Handwritten Devanagari Character Recognition

A report submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications

by

GAURAV

(17419MCA011)

Department of Computer Science

Institute of Science

Banaras Hindu University, Varanasi – 221005

May 2020

CANDIDATE’S DECLARATION

I **Gaurav** hereby certify that the work, which is being presented in the project report, entitled **Handwritten Devanagari Character Recognition**, in partial fulfillment of the requirement for the award of the Degree of **Master of Computer Applications** and submitted to the institution is an authentic record of my/our own work carried out during the period **February-2020** to **May-2020** under the supervision of **Dr. Vivek Kumar Singh**. I also cited the reference about the text(s) /figure(s) /table(s) /equation(s) from where they have been taken.

The matter presented in this report as not been submitted elsewhere for the award of any other degree or diploma from any Institutions.

Date: **Signature of the Candidate**

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge. The Viva-Voce examination of **Gaurav**, M.C.A. Student has been held on __________________.

Signature of
Research Supervisor

Signature of
Head of the Department

ABSTRACT

Devanagari is an ancient script used for over 120 spoken Indo-Aryan languages, including Hindi, Nepali, Marathi, Maithili, Awadhi, Newari and Bhojpuri. This script is used by millions of people in India to write documents in Marathi and Hindi. Most of the Indian mythology is written in this script. Handwritten Devanagari character recognition has gained popularity over the years due to such importance of the script.

Although significant research has been made in full character recognition of Devanagari characters using Convolution neural networks for both feature extraction and classification, the report experiments different classifiers for classifying and predicting the handwritten characters while using CNN and DNN for feature extraction. The scope of report has been widened by making the model to predict partial Devanagari characters while been trained on full characters and vice versa.

A novel technique is proposed for recognition Devanagari language characters using Artificial Neural Network including the schemes of feature extraction of the characters and implemented. The persistency in recognition of characters by the Artificial Neural network was found to be more than 93% of times.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	**Description**
NN	Neural Network
CNN	Convolutional Neural Network
HCR	Handwritten Character Recognition
GUI	Graphical User Interface
px	Pixels
reLU	Rectified Linear Unit
adam	Adaptive Moment Estimation
DNN	Deep Neural Network
SGD	Stochastic Gradient Descent
ANN	Artificial neural network
ConvNet	Convolutional Neural Network
API	Application Programming Interface
HWR	Handwriting Recognition
CPU	Central Processing Unit
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit

CHAPTER 1

INTRODUCTION

Handwriting recognition (or HWR) is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. The main aim of this project is to design expert system for , **“HCR using Neural Network”** that can effectively recognize a particular character of type format using the Artificial Neural Network approach.

Devanagari is part of the Brahmic family of scripts of Nepal, India, Tibet, and South-East Asia. The script is used to write Nepali, Hindi, Marathi and similar other languages of South and East Asia. The Nepalese writing system adopted from Devanagari script consists of 12 vowels, 36 base forms of a consonant, 10 numeral characters and some special characters. Devanagari is used to write in India and Nepal.

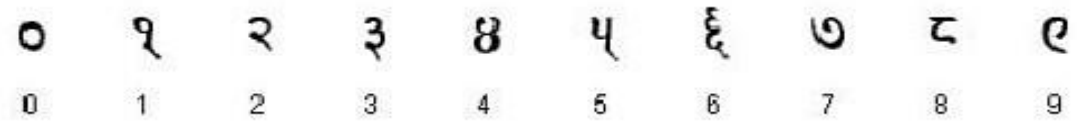


Fig 1.1 Devanagari – Numerals

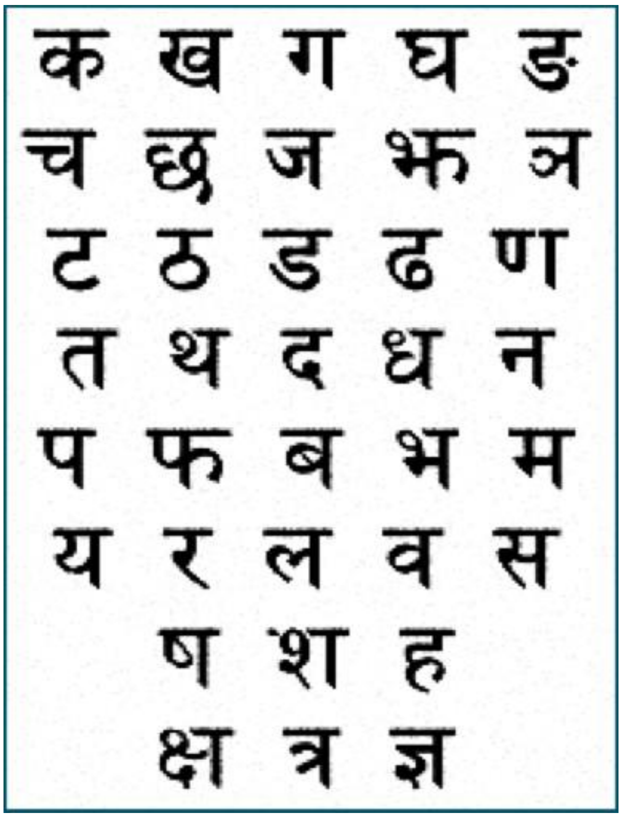

क ख ग घ ङ
च छ ज झ ञ
ट ठ ड ढ ण
त थ द ध न
प फ ब भ म
य र ल व स
ष श ह
क्ष त्र ज्ञ

Fig 1.2 Devanagari Consonants

1.1 PROJECT DEFINITION

This application is useful for recognizing all character(Devanagari) drawn on canvas. Once character is drawn on the canvas area using mouse or touchpad it is given to proposed system , then it will recognize input character. Recognition and classification of characters are done by Neural Network. The main aim of this project is to effectively recognize a particular character of type format using the Artificial Neural Network approach.
The program supported with the usage of Graphical User Interface (GUI).

1.2 RELEVANT THEORY

1.2.1 Benefits of Character Recognition :

1. The idea of Neural Network in HCR will brings us the reading of various combined style of writing a character.

2. In forensic application HCR will be an effective method for evidence collection.

3. Our method develop accuracy in recognizing character in divert font and size.

4. More set of sample invites more accuracy rate because of heavy training and testing session.

1.2.2 Implementation of HCR :

HCR works in stages as preprocessing, feature extraction and recognition using neural network. Preprocessing includes series of operations to be carried out on image, Each image is a 32 * 32 grayscale image which is to be converted into array and then flattened a stored in an image matrix to train the model.. Finally feature vector is presented to the selected algorithm for recognition. Here this extracted features are provided to NN for recognition of character.

1.3 SCOPE

1. System will be designed in way to ensure that offline Handwritten Recognition of Devanagari characters.
2. Use of Neural Network for classification .
3. Large number of training data set will improve the efficiency of the suggested approach.

1.4 DOMAIN INFORMATION

This project is developed using Keras , a high level API for deep learning, Tensorflow , an open source deep learning library, all these libraries are available in their python bindings. Python programming language is used for development. Jupyter Notebook is used for better interactive visualization and block wise running of python codes during development.

1.4.1. TensorFlow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms, and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, It comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains. It provides a variety of different toolkits that allow you to construct models at your preferred level of abstraction. We can use lower-level APIs to build models by defining a series of mathematical operations. Alternatively, we can use higher-level APIs to specify predefined architectures, such as linear regression or neural networks.

TensorFlow consists of the following two components a graph protocol buffer and runtime that executes the distributed graph. These two components are analogous to Python code and the Python interpreter. Just as the Python interpreter is implemented on multiple hardware platforms to run Python code, TensorFlow can run the graph on multiple hardware platforms, including CPU, GPU, and (Tensor Processing Unit).TPU is an accelerated integrated circuit developed by google specifically for processing Tensors .for this project Tensorflow is used as the backend for creating CNN model.

1.4.2 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of Tensorflow, CNTK or Theano. It was developed with a focus on enabling fast experimentation. It was developed to make implementing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license. In this project Keras is used as the abstraction layer running on top of TensorFlow. The CNN model creation can be done in Keras with a few lines of code and is readable also.

1.4.3 Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. All these aforementioned libraries are available in their python libraries which this project is made use of.

1.5 PROBLEM STATEMENT

Here Our goal is to predict the handwritten Devanagari characters & numeral characters in an image. The dataset contains Devanagari Characters. It comprises of 92000 images [32x32 px] corresponding to 46 characters, consonants "ka" to "gya", and the digits 0 to 9. The vowels are missing. The CSV file is of the dimension 92000 * 1025. There are 1024 input features of pixel values in grayscale (0 to 255). The column "character" represents the Devanagari Character Name corresponding to each image. There are 46 classes present out of which 10 classes for digits & 36 classes for characters & each class have 2000 images which can be used to train and test the model.

This is the supervised learning problem more specifically a classification problem. The model should be able to classify which Devanagari number (0 – 9) & consonants is handwritten in the image. The model can be scored for its ability to predict the number correctly over large different test data and real data. In this work, a deep convolutional neural network is applied for handwritten Devanagari characters recognition.

The main contributions of my work can be summarized in the following points:

1) To load necessary Python libraries and read/load the Kaggle Downloaded csv data through pandas.

2) The loaded data will be first explored and visualized using numpy and matplotlib library to understand the nature of the data. Exploring the data will help us in deciding how to approach and whether any preprocessing of the data is needed. Preprocessing of the data is done as required.

3) Once the data is ready, the Deep convolutional neural network(CNN) will be built based on the architecture (ConvNet) as discussed in the Implementation Section. Once the model is built, it will be compiled to check if the architecture has any error.

4) Then the compiled model will be trained on the training data and evaluated using accuracy score against the testing data. Then the results can be analyzed and compared with respect to the benchmark model to know the overall performance of the model.

5) Now we have a trained model, a test is conducted against the model by loading images of Devanagari characters which are not from kaggle downloaded dataset to evaluate the performance of the final model.

6) The images are loaded and then preprocessed to match the Kaggle downloaded dataset format so that we can test it. The model is then made to predict the Devanagari consonant& numeral characters in the preprocessed image.

Thus, now we can evaluate our model's performance over real-time data.

1.6 METHODOLOGY

1.6.1 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 (ANN). The key element of ANN is the novel structure of the information processing system. It is composed of large no. of highly interconnected processing element (neurons) working in union to solve specific problems. In deep learning we don't need to do handcrafted feature extraction like we do in classifiers like SVM and Random Forest. The ANN extracts features by its own. For character classification This project uses Convolutional neural network (CNN) which is a very popular ANN for finding patterns in data.

1.6.1.1 Convolutional Neural Networks (CNN)

Convolution neural network algorithm is a special kind of feed forwarder multilayer perceptron which is basically an Artificial Neural Network (ANN). It has proven its design for identification of patterns from two dimensional data. It has successfully been applying in image classification, natural language processing etc. The main difference between a CNN and an ANN is that CNN uses parameter sharing which makes the computation a lot more easier. A typical CNN Always has one input layer, convolution layers, pooling layers , reLU(Rectified Linear Unit) layers, fully connected layers and one output layer. In addition to this, the number of layer in a CNN is subjected to change according to the classification requirements . a simple CNN is shown in figure 2.1.

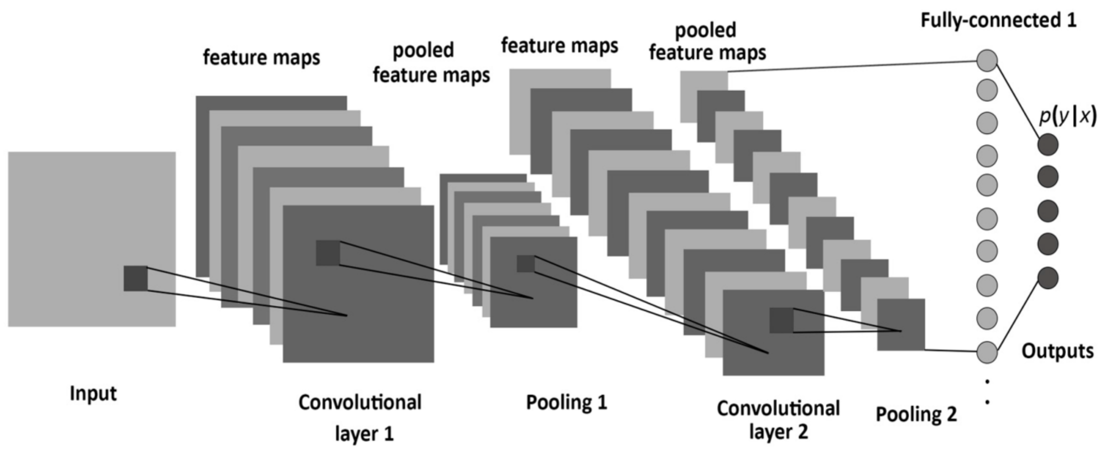


Figure 1.3 Convolutional Neural Network

CNNs use variation of multilayer perceptron designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks. Convolutional layers apply a convolution operation to the input, passing the result to the next layer. Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. This is done for down sampling a feature map obtained as the output of a convolutional layer. The final fully connected layers connects every neuron in the previous layer to the next layer. So the final downsampled featuremap is stretched into a single vector and is given as input to the fully connected layer. The output of the final fully connected layer contains the required number of classes.

1.7 METRICS

Like any other n-class image classification problem, the metric of central importance for us is accuracy. The final goal is to improve accuracy on the validation set.

For our refined DNN model, we compiled model with 'categorical_crossentropy' as loss function, 'adam'(Adaptive Moment Estimation) as optimizer and 'accuracy' as metrics. We achieved accuracy on testing data set is 97.38%.

Accuracy (formula defined below) is the most relevent metric for us because, it tells us how accurately the model performs on the unseen data.
Accuracy is the proportion of samples predicted correctly among the total number of samples examined. For Example, in our problem, it is the ratio of the handwritten Devanagari characters predicted correctly to the total number of handwritten Devanagari characters evaluated. It measures the correctness of a model.

$$\textbf{Accuracy} = \frac{\textbf{Number of samples predicted correctly}}{\textbf{Total number of samples examined}}$$

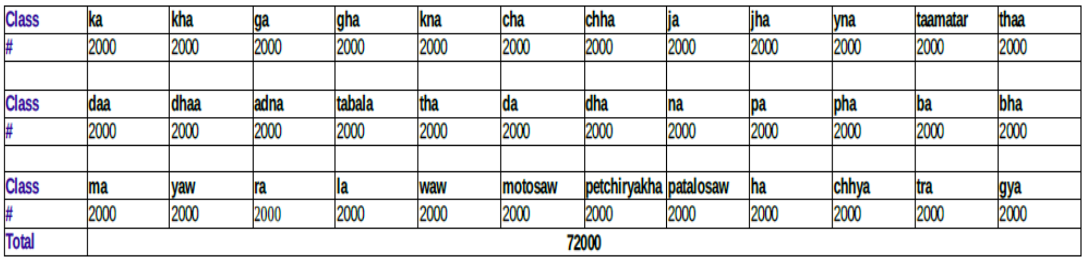
1.8 DATA EXPLORATION

Devanagari handwritten character dataset is created by collecting the variety of handwritten Devanagari characters from different individuals from diverse fields. Handwritten documents are then scanned and cropped manually for individual characters. Each character sample is 32x32 pixels and the actual character is centered within 28x28 pixels.

Padding of 0 valued 2 pixels is done on all four side to make this increment in image size. The images were applied grayscale conversion. After this, the intensity of the images was inverted making the character white on the dark background. To make uniformity in the background for all the images, we suppressed the background to 0 value pixel. Each image is a gray-scale image having background value as 0.

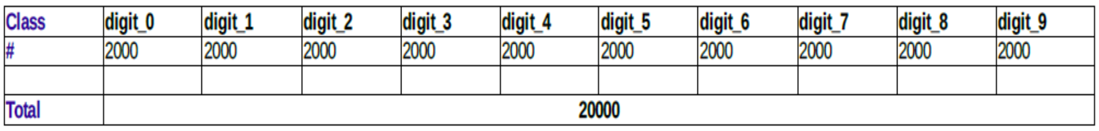
Devanagari Handwritten Character Dataset contains a total of 92,000 images with 72,000 images in consonant datasets and 20,000 images in the numeral dataset. Handwritten Devanagari consonant character dataset statistics are shown in Table I and handwritten Devanagari numeral character dataset statistics is shown in Table II.

Table I - Consonants Character Dataset



Class	ka	kha	ga	gha	kna	cha	chha	ja	jha	yna	taamatar	thaa
#	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
Class	daa	dhaa	adna	tabala	tha	da	dha	na	pa	pha	ba	bha
#	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
Class	ma	yaw	ra	la	waw	motosaw	petchiryakha	patalosaw	ha	chhya	tra	gya
#	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
Total	72000											

Table II - Numeral Dataset



Class	digit_0	digit_1	digit_2	digit_3	digit_4	digit_5	digit_6	digit_7	digit_8	digit_9
#	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
Total	20000									

1.9 EXPLORATORY VISUALIZATION

Let’s visualize all image of the Devanagari consonants & numeral characters in the training set of downloaded kaggle dataset.
Each character sample is 32x32 pixels and the actual character is centered within 28x28 pixels.

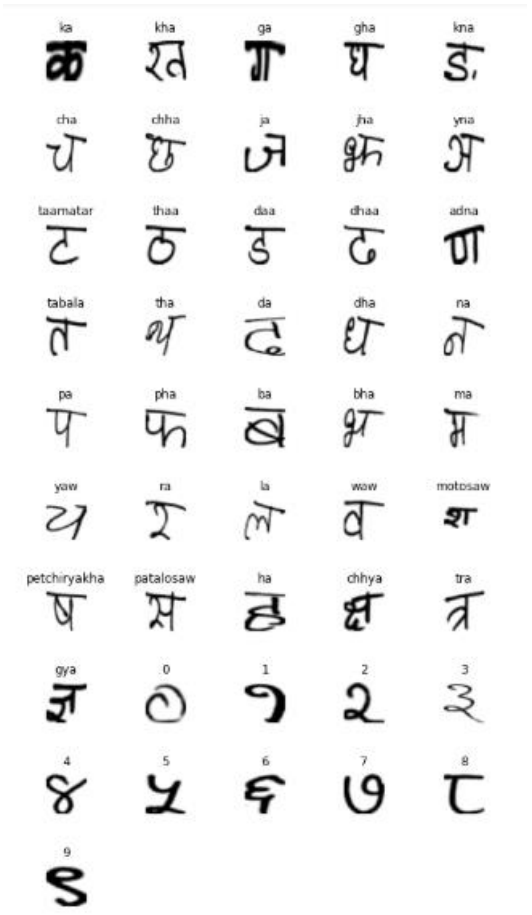


Fig 1.4 Visualization of all Devanagari Consonants & Numerals Character in the dataset

Let us verify the pixel distribution of any random character using matplotlib.

Plotting a histogram can give you a quick visualization of your data distribution. It is important to select the correct 'bin' size (groups of data) to get the best curve approximation.

This plot will show you if your data values are centered (normally distributed), skewed to oneside or the other, or have more than one 'mode' - localized distribution concentrations.

They can also be rearranged as a Pareto Plot from highest frequency to lowest, allowing you to focus on the most important factors to address in problem solutions.

One of the more common is to decide what value of threshold to use when converting a grayscale image to a binary one by thresholding, If the image is suitable for thresholding then the histogram will be bi-modal --- i.e. the pixel intensities will be clustered around two well-separated values.

This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. For an 8-bit grayscale image there are 256 different possible intensities, and so the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale values.

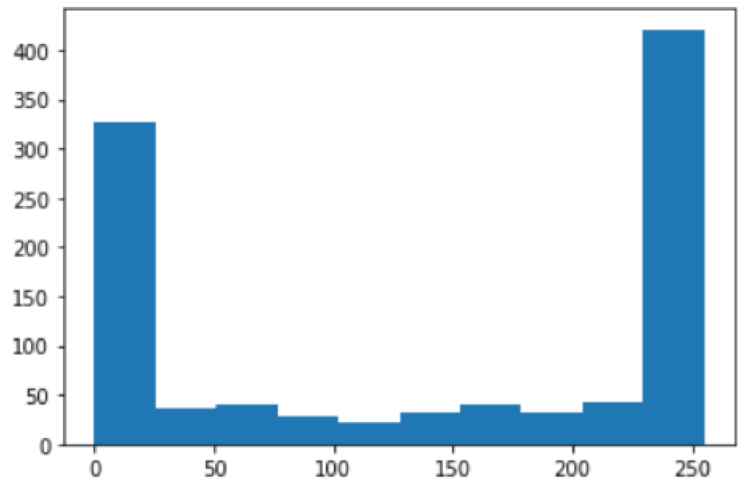


Fig. 1.5 Pixel Distribution

The above histogram shows the number of pixels for every pixel value, i.e from 0 to 255.
In this way this visualization will help us in data preprocessing.

Chapter 2

LITERATURE SURVEY

2.1 Survey on Offline Recognition of Handwritten Devanagari Script

This paper uses HMM for feature extraction of Devanagari characters and follows a image segmentation approach where the text is extracted from the image after segmenting the image to read individual characters. K Nearest Neighbors (KNN) SVM, MLP are used for classification of the handwritten images. Thus, the paper tries to achieve automatic recognition of handwritten Devanagari Script by using various algorithms.

2.2 Handwritten Devanagari Script Recognition: A Survey

This paper gives an insight of the character recognition and surveys the various research work done in this field. The paper discusses the method for character recognition and discusses the various applications of handwritten Devanagari character recognition systems.

2.3 Diagonal Based Feature Extraction for Handwritten Alphabets Recognition System using Neural Network

In this multilayer feed forward, neural network is used for handwritten character recognition. the paper introduces a new method, called, diagonal based feature extraction for extracting the features of the character images. Fifty data sets, each containing 26 alphabets written by various people, are used for training the neural network and 570 different handwritten alphabetical characters are used for testing.

2.4 Devanagari Character Recognition Using Neural Networks

Neural network approach is proposed to build an automatic offline character recognition system. Devanagari is an Indo-Aryan language spoken by about 71 million people mainly in the Indian state of Maharashtra and neighboring states. One may find so much work for Indian languages like Hindi, kanada, Tamil, Bangala, Malayalam etc but devanagari is a language for which hardly any work is traceable especially for character recognition. In this paper, work has been performed to recognize Devanagari characters using multilayer perceptron with hidden layer. Various patterns of characters are created in the matrix (n*n) with the use of binary form and stored in the file. We have used the back propagation neural net- work for efficient recognition and rectified neuron values were transmitted by feed forward method in the neural network.

2.5 Handwritten Devanagari Character Recognition using Neural Net- work

In this digital era, most important thing is to deal with digital documents, organizations using handwritten documents for storing their information can use handwritten character recognition to convert this information into digital. Handwritten Devanagari characters are more difficult for recognition due to presence of header line, conjunct characters and similarity in shapes of multiple characters. This paper deals with development of grid based method which is combination of image centroid zone and zone centroid zone of individual character or numerical image. In feature extraction using grid or zone based approach individual character or numerical image is divided into n equal sized grids or zones then average distance of all pixels with respect to image centroid or grid centroid is computed. In combination of image centroid and zone centroid approach it computes average distance of all pixels present in each grid with respect to image centroid as well as zone centroid which gives feature vector of size 2xn features. This feature vector is presented to feed forward neural network for recognition. Complete process of Devanagari character recognition works in stages as document preprocessing, segmentation, feature extraction using grid based approach followed by recognition using feed forward neural network.

Chapter 3

DESIGN

3.1 ALGORITHMS AND TECHNIQUES

Given that the problem is a supervised learning problem and more specifically a classification problem, there are lot of algorithms available for training a classifier to learn from the data. The algorithm chosen for this project is Deep Neural Network(DNN) using Convolutional Neural Network(ConvNet).

3.1.1 Convolutional Neural Networks

Convolutional Neural Network (CNN or ConvNet) is a biologically-inspired trainable machine leaning architecture that can learn from experiences like standard multilayer neural networks.

ConvNets consist of multiple layers of overlapped tiling collections of small neurons to achieve better representation of the original image. ConvNets are widely used for image recognition. A CNN consists of a lot of layers. These layers when used repeatedly, leading to a formation of a Deep Neural Network. Three main types of layers used to build a CNN are:

1. **First Convolution Layer:**
   The convolution layer is the core building block of a convolutional neural network. It convolves the input image with a set of learnable filters or weights, each producing one feature map in the output image.

2. **ReLU Function:**
   The Rectified Linear Unit apply an elementwise activation function, such as the max (0, x) thresholding at zero.

3. **Pooling Layer:**
   The pooling layer is used to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The pooling layer takes small rectangular blocks from the convolution layer and subsamples it to produce a single output from that block. There are several ways to do this pooling, such as taking the average or the maximum, or a learned linear combination of the neurons in the block.

4. **Fully Connected Layer:**

   The fully-connected layer is used for the high-level reasoning in the neural network. It takes all neurons in the previous layer and connects it to every single neuron it has. Their activations can be computed with a matrix multiplication followed by a bias offset as a standard neural networks.

An example of a Convolutional Neural Network is given below

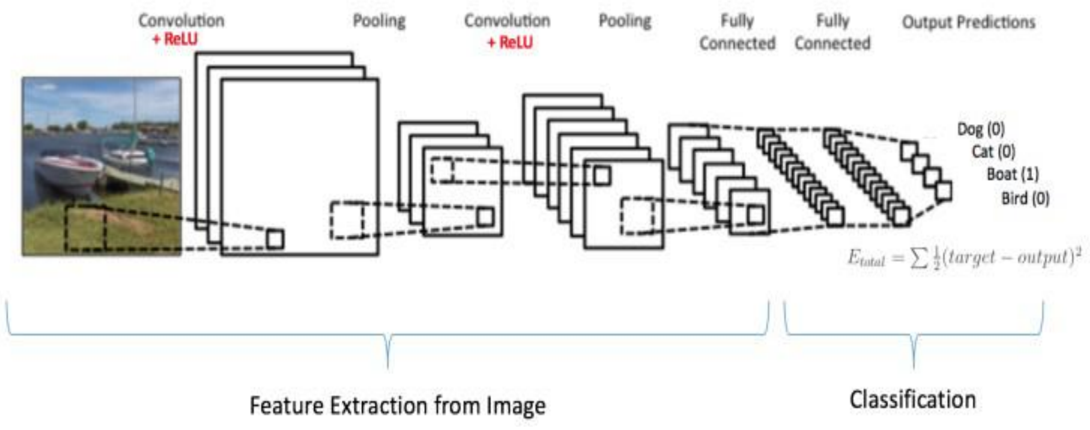


Fig 3.1 CNN Example

3.2 BENCHMARK

The benchmark model solves the same handwritten devanagari Consonants / Numeral characters recognition problem mentioned in this project by using CNN model.

The CNN has complexity in computations, it offers several advantages in pattern recognition and classification in the similar manner close to human intelligence to a small extent.

In this case, we compiled our initial CNN model with 'categorical_crossentropy' as loss function, 'sgd' (Stochastic Gradient Descent) as optimizer and 'accuracy' as metrics & achieved a test set accuracy of 80%. After that we added 2 dropout layer in our redefined model to achieve more accuracy on test set data than initial model. We have described each model architect clearly in implementation section. For Redeined CNN model, we compiled it with 'categorical_crossentropy' as loss function, 'adam' (Adaptive Moment Estimation) as optimizer and 'accuracy' as metrics. We achieved a test set accuracy of 97.38%. We used, Adam Optimizer because it achieves good results fast on large models and datasets.

3.3 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. Requirement analysis allows the software engineer (sometime called Analyst or Modeler in this role) to elaborate on basis requirements during earlier requirement engineering task and build models that depict user scenarios, functional activities , problem classes and their relationship s, system and class behavior and the flow of data as it is transformed.

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.

3.3.1 Requirement Specification

3.3.1.1 Functional Requirements

The functional requirements identified in this project are

1. The system should be able to detect Devanagari handwritten characters, to do the character classification task a deep learning model should be trained with dataset of Devanagari characters.
2. The system should enable the user to input an image containing Devanagari handwritten characters. For that a GUI is created.
3. System must provide accuracy for character recognition.

3.3.1.2 Non Functional Requirements

3.3.1.2.1 Performance Requirements

The main performance requirements that the product should satisfy are:

1. Accuracy**:** Accuracy in prediction of handwritten characters of different styles and slants.
2. Proper training**:** The training set used for training should be a wise choice of images. It should be able to cover all the possible shape of a particular character.
3. Algorithms **:** The algorithms used for different stages should be efficient for proper result.

3.3.1.2.2 Quality Requirements

The most important quality requirements that the system should satisfy are:

1. Scalability **:** The software will meet all of the functional requirements without an unexpected behavior.

2. Maintainability **:** The system should be maintainable. It should keep backups to atone for system failures, and should log its activities periodically.

3. Reliability **:** The acceptable threshold for down-time should be long as possible. i.e. mean time between failures should be large as possible. And if the system is broken, time required to get the system back up again should be minimum.

4. Testability **:** The proposed system should be properly tested under various light conditions and quality conditions of the input image to ensure that system performs well in all the circumstances.

.

3.4 SYSTEM SPECIFICATION

3.4.1. Hardware Requirement

The minimum hardware requirement of computer is

1. Processor : Intel i7 Processor
2. Storage : 10 GB Hard Disk space
3. Memory : 8 GB RAM
4. GPU : Nvidia Quadro K2200

3.4.2. Software Requirement

The minimum software requirement of computer is

1. Operating system : Windows
2. Language : Python
3. Frameworks : Keras, TensorFlow
4. Editors : Jupyter Notebook
5. Toolkit : Used TkInter to create the GUI and Pyinstaller to convert it into an executable.

Chapter 4

IMPLEMENTATION

4.1 DATA PREPROCESSING

Yes, downloaded dataset needs data preprocessing in this project. In previous sections, I have shown all different classes in dataset, character 'ka' to 'gya' & digit '0' to '9'. All these categorical features have string values.
Sklearn provides a very efficient tool for encoding the levels of a categorical features into numeric values. LabelEncoder encode labels with value between 0 and n_classes-1.
We can encode all the categorical features by using below code le = LabelEncoder()
After all categorical features are get encoded, I have reshaped trained & test data images into 32x32.

4.2 MODULES DESCRIPTION

In this project implementation process divided into two steps:

1. Build Model, Model training and evaluating stage.
2. Model testing on real data.

4.2.1 Build Model, Model training and evaluating stage

In the first stage, the model was trained on the preprocessed training data and evaluated against the test data. The following steps are done during the first stage:

1. Build Deep Neural Network (DNN) model with Keras.

2. Load both the training and testing data into memory and preprocess them as described in the above section.

3. Define the initial network architecture and training parameters as shown in the code and block diagram below in Fig. 6 & Fig. 7.

4. Define the loss function with metrics as Accuracy.

5. The network is trained on the training data and evaluated against the test data.

6. Note down the loss & accuracy score as well as accuracy curves

7. If the accuracy is not high enough, repeat from step 2 by architecture of the network &by changing the hyper parameters value and typing different loss function.

In the first stage, step 2,3 was difficult implementing since finding the optimum architecture, hyperparamenters and loss function was time consuming because a single run takes 30-40 mins.

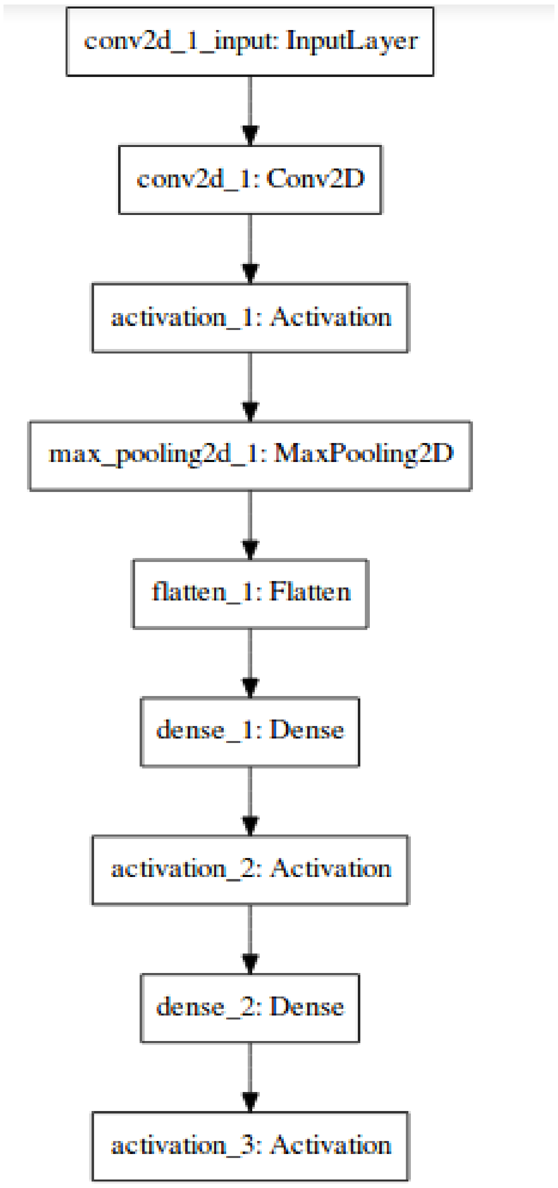
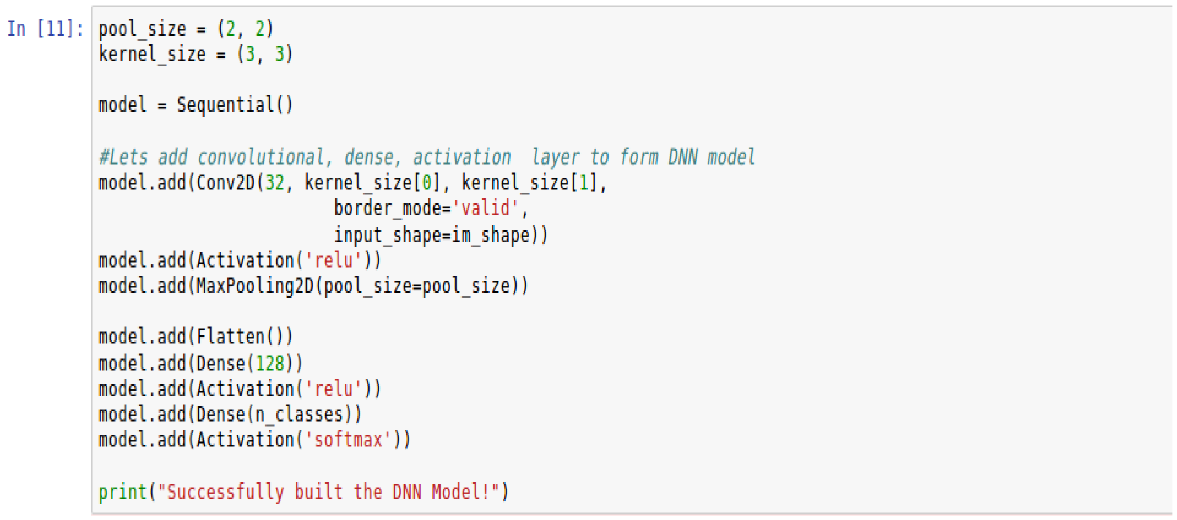


Fig. 4.1 Initial CNN Model Block Diagram



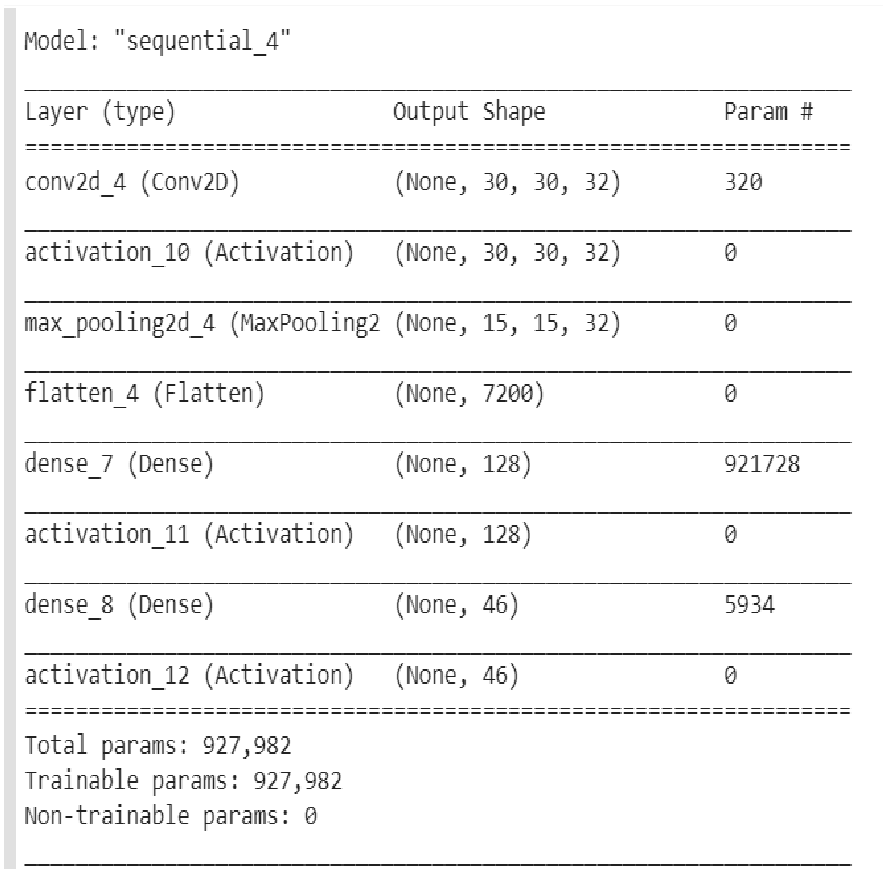
```
In [11]: pool_size = (2, 2)
         kernel_size = (3, 3)

         model = Sequential()

         #Lets add convolutional, dense, activation  layer to form DNN model
         model.add(Conv2D(32, kernel_size[0], kernel_size[1],
                         border_mode='valid',
                         input_shape=im_shape))
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=pool_size))

         model.add(Flatten())
         model.add(Dense(128))
         model.add(Activation('relu'))
         model.add(Dense(n_classes))
         model.add(Activation('softmax'))

         print("Successfully built the DNN Model!")
```

Fig. 4.2 Initial CNN Model Built Code

4.2.1.1 Initial CNN Model Description

The first layer of the model is the Input Layer which gets the input data as arrays during the training phase. The next layer is the convolutional layer with 32 filters and 3,3 kernel size which convolutes the input image. The next layer is an activation Layer made of Rectified Linear Unit (ReLU) which acts as an elementwise activation function.

The next layer is the max pooling layer where down sampling is carried out. Then a flatten layer is used to flatten the data. Then A dense or fully connected layer of size 128 is applied followed by an activation layer. The next layer is fully connected layer of size 10 i.e. it is the output size followed by a softmax activation layer. The output of the final activation layer is the final output. Once the model is built using the above architecture, it is compiled by specifying the loss function as categorical cross entropy, optimizer as SGD(Stochastic Gradient Descent), and metrics as accuracy. Once the modelis complied with no errors, it is trained over the training dataset x_train and y_train. The model is then evaluated against thetest dataset x_test and y_test. The accuracy score of the initial model is measured to be 80%.



```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 30, 30, 32)        320
_________________________________________________________________
activation_10 (Activation)   (None, 30, 30, 32)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 7200)              0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               921728
_________________________________________________________________
activation_11 (Activation)   (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 46)                5934
_________________________________________________________________
activation_12 (Activation)   (None, 46)                0
=================================================================
Total params: 927,982
Trainable params: 927,982
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4.3 Initial CNN Model

4.2.2 Model testing on real data

During the second stage, the final model is chosen and is tested against real data rather than the downloaded kaggle dataset itself.

The following steps are involved in this stage:

1. Use the Canvas Area to Draw the character using mouse or touch

2. Click on Run to see the predictions

3. Use Clear Button to Clear Canvas

4.3 REFINEMENT

The initial model is improved by using the following technique

1. It is made further deep by adding two more convolutional layer and one more extra dense (fully connected) layer.

2. Two dropout layer is added to ensure that overfitting does not happen.

3. Using Adam optimizer instead of SGD(Stochastic Gradient Descent) to update the weights efficiently.

4. Fig. 8 & 9 illustrate the whole refine CNN model.

Once the model is built using the above architecture, it is compiled by specifying the loss function as categorical cross entropy, optimizer as Adam, and metrics as accuracy. Once the model is complied with no errors, it is trained over the training dataset x_train and y_train. The model is then evaluated against the test dataset x_test and y_test. The accuracy score for refined model is measured to be 97.38%.

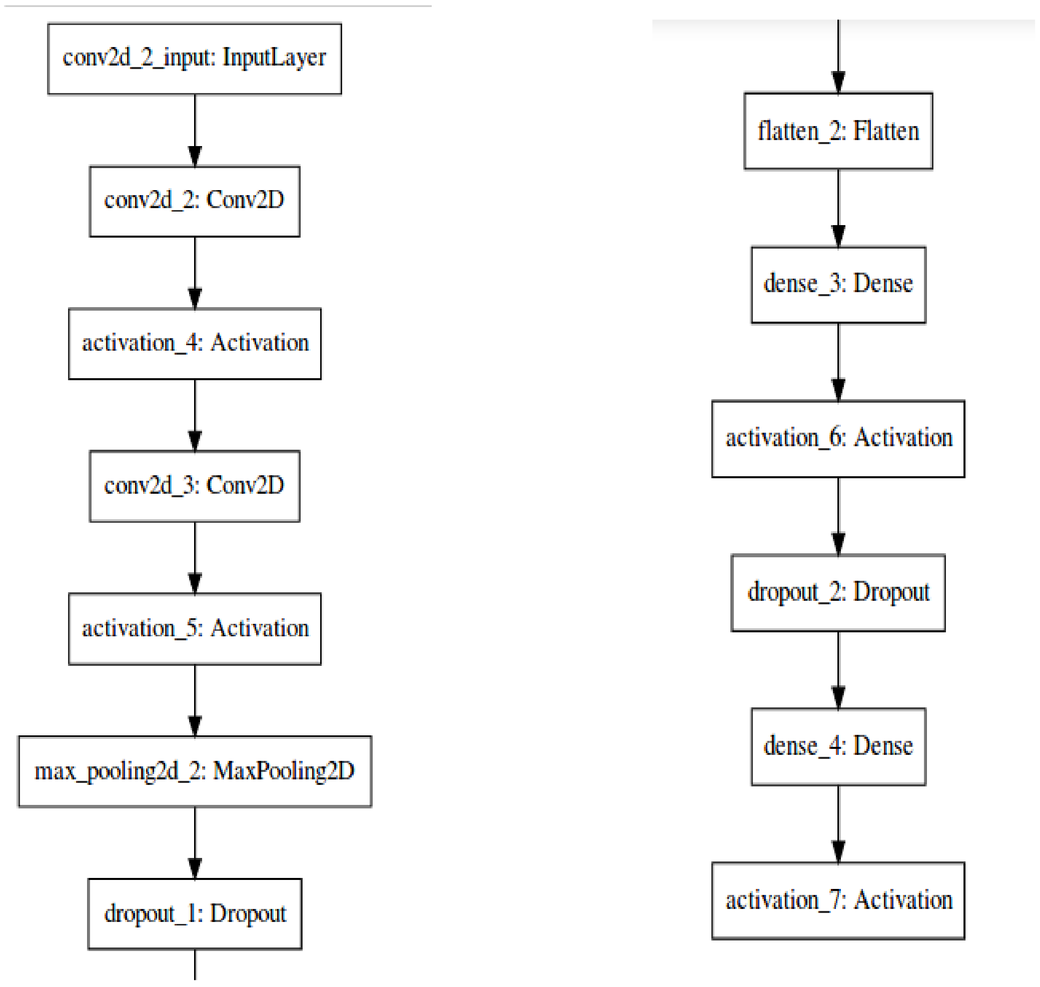
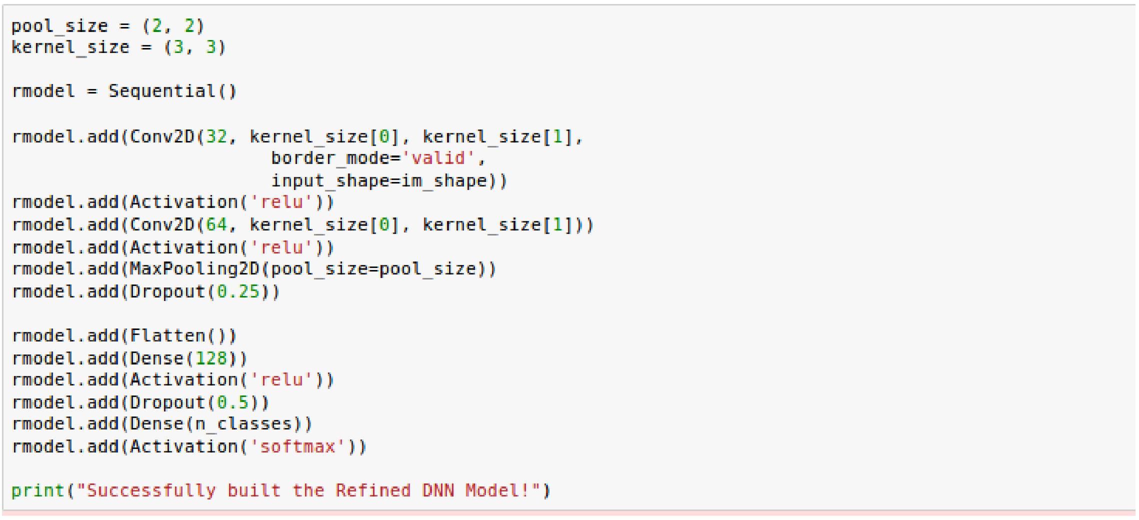
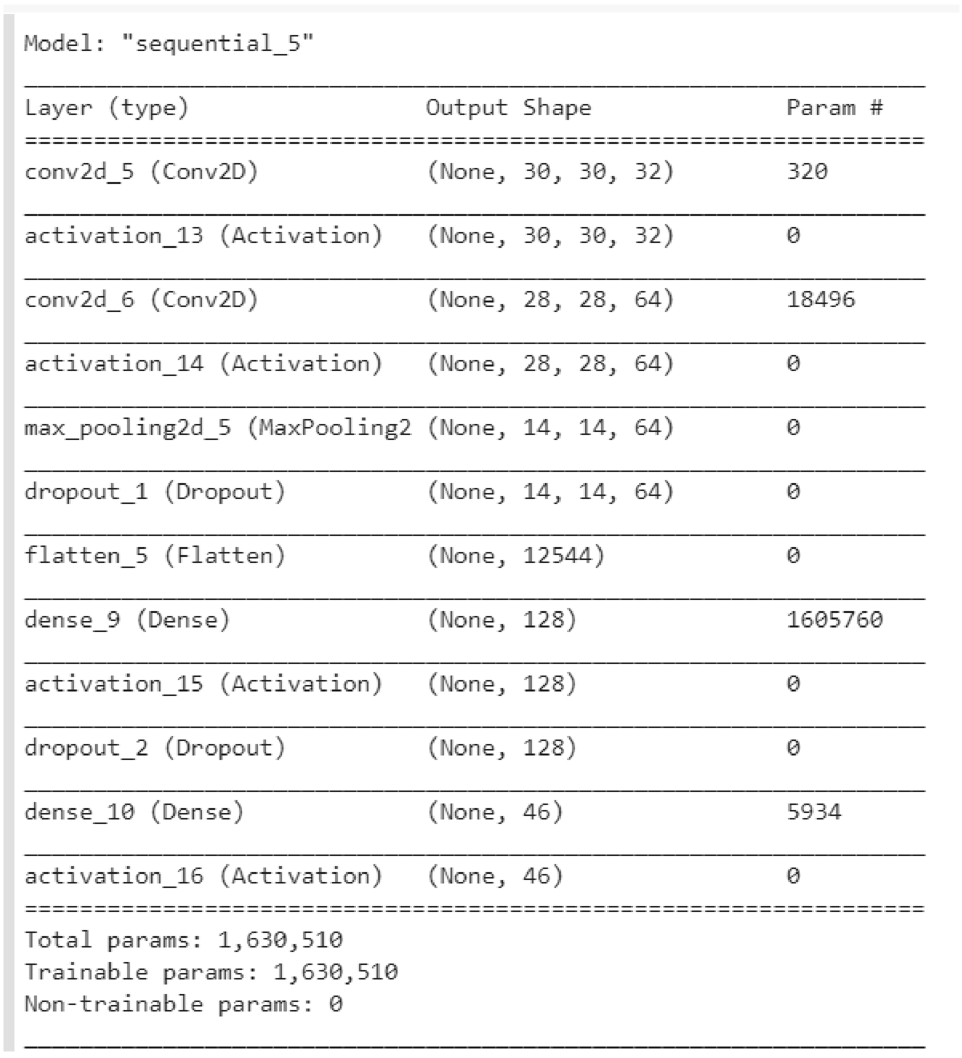


Fig. 4.4 Refined CNN model block diagram



```
pool_size = (2, 2)
kernel_size = (3, 3)

rmodel = Sequential()

rmodel.add(Conv2D(32, kernel_size[0], kernel_size[1],
                        border_mode='valid',
                        input_shape=im_shape))
rmodel.add(Activation('relu'))
rmodel.add(Conv2D(64, kernel_size[0], kernel_size[1]))
rmodel.add(Activation('relu'))
rmodel.add(MaxPooling2D(pool_size=pool_size))
rmodel.add(Dropout(0.25))

rmodel.add(Flatten())
rmodel.add(Dense(128))
rmodel.add(Activation('relu'))
rmodel.add(Dropout(0.5))
rmodel.add(Dense(n_classes))
rmodel.add(Activation('softmax'))

print("Successfully built the Refined DNN Model!")
```

Fig. 4.5 Refined CNN model code



```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 30, 30, 32)        320
_________________________________________________________________
activation_13 (Activation)   (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
activation_14 (Activation)   (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
flatten_5 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               1605760
_________________________________________________________________
activation_15 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 46)                5934
_________________________________________________________________
activation_16 (Activation)   (None, 46)                0
=================================================================
Total params: 1,630,510
Trainable params: 1,630,510
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4.6 Final CNN Model

4.4 PYTHON GUI – tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

There are two main methods used which the user needs to remember while creating the Python application with GUI.

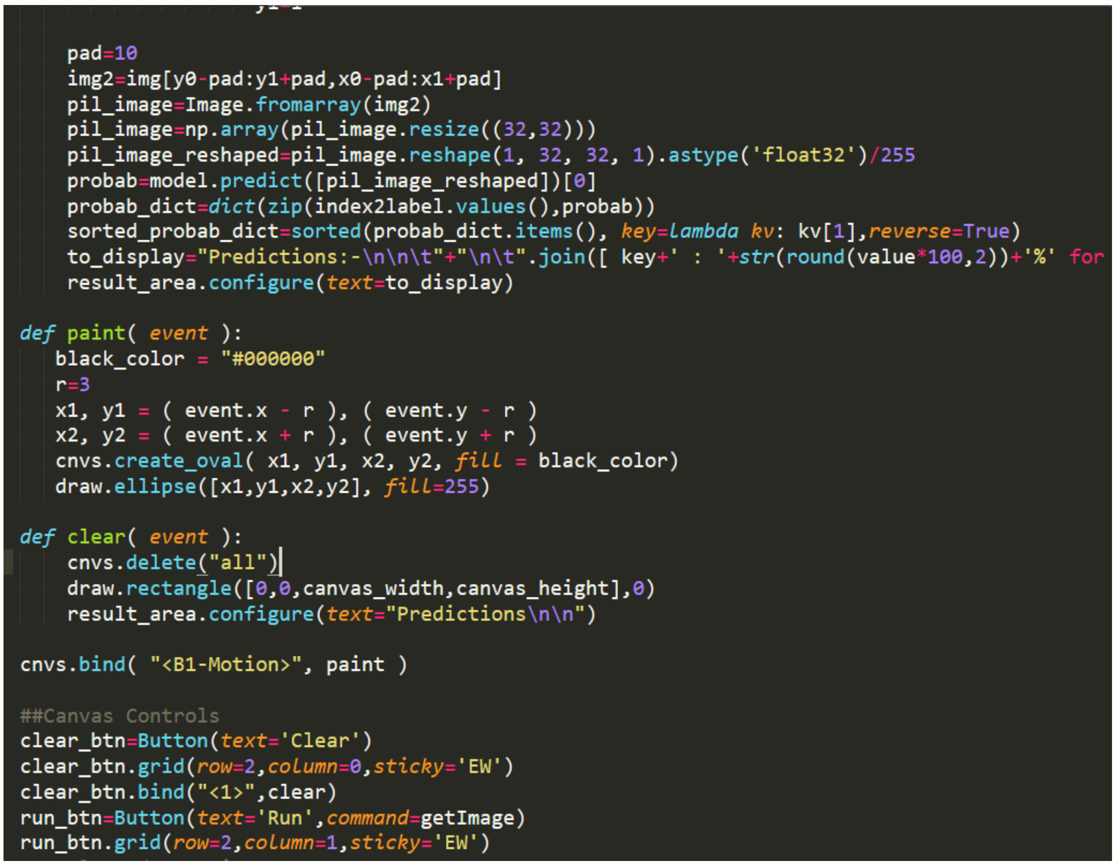
1) **Tk(screenName=None, baseName=None, className='Tk', useTk=1):**
   To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

   m = tkinter.TK( ) where m is the name of the main window object

2) **mainloop():** There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

   m.mainloop( )

Driver Class / Application Class :



```
    pad=10
    img2=img[y0-pad:y1+pad,x0-pad:x1+pad]
    pil_image=Image.fromarray(img2)
    pil_image=np.array(pil_image.resize((32,32)))
    pil_image_reshaped=pil_image.reshape(1, 32, 32, 1).astype('float32')/255
    probab=model.predict([pil_image_reshaped])[0]
    probab_dict=dict(zip(index2label.values(),probab))
    sorted_probab_dict=sorted(probab_dict.items(), key=lambda kv: kv[1],reverse=True)
    to_display="Predictions:-\n\n\t"+"\n\t".join([ key+' : '+str(round(value*100,2))+'%' for
    result_area.configure(text=to_display)

def paint( event ):
   black_color = "#000000"
   r=3
   x1, y1 = ( event.x - r ), ( event.y - r )
   x2, y2 = ( event.x + r ), ( event.y + r )
   cnvs.create_oval( x1, y1, x2, y2, fill = black_color)
   draw.ellipse([x1,y1,x2,y2], fill=255)

def clear( event ):
    cnvs.delete("all")
    draw.rectangle([0,0,canvas_width,canvas_height],0)
    result_area.configure(text="Predictions\n\n")

cnvs.bind( "<B1-Motion>", paint )

##Canvas Controls
clear_btn=Button(text='Clear')
clear_btn.grid(row=2,column=0,sticky='EW')
clear_btn.bind("<1>",clear)
run_btn=Button(text='Run',command=getImage)
run_btn.grid(row=2,column=1,sticky='EW')
```

Fig. 4.7

Chapter 5

Results

5.1 MODEL EVALUATION & VALIDATION

The model is evaluated against the test set (x_test , y_Test). The final architecture and hyperparameter are chosen because they performed best out of the other models tried. The robustness of the final model is verified by conducting a test against the final model using different images of Devanagari Consonants & Numerals character other than Downloaded kaggle dataset itself.

The following observations are based on the results of the test:

1. The final model has predicted the Devanagari characters of 7 out 8 correctly.

2. Our model had a problem in predicting Devanagari Consonants & Numerals that have thin stokes like the consonant character 'yna' with thin stroke in the test cases The observations of test confirms that our model is reliable and robust enough to perform well on real data.

Use of dropout layer in the final model ensures that small changes in training data or the input space do not affect the results greatly.

5.2 JUSTIFICATION

In order to justify that our model has potentially solved the problem, we need to look at the results of the real-time test conducted against model. Our model can correctly predict 7 out of 8 tests. Thus, our model solved the problem with a good accuracy score and reliability in real-time applications.

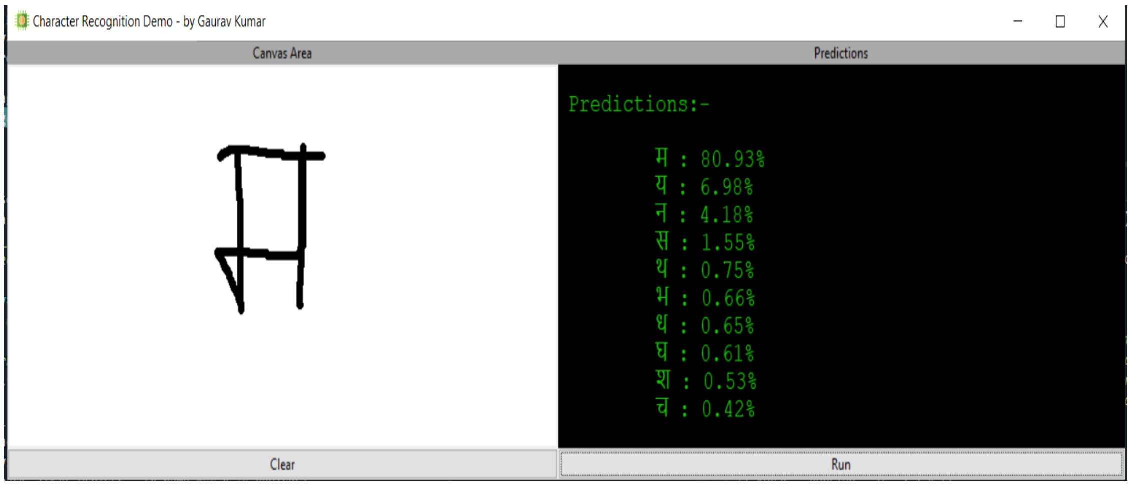
Lets revisit our performance summary.

- Test data set Accuracy of 80% of initial model.
- Test data set Accuracy of 97.38% of refined model.
- The final model has predicted the Devanagari characters(not from the dataset) of 13 out 14 images correctly. Thus our model has performed well enough.

We have achieved an accuracy of 97.38% on test set. While it is ~6% away from human performance, it should be considered good because, It performed extremely well on unseen images (not a part of original dataset).

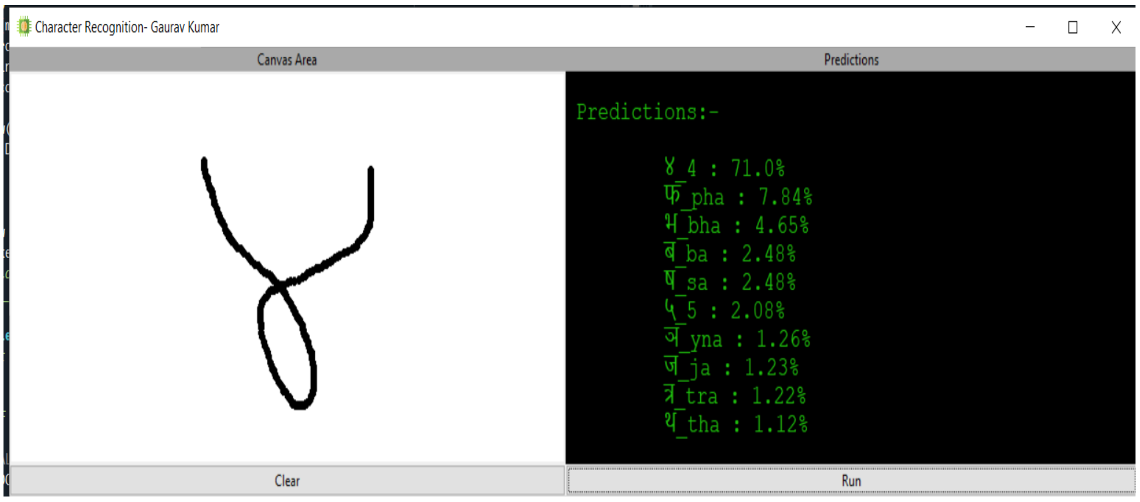
Application Class:

Giving input to machine using using mouse or touchpad, Click on Run to see the predictions and Use Clear Button to Clear Canvas.



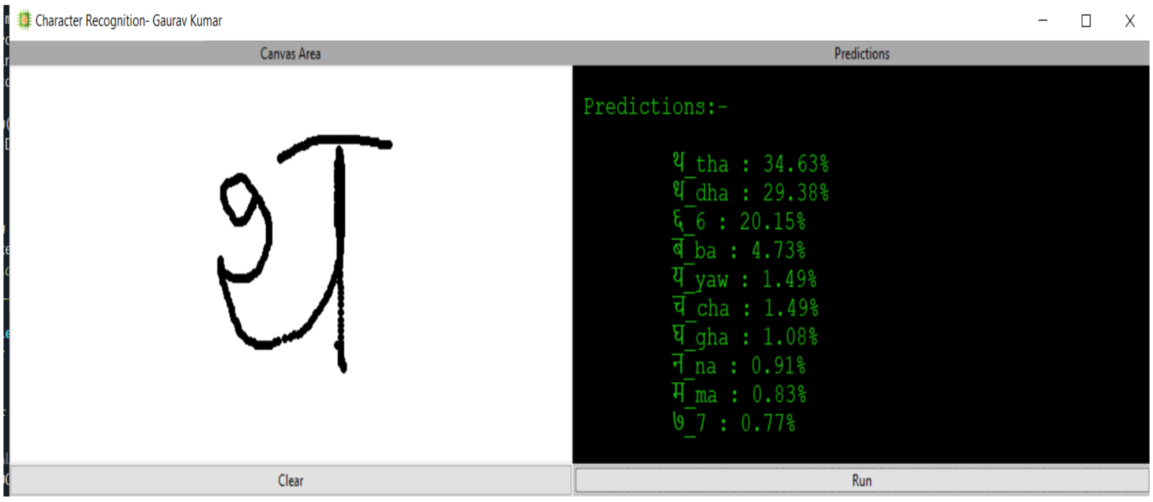
```
Predictions:-

        म : 80.93%
        य : 6.98%
        न : 4.18%
        स : 1.55%
        थ : 0.75%
        भ : 0.66%
        ध : 0.65%
        घ : 0.61%
        श : 0.53%
        च : 0.42%
```

Fig 5.1 Character = ma



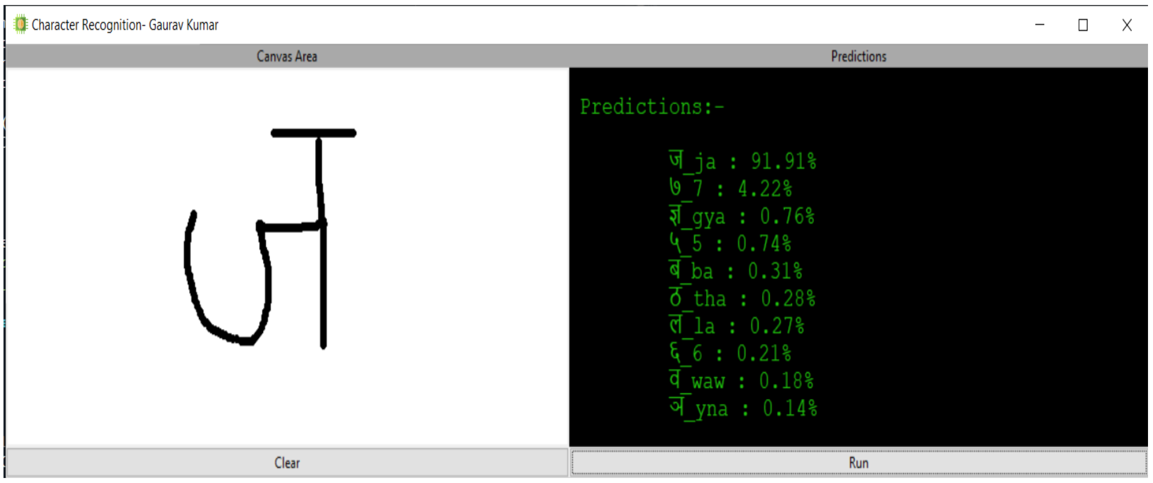
```
Predictions:-

        ४_4 : 71.0%
        फ_pha : 7.84%
        भ_bha : 4.65%
        ब_ba : 2.48%
        ष_sa : 2.48%
        ५_5 : 2.08%
        ञ_yna : 1.26%
        ज_ja : 1.23%
        त्र_tra : 1.22%
        थ_tha : 1.12%
```

Fig 5.2 Character = 4



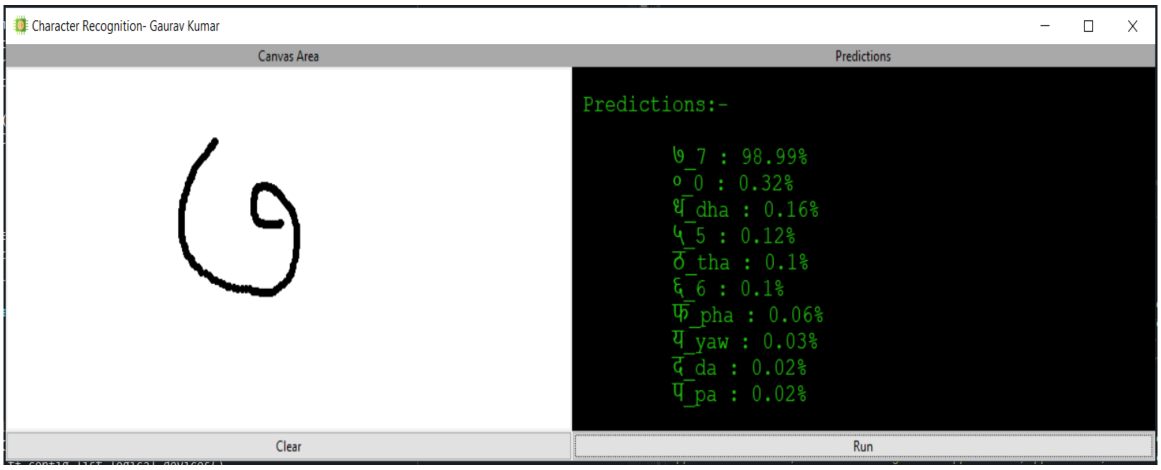
```
Predictions:-

        थ_tha : 34.63%
        ध_dha : 29.38%
        ६_6 : 20.15%
        ब_ba : 4.73%
        य_yaw : 1.49%
        च_cha : 1.49%
        घ_gha : 1.08%
        न_na : 0.91%
        म_ma : 0.83%
        ७_7 : 0.77%
```

Fig 5.3 Character = tha



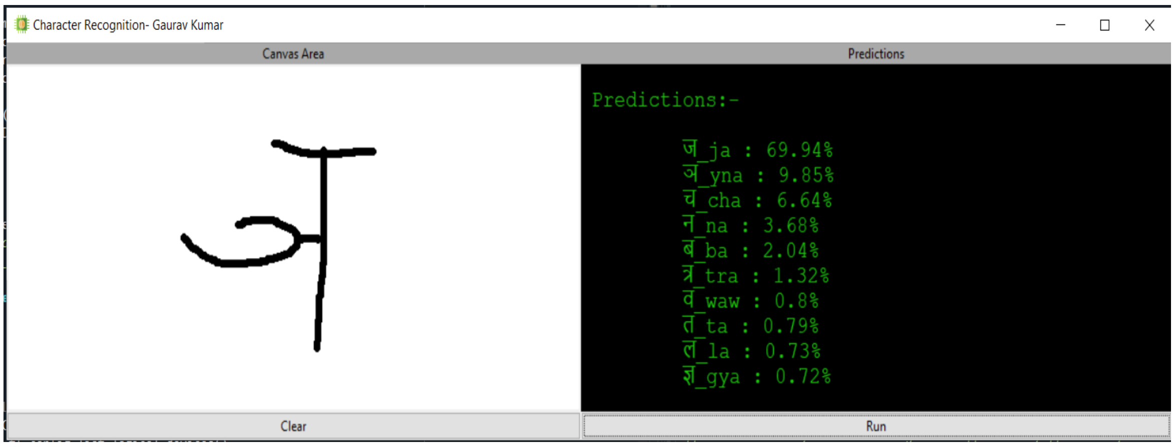
```
Predictions:-

        ज_ja : 91.91%
        ७_7 : 4.22%
        ज्ञ_gya : 0.76%
        ५_5 : 0.74%
        ब_ba : 0.31%
        ठ_tha : 0.28%
        ल_la : 0.27%
        ६_6 : 0.21%
        व_waw : 0.18%
        ञ_yna : 0.14%
```

Fig 5.4 Character = ja



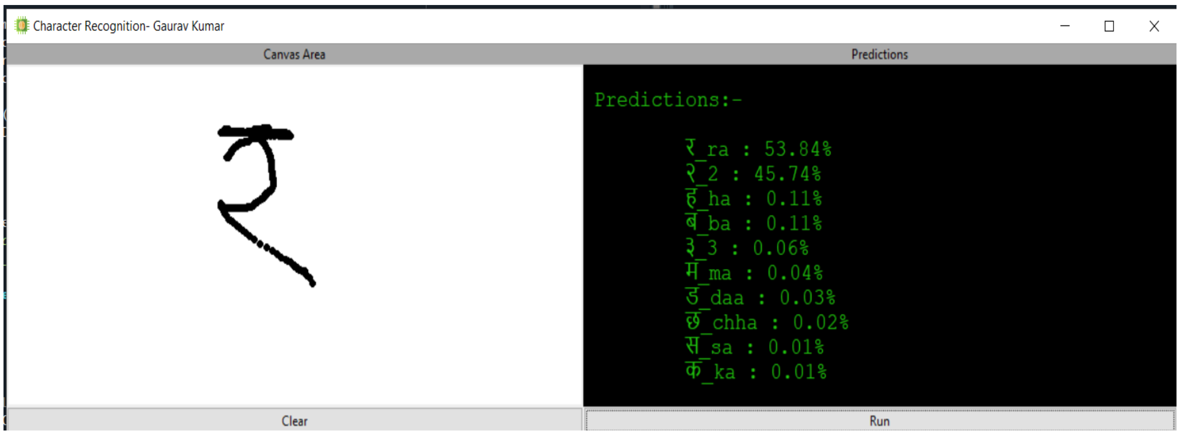
```
Predictions:-

        ७_7 : 98.99%
        ०_0 : 0.32%
        ध_dha : 0.16%
        ५_5 : 0.12%
        ठ_tha : 0.1%
        ६_6 : 0.1%
        फ_pha : 0.06%
        य_yaw : 0.03%
        द_da : 0.02%
        प_pa : 0.02%
```

Fig 5.5 Character = 7



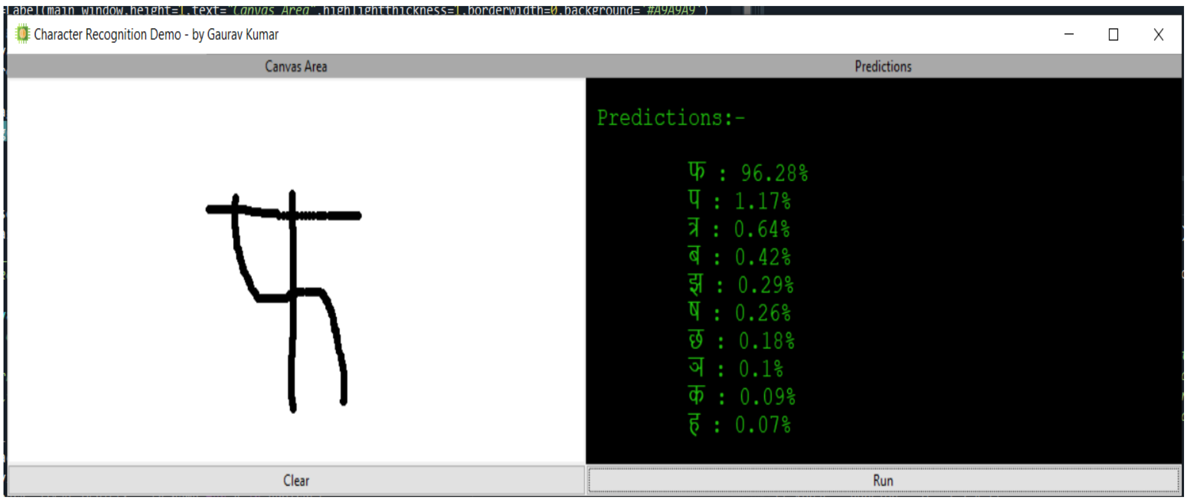
```
Predictions:-

        ज_ja : 69.94%
        ञ_yna : 9.85%
        च_cha : 6.64%
        न_na : 3.68%
        ब_ba : 2.04%
        त्र_tra : 1.32%
        व_waw : 0.8%
        त_ta : 0.79%
        ल_la : 0.73%
        ज्ञ_gya : 0.72%
```

Fig 5.6 Character = yna (Wrongly predicted)



```
Predictions:-

        र_ra : 53.84%
        २_2 : 45.74%
        ह_ha : 0.11%
        ब_ba : 0.11%
        ३_3 : 0.06%
        म_ma : 0.04%
        ड_daa : 0.03%
        छ_chha : 0.02%
        स_sa : 0.01%
        क_ka : 0.01%
```

Fig 5.7 Character = ra



```
label(main_window,height=1,text="Canvas Area",highlightthickness=1,borderwidth=0,background="#A9A9A9")
```

```
Predictions:-

        फ : 96.28%
        प : 1.17%
        त्र : 0.64%
        व : 0.42%
        झ : 0.29%
        ष : 0.26%
        छ : 0.18%
        ञ : 0.1%
        क : 0.09%
        ह : 0.07%
```

Fig 5.8 Character = fa

Chapter 6

CONCLUSION & FUTURE ENHANCEMENTS

6.1 CONCLUSION

Handwritten character recognition is a difficult task as the characters usually has various appearances according to different writer, writing style and noise. Researchers have been trying to increase the accuracy rate by designing better features, using different classifiers and combination of different classifiers. These attempts however are limited when compared to CNN. CNNs can give better accuracy rates but it has some problems that need to be addressed. Devanagari characters are complex due to their curved nature. Here this project built a system that recognizes Devanagari handwritten characters using Convolutional neural network approach. This project will greatly help to reduce the digitalization work of any Devanagari handwritten document.

OCR has a wide variety of real time applications. It can be used for office automation and teaching. This work implements a handwritten Devanagari character recognition system. The characters. Both Sample generation and CNN modeling are time consuming tasks and the later also requires a CUDA enabled GPU for parallel processing. Preprocessing helps to remove the undesired qualities of an image and hence can play an important role in increasing the role. So is the sample generation process that reduces overfitting. The drop out layer also reduces overfitting while also decreasing the overall training time. CNN has proved to be the state-of-the-art technique for other languages and hence provides the chance for giving higher accuracy rate for Devanagari characters too.

6.2 FUTURE ENHANCEMENTS

This model can classify 46 Devanagari characters .There are more symbols in Devanagari language it has to be improved to classify all the characters and symbols that is present in Devanagari language. Devanagari language has a peculiarity that unlike English language a character may have different meaning with respect to its position. In current system , if the input image containing words that are connected the current system may not give intended results. Some more researches have to be done on this. The state based model such as RNN (Recurrent neural networks) can be incorporated with this model so that we can improvise the current model.

The final model in this project uses only simple Convolutional Neural Network architecture. There are more advanced architecture like the Deep Residual Neural Network(ResNet) and VGG- 16 which could be used to improve the accuracy a little. Also we could hyperas library to fine tune the hyper-parameters and architecture of the model.

7. REFERENCES

[1] Ashwin S Ramteke, Milind E Rane "A Survey on Offline Recognition of Handwritten Devanagari Script". Available at
https://pdfs.semanticscholar.org/0bdd/aeae131a4bcfa7832d10efb049420cc875ab.pdf

[2] Aradhana A Malanker , Prof. Mitul M Patel "Handwritten Devanagari Script Recognition: A Survey". Available at
http://www.iosrjournals.org/iosr-jeee/Papers/Vol9-issue2/Version-2/L09228087.pdf

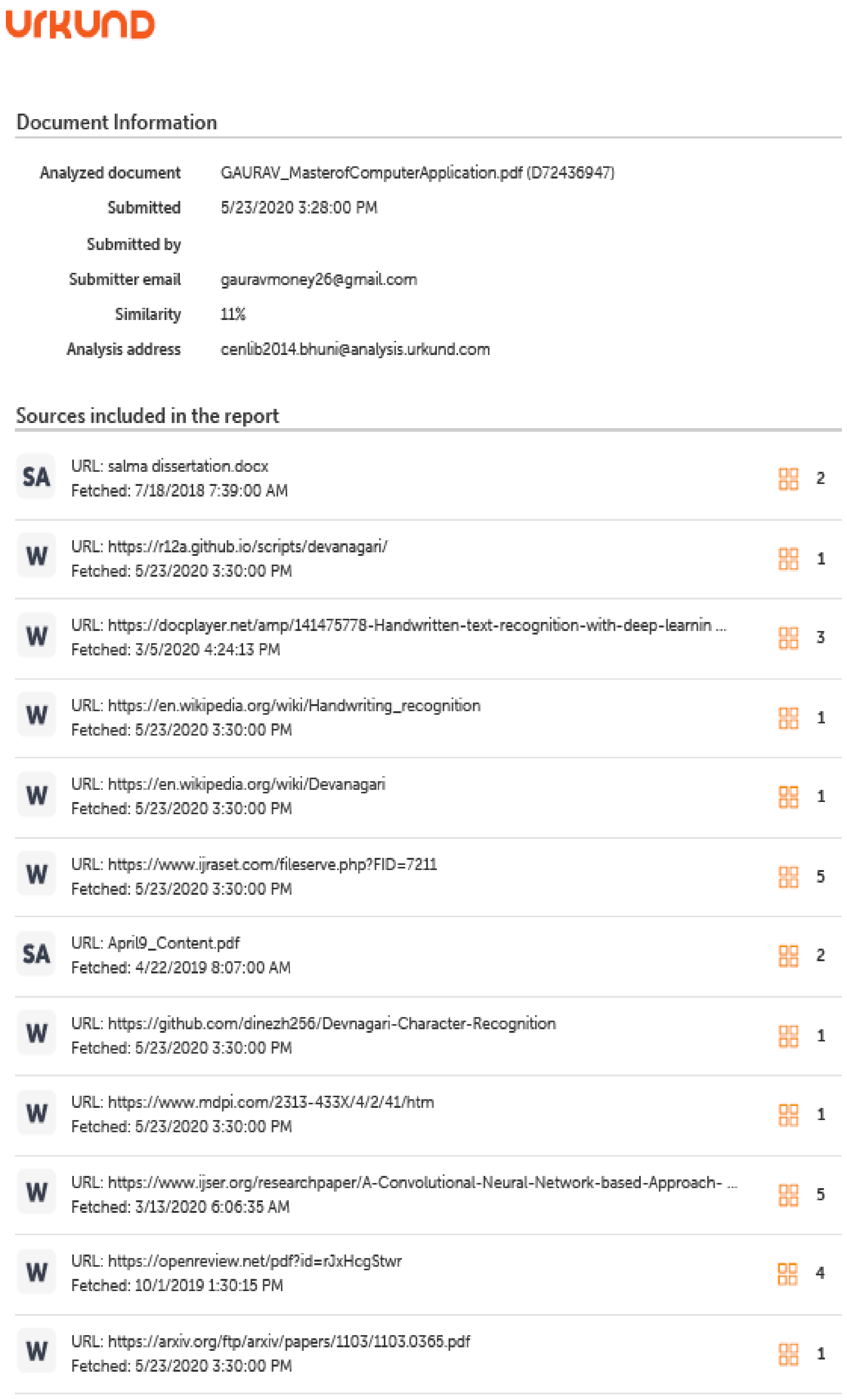
[3] J.Pradeep, E.Srinivasan and S.Himavathi "DIAGONAL BASED FEATURE EXTRACTION FOR HANDWRITTEN ALPHABETS RECOGNITION SYSTEM USING NEURAL NETWORK". Available at
https://arxiv.org/ftp/arxiv/papers/1103/1103.0365.pdf

[4] S S Sayyad, Abhay Jadhav, Manoj Jadhav, Smita Miraje, Pradip Bele, Avinash Pandhare, "Devnagiri Character Recognition Using Neural Networks", International Journal of Engineering and Innovative Technology (IJEIT)Volume 3, Issue 1.

[5] Ms. Seema A. Dongare , Prof. Dhananjay B. Kshirsagar, Ms. Snehal V. Waghchaure "Handwritten Devanagari Character Recognition using Neural Network", IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727 Volume 16, Issue 2.

[6] https://en.wikipedia.org/wiki/Handwriting_recognition

8. PLAGIARISM REPORT


URKUND

Document Information

Analyzed document	GAURAV_MasterofComputerApplication.pdf (D72436947)
Submitted	5/23/2020 3:28:00 PM
Submitted by	
Submitter email	gauravmoney26@gmail.com
Similarity	11%
Analysis address	cenlib2014.bhuni@analysis.urkund.com

Sources included in the report

SA	URL: salma dissertation.docx Fetched: 7/18/2018 7:39:00 AM	2
W	URL: https://r12a.github.io/scripts/devanagari/ Fetched: 5/23/2020 3:30:00 PM	1
W	URL: https://docplayer.net/amp/141475778-Handwritten-text-recognition-with-deep-learnin ... Fetched: 3/5/2020 4:24:13 PM	3
W	URL: https://en.wikipedia.org/wiki/Handwriting_recognition Fetched: 5/23/2020 3:30:00 PM	1
W	URL: https://en.wikipedia.org/wiki/Devanagari Fetched: 5/23/2020 3:30:00 PM	1
W	URL: https://www.ijraset.com/fileserve.php?FID=7211 Fetched: 5/23/2020 3:30:00 PM	5
SA	URL: April9_Content.pdf Fetched: 4/22/2019 8:07:00 AM	2
W	URL: https://github.com/dinezh256/Devnagari-Character-Recognition Fetched: 5/23/2020 3:30:00 PM	1
W	URL: https://www.mdpi.com/2313-433X/4/2/41/htm Fetched: 5/23/2020 3:30:00 PM	1
W	URL: https://www.ijser.org/researchpaper/A-Convolutional-Neural-Network-based-Approach- ... Fetched: 3/13/2020 6:06:35 AM	5
W	URL: https://openreview.net/pdf?id=rJxHcgStwr Fetched: 10/1/2019 1:30:15 PM	4
W	URL: https://arxiv.org/ftp/arxiv/papers/1103/1103.0365.pdf Fetched: 5/23/2020 3:30:00 PM	1