiency of deep learning. They are powering major advances in Computer Vision (CV), which has obvious applications for self-driving cars, robotics, drones, security, medical diagnoses, and treatments for the visually impaired. Here are the different steps that are followed in CNN algorithm:

6.4.1. Convolutional Layer

The primary purpose of Convolution in Convolutional Neural Network is to extract features from the input image. A kernel (also called filter) is a smaller-sized matrix in comparison to input dimension of the image, which consists of real value entries. Kernel then is convolved with the input volume to obtain so called ‘Activation maps’ or ‘Feature maps’. Activation maps indicate the activated regions where the features specific to the kernel have been detected in the input. Instead of wholly connecting the input to the neural network we connect the convolved matrix which is smaller than the input matrix.

At first, the input image which is dataset is already preprocessed. The proposed architecture has four convolution which is enough for feature extraction. The first Convolution layer has 32 filters/output channels, with the kernel size of 3*3 and an activation function. The layer is able to detect pattern (edges, shapes, textures, objects etc.) in the images. Similarly, the next 3 layers have 64 filters, each of size 3*3 which is enough for feature extraction. Here a filter with a kernel size 3*3 is just a matrix initialized with random numbers.

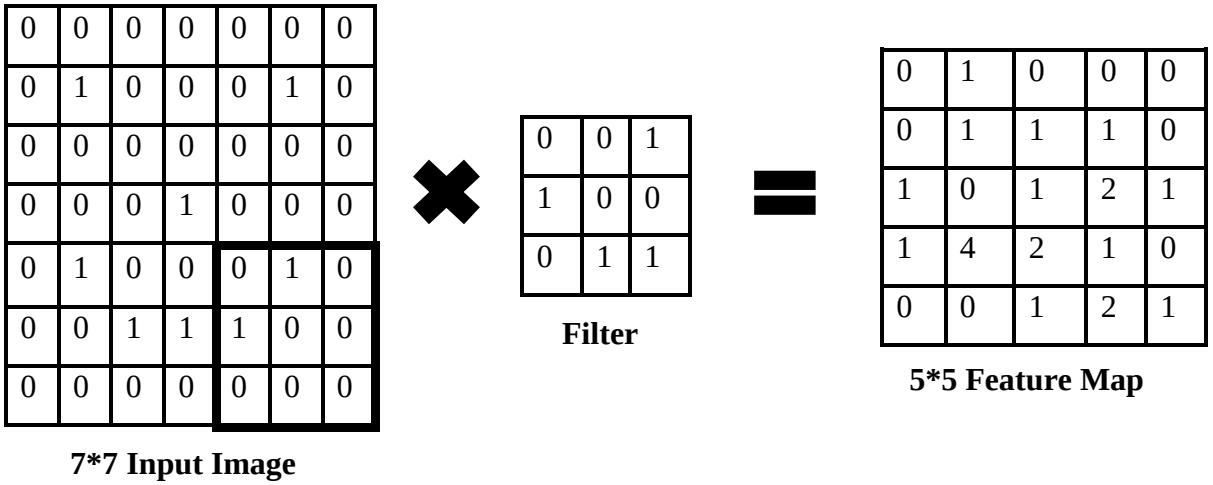

*Figure 6.4-1 Convolution on (7*7) image with (3*3) filter*

6.4.2. Max Pooling

The next layer in a convolution is Max Pooling with the pool size of 2*2. Max Pooling reduces the dimensionality of the images by reducing the number of pixels in output from previous Convolutional Layer.

We have taken two poling layers for training the model. 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. This process is repeated once more.

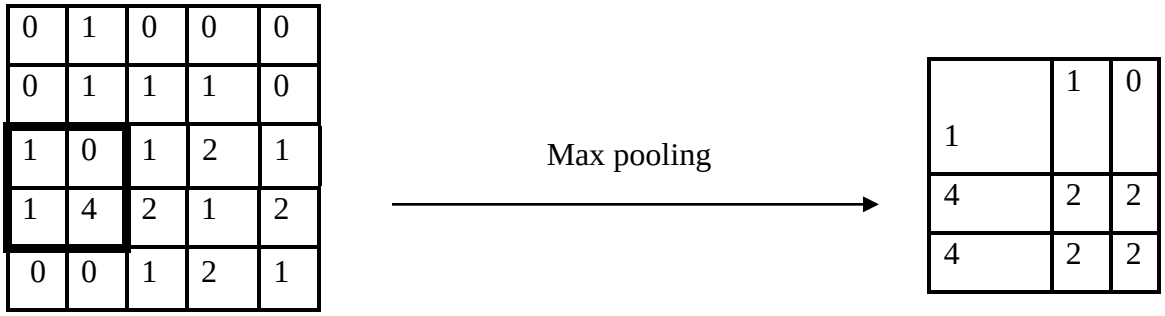

*Figure 6.4-2 Max pooling operation with 2*2*

6.4.3. Flattening

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.

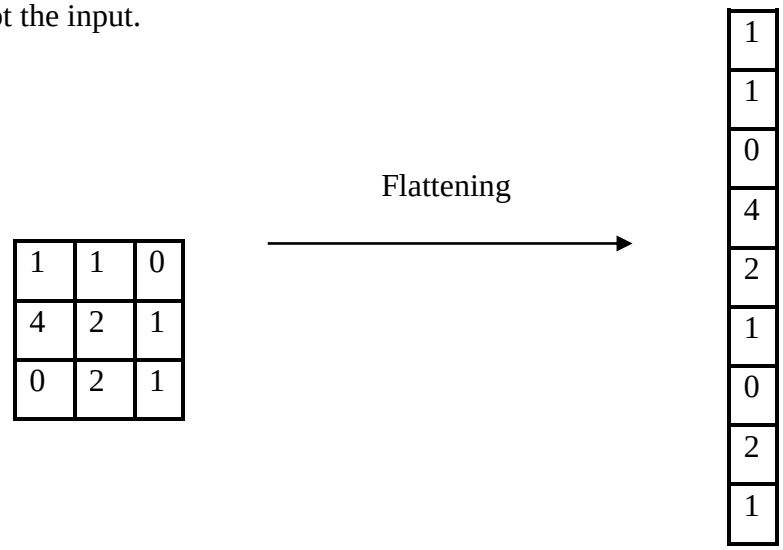

Figure 6.4-3 Flattening

6.4.4. Full Connection

Next Layer is Fully Connected Layer (Dense Layer) where every layer is connected to the final layer which gives the probability values for each class. A flatten function is used to convert the outputs to probability values for each class. And the class with the max probability will be selected as the label.

We have considered 3 dense layers. The first dense layer has 128 output dimensions and the next with 64. 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.

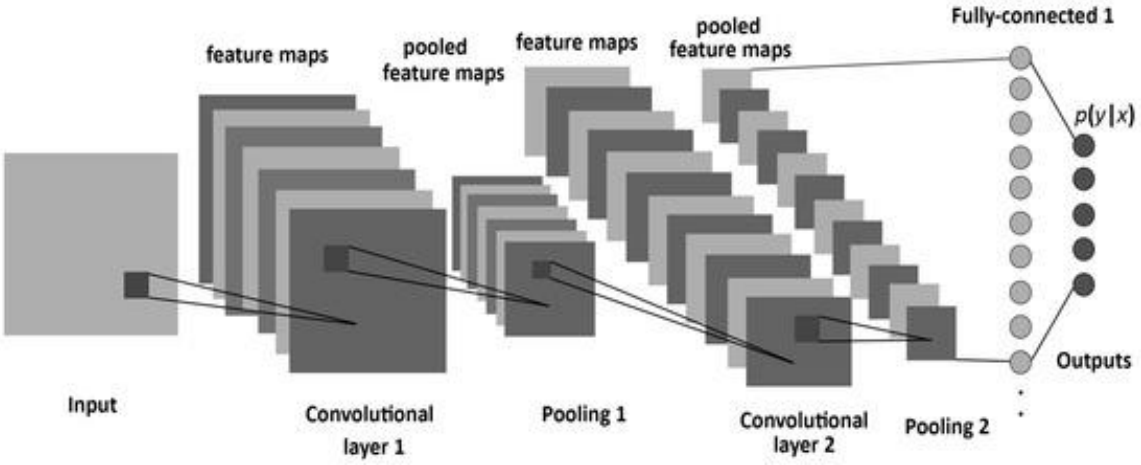

Figure 6.4-4 CNN with Full Connection

6.5. Activation Function

We will be using Rectified Linear Units (ReLU) activation function in our layers except in last output layer. [7] 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. When you get the input is positive, the derivative is just 1, so there isn't the squeezing effect. It can be written as:

$$f(x) = \max(0, x)$$

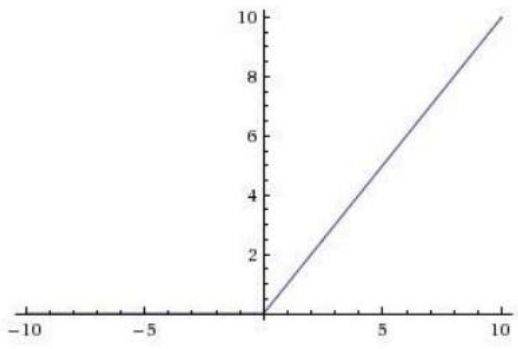

Figure 6.5-1 ReLU activation function

1	14	-9	4
-2	-20	10	6
-3	3	11	1
2	54	-2	80

1	14	0	4
0	0	10	6
0	3	11	1
2	54	0	80

Figure 6.5-2 Activation by RELU function

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.

7. TESTING

The testing phase can be carried out manually or by using automated testing tools to ensure each component works fine. After the project is ready its various components were tested in terms of quality, performance to make it error free and remove any sort of technical jargons. Testing also is done to measure the difference between the desired and the developed system. Testing is need on development cycle of system to ensure that the system's every component works fine.

7.1. Unit Testing

During the coding phase each individual module was tested to check whether it works properly or not. Different errors found during unit testing were debugged. Some of the major test cases are listed below:

Unit Testing of Upload Image

Test Scenario: Unit testing of upload Image to the system.

Table 7.1-1 Unit testing of upload Image to the system.

Test Scenario	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Remarks**
Unit Testing of Upload Image	1. Choose to Upload Image	Image with name 'test.jpg'	Save image in upload folder and display	As Expected	Pass

Unit Testing of Resize Image

Test Scenario: Unit testing of Resize Image

Table 7.1-2 Unit testing of Resize Image

Test Scenario	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Remarks**
Unit Testing of	**1.** Choose to Upload Image of any size.	Image with name 'test.jpg'	Image with name 'test.jpg' of size	Output image is	Pass

Resize Image	**2.** Image is resized to target size without distortion.	upload of size 512*512.	512*512 resized to 32*32 without distortion.	resized to 32*32.	

7.2. Integration Testing

Integration testing integrates individual modules and tested as groups. Integration testing takes the unit tested modules, group them into the larger aggregates, applies tests and delivers the output.

Test Scenario: Integration Testing of Prediction of model with Web Platform

Table 7.2-1 Integration Testing of Prediction of model with Web Platform

Test Scenario	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Remarks**
Upload Image	1. Choose to Upload Image	Image with name ‘test.jpg’	Save image in upload folder and display	As Expected	Pass
Prediction of Image	1. Choose to Upload Image 2. Submit the image for prediction	Image with name ‘predict.jpg’ with handwritten character will go for prediction	Save image in upload folder and open the new window with correct prediction	As Expected	Pass

7.3. System Testing

System testing has done after integrating testing in order to ensure that the whole systems functions properly. After the integration testing the whole system working process was checked. The output was as per the system specifications and hence the system was found to work properly.

Test Scenario: System testing of whole system and their components

Table 7.3-1 System testing of whole system and their components

Test Scenario	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Remarks**
Unit Testing of Upload Image	1. Choose to Upload Image	Image with name 'test.jpg'	Save image in upload folder and display	As Expected	Pass
Load Data	Load data into batches onto folder for training	Image is loaded into the train	Image, of training and validation set should be loaded with its label	As Expected	Pass
Start Training of Model	**1.** Start training of the Neural Network	All the loaded data passed one by one batches to the model	Model should process images and labels and output loss value on the iteration	As Expected	Pass

8. DISCUSSION AND RESULT

8.1. Product Description

Our program produced a result such that character drawn is detected. The character is exported as an image and then passed to the saved model .The system will be coded since it is research type application where much focus has been given to the research part. Since a lot of dependencies were used, it was a little complicated to develop an executable file. The system we developed would classify character classes.

8.2. Initialization

We developed our convolutional neural network and trained it from scratch. We did not use pre-trained models for initializing the neural network. The neural network was trained with the images selected from dataset. The image dataset available to us was split into training dataset and testing dataset. The dataset was in image format so, it was converted into arrays and stored in matrix form. The dataset was randomly shuffled so that it could be trained accurately. The class vectors were converted into binary class matrices so that they could be fed to our network.

8.3. Implementation Details

Dataset was divided into train set and test set. Train set includes 80 percentage of total dataset which is 73600 and test set includes twenty percentage of total dataset which is 18400. Data set is feed into the neural network .Training is performed with batch size of 32 images. Total epoch was 100. The trained file was then saved into an h5 file format for future use.

```
In [17]: model.summary()
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 15, 15, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 64)          36928
_________________________________________________________________
global_average_pooling2d (Gl (None, 64)                0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 128)               8320
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 46)                2990
=================================================================
Total params: 112,942
Trainable params: 112,878
Non-trainable params: 64
```

Figure 8.3-1 Model Summary

```
230/230 [==============================] - 139s 606ms/step - loss: 2.5887 - accuracy: 0.2895 - val_loss: 2.5606 - val_accuracy: 0.4492
Epoch 2/100
230/230 [==============================] - 82s 357ms/step - loss: 0.9659 - accuracy: 0.7008 - val_loss: 1.4109 - val_accuracy: 0.6108
Epoch 3/100
230/230 [==============================] - 82s 356ms/step - loss: 0.5604 - accuracy: 0.8281 - val_loss: 1.1970 - val_accuracy: 0.665222s - loss: 0.6049 -
accuracy: 0.8107
Epoch 4/100
230/230 [==============================] - 81s 354ms/step - loss: 0.4005 - accuracy: 0.8814 - val_loss: 1.3202 - val_accuracy: 0.6607
Epoch 5/100
230/230 [==============================] - 82s 356ms/step - loss: 0.3227 - accuracy: 0.9013 - val_loss: 1.1217 - val_accuracy: 0.7122
Epoch 6/100
230/230 [==============================] - 82s 354ms/step - loss: 0.2770 - accuracy: 0.9211 - val_loss: 1.1149 - val_accuracy: 0.7211
Epoch 7/100
230/230 [==============================] - 83s 359ms/step - loss: 0.2337 - accuracy: 0.9334 - val_loss: 1.1397 - val_accuracy: 0.7270
Epoch 8/100
230/230 [==============================] - 82s 354ms/step - loss: 0.2099 - accuracy: 0.9409 - val_loss: 1.1039 - val_accuracy: 0.7381
Epoch 9/100
230/230 [==============================] - 82s 356ms/step - loss: 0.1809 - accuracy: 0.9500 - val_loss: 1.4198 - val_accuracy: 0.6842
Epoch 10/100
230/230 [==============================] - 81s 351ms/step - loss: 0.1823 - accuracy: 0.9493 - val_loss: 1.0839 - val_accuracy: 0.7546
Epoch 11/100
230/230 [==============================] - 80s 349ms/step - loss: 0.1615 - accuracy: 0.9548 - val_loss: 1.1297 - val_accuracy: 0.7416
Epoch 12/100
230/230 [==============================] - 80s 350ms/step - loss: 0.1583 - accuracy: 0.9539 - val_loss: 1.1577 - val_accuracy: 0.7406
Epoch 13/100
230/230 [==============================] - 80s 348ms/step - loss: 0.1341 - accuracy: 0.9610 - val_loss: 1.1126 - val_accuracy: 0.7390
Epoch 14/100
230/230 [==============================] - 81s 350ms/step - loss: 0.1305 - accuracy: 0.9640 - val_loss: 1.0970 - val_accuracy: 0.7469
Epoch 15/100
230/230 [==============================] - 81s 353ms/step - loss: 0.1286 - accuracy: 0.9632 - val_loss: 1.0879 - val_accuracy: 0.7516
```

Figure 8.3-2Training of NN

8.4. Result

- Epoch = 100
- Accuracy = 0.99
- Validation Accuracy = 0.778
- Loss = 0.337
- Validation Loss = 1.358

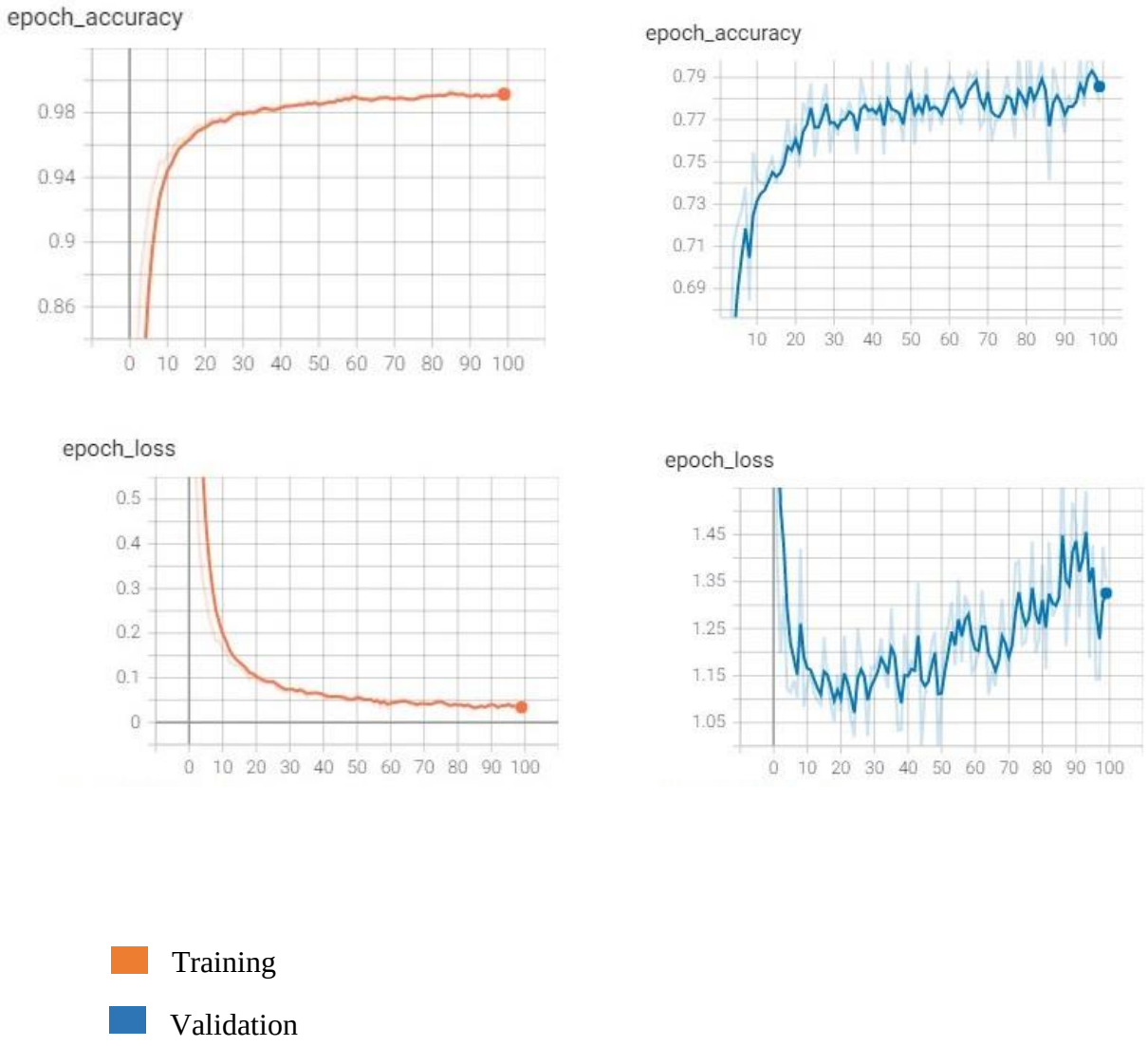

Figure 8.4-1 Obtained output of most Favorable Selection

The system succeeded to obtain the accuracy of 99% for training and 77% for validation. The graphs have been obtained using Tensorboard which comes preinstalled in Tensorflow.

9. CONCLUSION AND FUTURE ENHANCEMENT

9.1. Conclusion

Many regional languages throughout world have different writing styles which can be recognized using proper algorithm and strategies. Classification of characters and learning of image processing techniques is done in this project. Also the scheme through which project is achieved is Convolutional Neural Network scheme. The result which was achieved was correct up to more than 90% of the cases, but it would be improved at the end. This work was basically focused on envisaging methods that can efficiently extract feature vectors from each individual character. The method we came up with gave efficient and effective result both for feature extraction as well as recognition of Devanagari characters. There are also different methods through which 'handwritten character recognition' is achieved. This work will be helpful to the researchers for the work towards other scripts.

9.2. Recommendations

This Character Recognition system can be used in different sectors. In education, this can be used in taking notes. It can be recommended in various government offices to digitize their systems. It can also be used in banks as it can help to know the authentic ways to carry out the transactions more reliably and securely.

9.3. Future Enhancements

Some of our future enhancements include:

- Developing a mobile based applications (iOS and Android).
- Adding the text conversion to different languages.
- Adding the complex characters recognition.
- Develop license plate recognition system.
- Developing the OCR using Raspberry pi and Arduino.
- Developing a reinforcement learning.
- Developing web based application which can store the user details in database.
- Adding Natural Language Processing features.

10.REFERENCES

[1] S. Acharya, A. K. Pant and P. K. Gyawali, "Deep learning based large scale handwritten Devanagari character recognition," 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Kathmandu, Nepal, 2015, pp. 1-6, doi: 10.1109/SKIMA.2015.7400041.

[2] S. Shakya, A. Basnet, S. Sharma, and A. Gurung, "Optical character recognition for nepali, english character and a simple sketch using neural network," Recent Advances in Information and Communication Technology, pp. 45–53, 2016.

[3] Journal of Pattern Recognition Research 4 (2009) 52-68 Received Jul 23, 2008. Revised Aug 28, 2008. Accepted Feb 6, 2009.

[4] Dr. T. Kameswara Rao et al Int J Sci Res CSE & IT. March-April-2019; 5(2): 597-604.

Yusuf Perwej, Ashish Chaturvedi "Neural Networks for Handwritten English Alphabet Recognition" International Journal of Computer Applications (0975 – 8887) Volume 20– No.7, April 2011.

[5] Khedher, Mohammed & Abandah, Gheith & Al-Khawaldeh, Ahmed. (2005). Optimizing Feature Selection for Recognizing Handwritten Arabic Characters. 81-84.

[6] Anita Pal & Dayashankar Singh, "Handwritten English Character Recognition Using Neural Network," International Journal of Computer Science & Communication. Vol. 1, No. 2, July-December 2010, pp. 141-144.

[7] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. IEEE International Conference on Computer Vision (ICCV 2015). 1502. 10.1109/ICCV.2015.123.

[8] Lecun, Yann & Bottou, Leon & Bengio, Y. & Haffner, Patrick. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE. 86. 2278 - 2324. 10.1109/5.726791.c.

[9] Wu, J. (2017). Introduction to Convolutional Neural Networks.

APPENDIX I

Output Screen

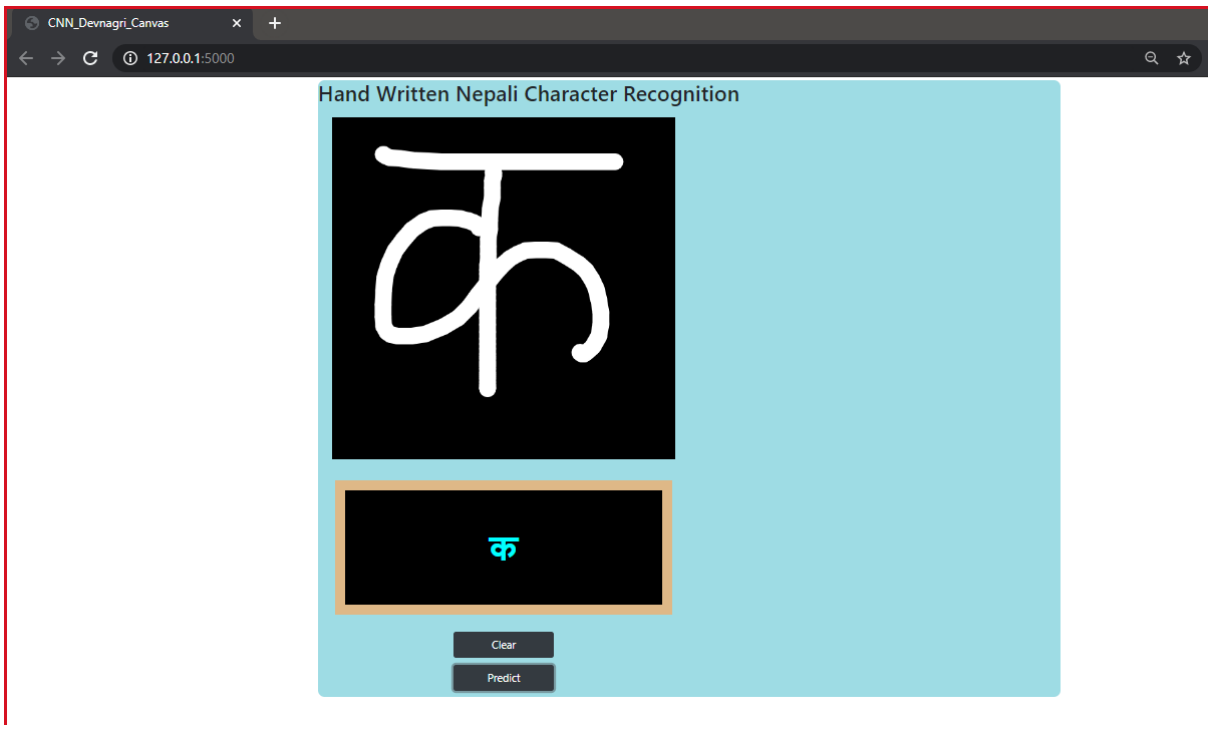

Figure I- Screenshot of UI

APPENDIX II

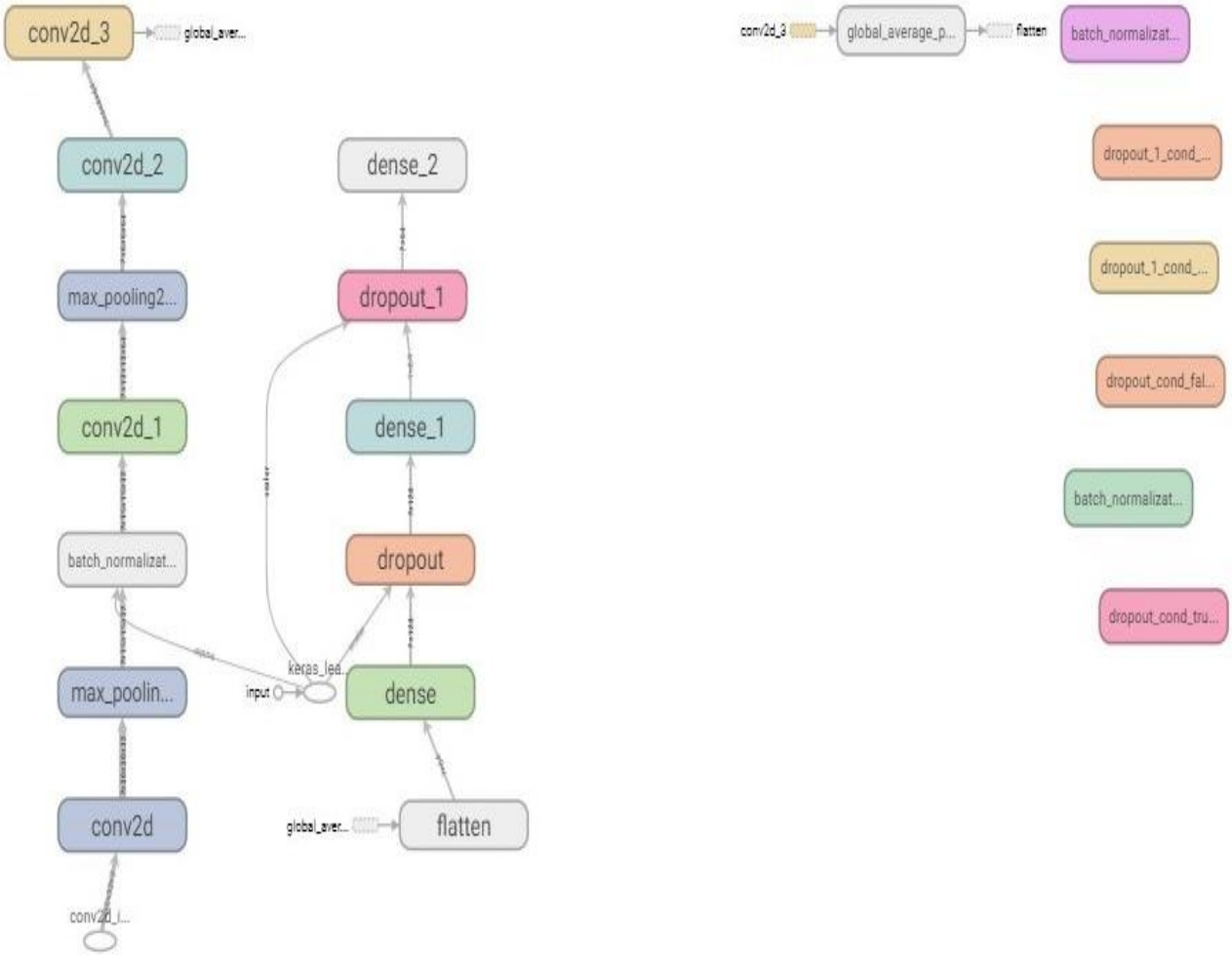

Figure II-Model Visualization Using Tensor Board