A

PROJECT REPORT

ON

Handwritten Recognition System

Submitted by

Hariti Bhatia (18IT407)
Bhavisha Patel (18IT424)
Foram Patel (18IT425)

For Partial Fulfillment of the Requirements for Bachelor of Technology in Information Technology

Guided by

Prof. Bijal N. Dalwadi

Dec, 2021

Information Technology Department

Birla Vishvakarma Mahavidyalaya Engineering College

(An Autonomous Institution)

Vallabh Vidyanagar – 388120

Gujarat, INDIA

Birla Vishvakarma Mahavidyalaya Engineering College

(An Autonomous Institution)

Information Technology Department

AY: 2021-22, Semester I

CERTIFICATE

This is to certify that the project work entitled **Handwritten Recognition System** has been successfully carried out by **Hariti Bhatia (18IT407), Bhavisha Patel (18IT424), Foram Patel (18IT425)** for the subject **Project I (4IT31)** during the academic year 2021-22, Semester-I for partial fulfilment of Bachelor of Technology in Information Technology. The work carried out during the semester is satisfactory.

Prof. Bijal N. Dalwadi
IT Department
BVM

Dr. Keyur Brahmbhatt
Head, IT Department
BVM

Acknowledgement

We would like to express our gratitude and appreciation to all those people who give us the possibility and helped us most throughout the semester to complete this project. We are very grateful to our project guide Prof. Bijal Dalwadi for giving us valuable guidance and kind supervision throughout the entire project. A special thanks to all who encouraged and supported us for our hard work to complete this project.

Hariti Bhatia (18IT407)

Bhavisha Patel (18IT424)

Foram Patel (18IT425)

Abstract

Handwriting recognition also called Handwritten Text Recognition 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. As handwriting recognition capability highly depends on neural networks, advances in these algorithms profoundly affect the performance of handwriting recognition tools. Thus, active research on handwriting recognition is generally based on neural network algorithms. This handwritten recognition system recognizes the 446 gujarati characters by using CNN model to train on the image dataset of 31765 images. The website will take input of gujarati character as image and it will give the result as an output to us.

List of Figures

List of Abbreviations

Sr No.	Abbreviations	Explanation
1	HCR	Handwritten Character Recognition
2	OCR	Optical Character Recognition

3	CNN	Convolutional Neural Network

Table of Content

Chapter 1: Introduction

1.1 Brief overview of the work

The project aims to develop a website that takes gujarati handwritten characters as input and extract characters from it. At first the dataset is made of 31735 images for 446 gujarati characters along with 0-9 digits. Then using these dataset the CNN model is created and then this dataset is trained on that model. Then after testing the model we created the website which is made with the help of bootstrap, html and css for the design purpose. Then the CNN model and application python code is embedded to the website using python flask.

1.2 Objective

The objective of this project is to collect the handwritten gujarati characters dataset. To decide the machine learning model appropriate for our dataset and then train it. After that testing the model and creating website for it. And finally integrating model to the website. Therefore, our main purpose is to recognize the gujarati handwritten characters and convert it to readable text.

1.3 Scope

This website can be used by anyone to recognize handwritten gujarati character and convert it to a readable format. The website might recognize character with an accuracy of 93.40%. The website capability also depends on the quality of image, if quality is good then it is able to recognize the character easily and with greater accuracy.

1.4 Project Modules

- Image pre processing
- Feature Extraction
- Creating CNN module
- Training and testing of module
- Classification and Recognition

1.5 Project Hardware/Software Requirements

- Windows 10
- RAM 12 or 6 GB
- Any Text editor/IDE for website
- Tensorflow, OpenCV, Keras, Numpy, Matplotlib, Pandas libraries.
- Language: Python 3.9
- IDE: Jupyter, Google Colab
- Python Flask for framework.

Chapter 2: Literature Review

2.1 Handwritten Character Recognition using Neural Network and Tensor Flow. [1]

Review:

This paper describes the offline handwritten character recognition using Convolutional neural network and Tensorflow. The purpose of this paper the handwritten characters or typed documents is simple to the human beings as we have the ability to learn. The same ability can be induced to the Machines also by the use of Machine Learning and Artificial Intelligence. The field which deals with this problem is called as the OCR or also known as Optical Character Recognition. It's the area of study among various fields such as recognizing of pattern, also Image vision and also AIis to develop the software with a very high accuracy rate and with minimal time and space complexity and also optimal.

Normally HCR is categorized into six phases which are acquisition of image, pre-processing of input image, segmentation, feature extraction, classification and post processing.

- Preprocessing: It deals with the methods necessary to construct the raw input images.
- Segmentation: It deals with segmenting the raw image into single characters and then arranging it into m*n pixels to deal with it mathematically.
- Feature extraction: It is the method is perhaps of main significant issue in achieving high recognition percentage for which various techniques like Deformable templates, Gradient feature, Contour profiles, Fourier descriptors, Gabor features, Graph description, etc.
- Classification: The decision making is done in the classification phase. For recognizing the characters, the extracted features are used. Different classifiers like SVM and Neural Networks are used. The classifiers sorts the given input feature with reserved pattern and find the best matching class for input, for which Soft Max Regression is used.

- Post processing: It is the procedure for correcting the misclassified output by using natural language. It processes output by getting it after the shape have been recognized. If the shape is recognized purely then the accuracy can be improved in accordance with the knowledge of language.

It can be concluded that feature extraction method like diagonal and direction techniques are way better in generating high accuracy results compared to many of the traditional vertical and horizontal methods. Also using a Neural network with best tried layers gives the plus feature of having a higher tolerance to noise thus giving accurate results. Also, bigger our training data set and better neural network design, the better accurate is the result.

2.2 Handwritten character recognition using convolutional neural network [2].

Review:

Handwritten character recognition (HCR) is the detection of characters from images, documents and other sources and changes them in machine-readable shape for further processing. The main focus of this work is to investigate CNN capability to recognize the characters from the image dataset and the accuracy of recognition with training and testing. CNN recognizes the characters by considering the forms and contrasting the features that differentiate among characters.

Convolutional Neural Network (CNN) is a very well-known deep learning architecture motivated by the natural visual perception technique of human brain. there are many variants of CNN architectures, their basic elements are very similar. It comprised of three types of layers, as convolutional, pooling, and fully-connected layers. It has brought great advances in the area of computer vision and it has been introduced to the HCR to achieve excellent recognition performance. The 3 layers discussed are:

- Convolution: It is commonly used only for image manipulation, such as camera smoothing, feature enhancement and sharpening. Convolution not only used in CNNs, it is still a core feature in several other algorithms for machine learning. It is a process where a single matrix of numbers (known a kernel or filter), pass this over an image to reshape the image with the filter. Each selected pixel value of the image submatrix needs to be multiplied with the corresponding pixel value from the kernel and then sum up the result to formulate a single pixel value of the filtered image.
- Pooling: The purpose is to slowly raise the spatial scale of the image description to reduce the computational complexity within the network. Max-Pooling gives maximum value from its image part covered within the kernel. But on the other side, average pooling provides better noise suppression. It discards the noisy activations and also conducts de-noise along with reduction of the dimensionality.
- Fully-connected neural network: It receives the end result of convolution/pooling operation and computes the most-matched label that represents the image. This part makes the connection of the image feature vector with the class of the image. The outputs from the convolution/pooling operation are multiplied by the weights associated with the network connection path. The result is then passed through an activation function.

The proposed model in this paper uses four stages for the classification and detection purpose: Pre-processing, Segmentation & Clipping, Feature extraction and Classification & Recognition.

- Pre-processing: Pre-processing takes input image is to perform cleaning tasks. It effectively enhances the image by noise removal.
- Segmentation: The characters after pre-processing are then stored into a sequence of images. Then boarders in each character image are eliminated if the boarder is available. Next, individual characters are scaled to specific size
- Feature extraction: The features are extracted using CNN with ReLU activation function. CNN works on each character image to form a matrix of reduced size

using convolution and pooling. Finally, the reduced matrix is compacted to a vector form using the ReLU function. This vector is regarded as feature vector.

- Classification & Recognition: During the training phase, the parameters, biases, and weights are calculated. The calculated parameters, biases, and weights are used in the testing phase for classification and recognition purposes.

From this paper, it is concluded that the accuracy obtained from 200 training images for NIST dataset as 65.32% is improved gradually with increasing training images. The accuracy reaches to 92.91% with the 1000 training images. Thus, increasing the number of training images can lead to increase accuracy.

2.3 Machine Learning for Handwriting Recognition [3]

Review:

This paper discusses how machine learning can be used for handwritten recognition. Machine learning tries to extract hidden information that lies in the data. Pattern recognition is one of the main application of ML. Patterns are usually recognized with the help of large image data-set. Handwriting recognition is an application of pattern recognition through image.

Hand write recognition is one of the type of Optical Character Recognition(OCR). OCR is identification of text, which may be printed or hand-written. In OCR, the document is captured via camera as image and can be converted to desired formats like PDFs. Then the file is fed to the algorithm for character recognition. The two derivatives of OCR are Printed character recognition and Hand-written character recognition. Printed character recognition is as the name suggests, recognition of characters in the image of news paper or any other form of printed document. Hand-written character recognition involves recognition of characters written by human or has human involvement in it.

It is divided into two types namely Online character recognition and Offline character recognition. Offline character recognition involves parsing image of document to series

of texts and words. Online character recognition is a bit complicated process. It is a dynamic process. It involves of recognizing character data at the time of writing itself. Image source, preprocessing, segmentation, feature extraction and classification are the phases of handwritten character recognition.

There are many algorithms to recognize the handwriting using Machine Learning like:

- Convolutional Neural network
- Semi Incremental Recognition
- Incremental Recognition
- Line and word segmentation
- Part based Method
- Slope and Slant correction method
- Ensemble method.

Therefore, it is concluded from this paper that there are many approaches for hand writing recognition. Among these methods, highest accuracy is achieved from Convolutional Neural Network (CNN) and the least accuracy is achieved from Slope and Slant Correction method. When the images are trained with CNN, we will achieve good accuracy but only disadvantage with this method is that training time of the model is too high because lot of image samples are included. In Zoning method, if zones which are achieved after dividing input image and if the count of these zones are lesser then accuracy will decrease. Main disadvantage of this method is that developers will face lot of problems while segmentation process but this method is too simple for hand writing recognition. This method only sees the Lat and which makes it simple.

2.4 Handwritten Character Recognition Using CNN [4]

Review:

This paper presents the related and proposed work. It also presents the module description and the experimental results for handwritten character recognition and finally

the performance evaluation. Here, EMNIST dataset is extended by adding some more characters from Tamil language. First the input image is provided and is converted into a gray-scale image and normalized in such a way that it represents the same resolution (28 x 28) as that of EMNIST dataset. CNN is trained using EMNIST dataset and use it as a classifier which will yield better results when compared with other machine learning algorithms. The feature vectors are extracted from the input image and provided to the trained model of Convolution Neural Network which recognizes and provides the desired output.

The various modules includes Pre-processing, Feature Extraction, MinMaxScaler -fitting of data, Image normalization and Classification.

- Pre-processing: Pre-processing of input image is carried out by converting the given image into gray-scale image. The coloured image is converted it to gray-scale image which consist of single monochrome channel in order to avoid unwanted noise in the image. o the image is resized and placed upon a empty 28 x 28 pixel blank image so that the image resolution matches the resolution of EMNIST dataset.
- Feature Extraction: It is the process of transforming the input data into a set of features which can very well represent the input data. Feature extraction is related to dimensionality reduction. When the input data is too large to be processed, then it can be transformed into a reduced set of features. Determining a subset of the initial features is called feature selection. After resizing the image, pixel values are obtained in the form of 1D array which represents values between 255 and 0 based on pixel intensity.
- Min Max Scaler: The min-max scalar form of normalization uses the mean and standard deviation to box all the data into a range lying between a certain min and max value. It transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set.
- Image Normalization: Normalization is a process that changes the range of pixel intensity values. Normalization is sometimes called contrast stretching or

histogram stretching. In this input image the normalization is carried out by removing the background pixels and the character alone will be provided as it is in the image.

- Classification: Convolutional neural network is used as a classifier for classifying the handwritten character from the input image. A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers. A CNN consists of three major components which are convolutional layer, pooling layer and output layer. The activation function that is commonly used with CNN is ReLU which stands for Rectified Linear Unit.

  The softmax function is often used in the final layer of a neural network-based classifier. The softmax function squashes the outputs of each unit to be between 0 and 1. The output of the softmax function is equivalent to a categorical probability distribution. Thus, softmax function calculates the probabilities distribution of the event over 'n' different events.

  Thus, we can conclude that due to development of machine learning , image recognition for computer vision has seen enormous improvement. In this paper, EMNIST dataset provides about 132,000 images of 47 characters to be trained and recognised. The convolution neural network was used to train EMNIST dataset to obtain high accuracy. The EMNIST dataset is extended to support the 12 characters from Tamil language and the recognition of these character are tested. The input image is pre-processed, standardized normalized and given to the classifier to predict the character.

2.5 Gujarati Handwritten Character Recognition from Text Images. [5]

Review:

The HCR systems are readily available for foreign languages and many of the Indian languages like Bangla, Devanagari and Gurumukhi but for Gujarati language the HCR

development is still in its infancy stage. This paper describes a supervised classifier approach based on CNN (Convolutional Neural Networks) and MLP (Multi-Layer Perceptron) for recognition of handwritten Gujarati characters. A success rate of 97.21% is obtained using CNN and 64.48% using MLP.

The very first step described in this paper is the dataset generation. Nearly 10000 images is collected and grouped into train and test set with an 80-20% ratio. The dataset images are pre-processed, binarized and segmented into individual characters of size 28 x 28 pixels and then each character is stored into its corresponding class folder. These dataset images are used to train our classifier. The binary image has characters (foreground) as 1 and background as 0. There is a total of 59 classes including vowels, modifiers (Matras), and consonants. Then, going through below phases.

- Pre-processing: It is the first and foremost step required for the Character recognition system. To convert the real time images in the same format as our dataset, and to reduce noise and remove unwanted background, the following pre-processing techniques are applied.
    1. Scanning and Resizing
    2. Noise Removal and Binarization
    3. Skew Correction
- Segmentation: Segmentation can be done by, identifying the sub-components, lines, and words and then character segmentation. These segmentation units are executed one after another. Segmentation techniques used in this paper are:
    1. Connected Component Labelling
    2. Histogram Projection Profile
    3. Text Blob detection
    4. Contour segmentation
- Classification: Classification prediction was made on test data, in order to estimate the skill of the model, on unseen data using MLP and CNN models. Due to small size of dataset, data augmentation such as rotation, shift and normalization has been done to the character dataset to increase size of the data

during training. An Accuracy of 64.87% was achieved in MLP & 97.21% accuracy was achieved using CNN in the experiment done in this paper.

- Post-Processing: The prediction contains many candidate characters, this can be corrected using below approaches:
  1. Grouping
  2. Error identification and correction
  3. Text file generation and Unicode encoding

Finally, a simple GUI is built to bind all these processes and show only the desired output to the user and then the output is saved in the text file.

Thus, it can be concluded from this paper that how an artificial intelligence based offline HCR system for Gujarati language can be build going through different phases and how we can improve the accuracy of the model with the certain described techniques in the paper.

Chapter 3: System Analysis & Design

3.1 Comparison of Existing Applications with your Project with merits and demerits.

The Handwritten recognition system are already available for languages like English, Bangla, Hindi , Tamil, etc. Also, Gujarati handwritten character recognition existing applications have done just for only the individual ka, kha and so on upto gya. Not everybody have done including the gujarati barakhadi in their dataset. Some of the related projects have achieved the accuracy of around 90%.

Our web application can predict any 446 gujarati characters including 0-9 digits with the accuracy of 92.08%. Also, the website design makes the handwritten gujarati character recognition system attractive. The further features like handwritten recognition of gujarati

characters along with sentences and words from the entire document can be done with more research in this area and so this can be considered as our demerit.

3.2 Project feasibility study

- Technical Feasibility

  For Handwritten recognition, the system will interact with the wide range of users and is available 24*7. One can do the handwritten text recognition anytime with just taking an image and uploading it to get the output.

- Operational Feasibility

  The output will be based on the uploaded image so there will be no error except any problem with the type of image uploaded for handwritten recognition. And the programs to perform these operations will be tested and analyzed multiple times so no space for error will be possible.

- Implementation Feasibility

  Implementation of this website is very feasible as it is very ease to use and the interface is user friendly to understand by the consumers. Only it requires good internet connection for better interactivity.

- Economic Feasibility

  The usage of the website will be completely free of cost as the application is focused to be build for the goodwill of users. Only the website will use the internet data and battery for the application from the user side.

3.3 Project timeline chart

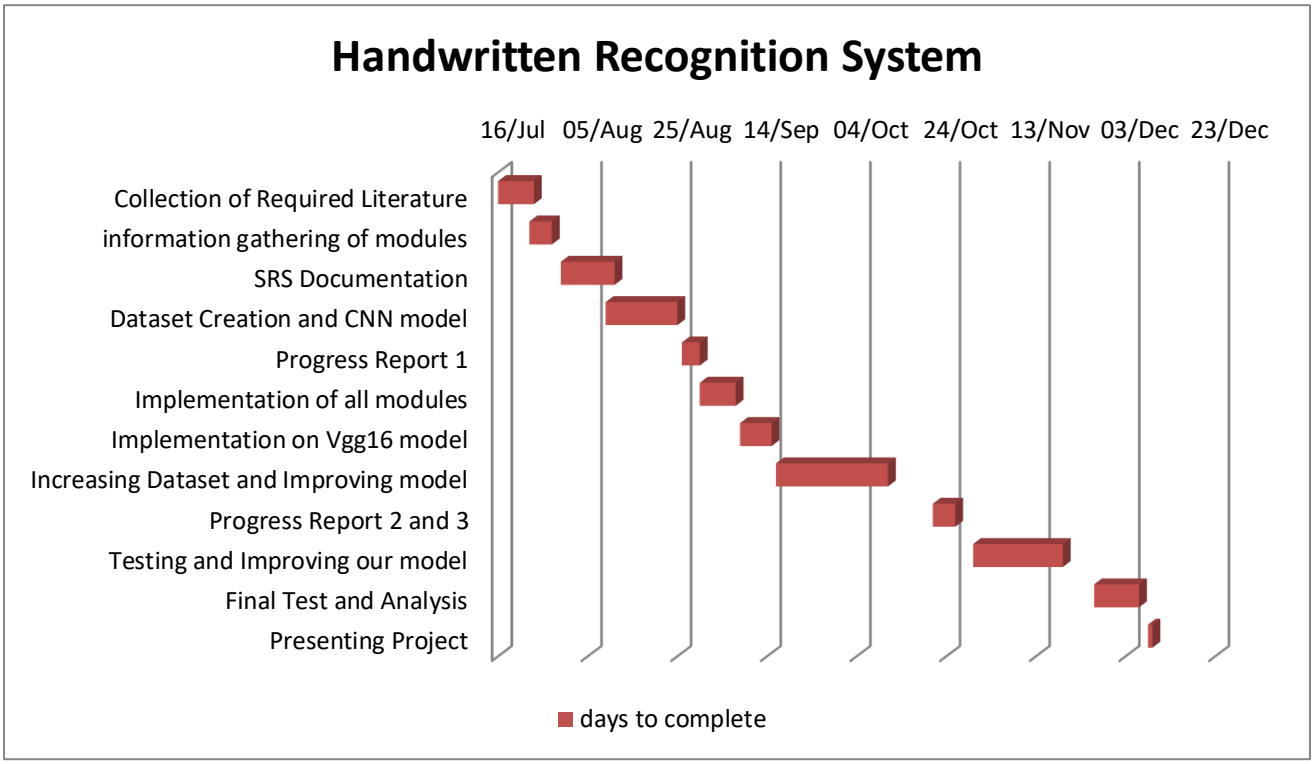


Figure 3.3 Timeline chart

3.4 Detailed Modules Description

- Image pre processing

  The role of image pre processing is to perform various tasks on input image by making it suitable for further feature extraction of character. Mostly noise filtering, smoothing and standardization are done in this stage to take off the fascinating things from the background besides the text in image. Perprocessing is performed by binarization, grayscaling and inversion of the image. Here, we resize our all images to a size of 28*28 with grayscaling.

- Feature Extraction

  Once the character is preprocessed we generate the binary glyphs which contains pixels of images that are of very large size in the form of matrix. Now calculating the summary of each rows and columns extracts the important features. The labeling and converting to image to array for feature extraction is with the help of flow_from_directory() method. Here we labeled the images belonging to their character by creating a csv file for it containing information about 784 pixels of each image with their label.

- Creating Convolutional Neural Network model

  CNN model created is used to extract the features of the images using several layers of filters. The layers are generally followed by maxpool layers that are used to reduce the number of features extracted and ultimately the output of the maxpool and layers and convolution layers are flattened into a vector of single dimension and are given as an input to the Dense layer. And the Dropout layer will prevent overfitting. Also, Batchnormalization and regularization techniques are added to avoid overfitting and increasing the training speed.

- Training and testing of model

This is an important module for handwritten recognition because here the CNN model is trained and tested to improve the accuracy of the model to predict the characters more efficiently. In our work, 92.08% testing and 92.52% training accuracy is achieved. Also, we tested our dataset with VG166 pre-trained model where we get nearly 99% accuracy.

- Recognition

  In this module the gujarati handwritten character is recognized. We recognized different gujarati characters images from inside and even outside of the dataset

3.5 SRS Diagrams.

3.5.1 Use Case Diagrams

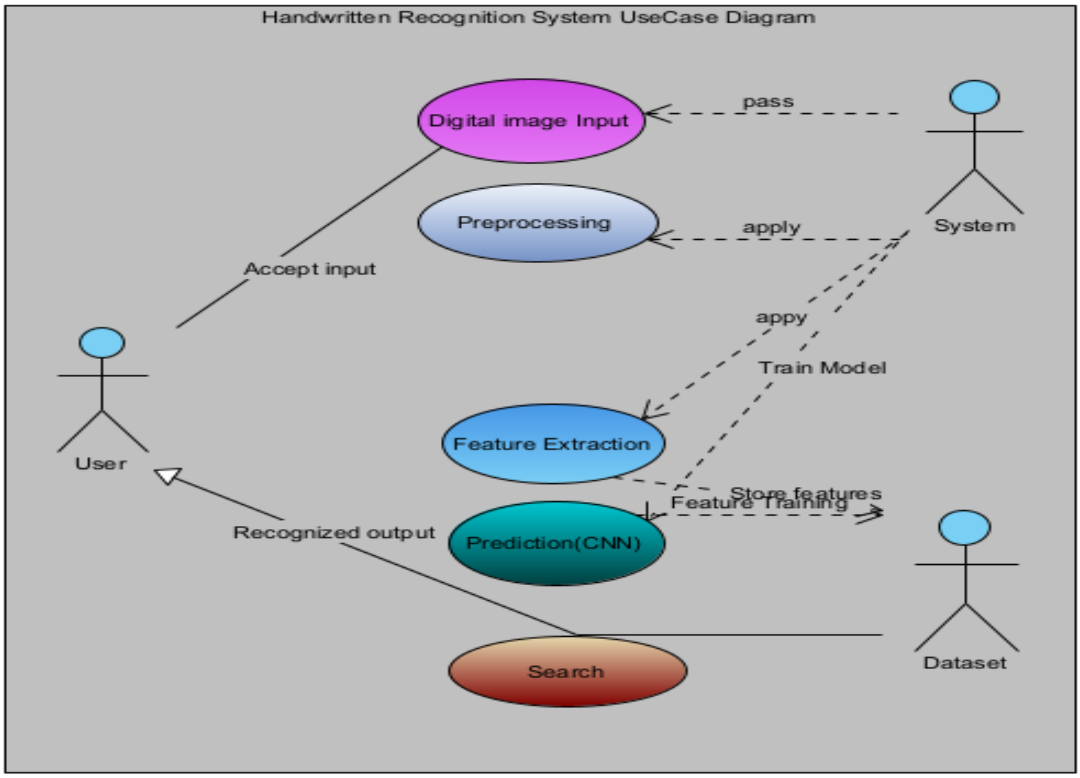


Figure 3.5.1 Use Case Diagram

3.5.2 Data Flow Diagrams

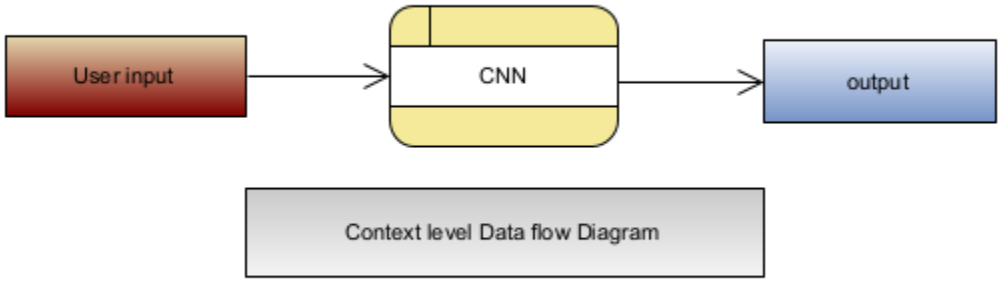


Figure 3.5.2.1 Data Flow Diagram

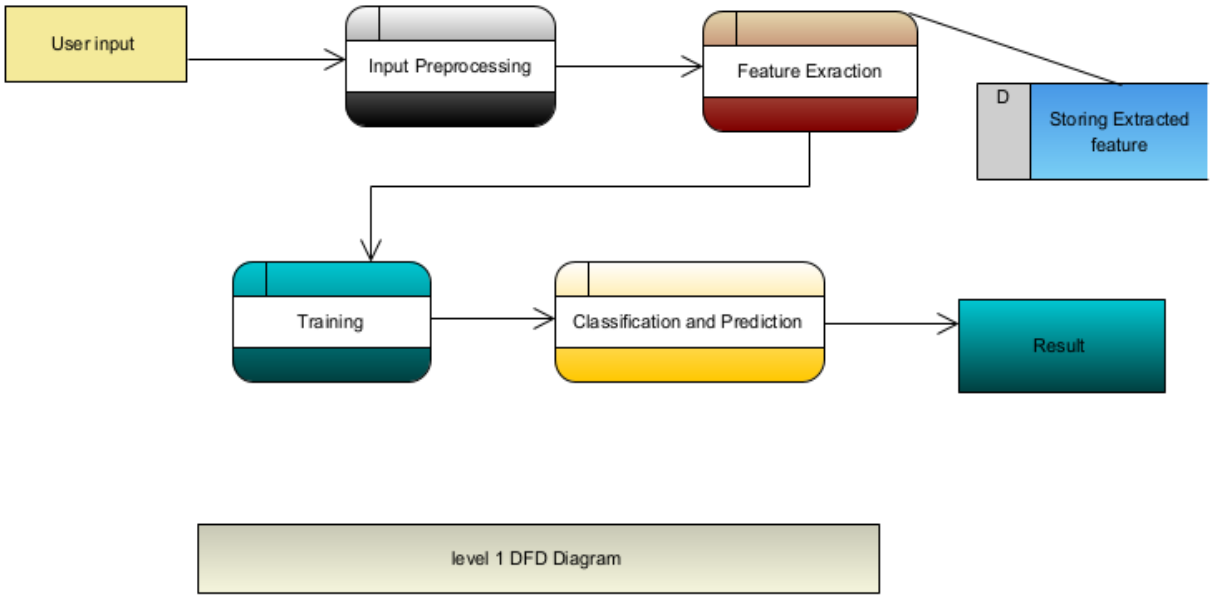


Figure 3.5.2.2 Data Flow Diagram

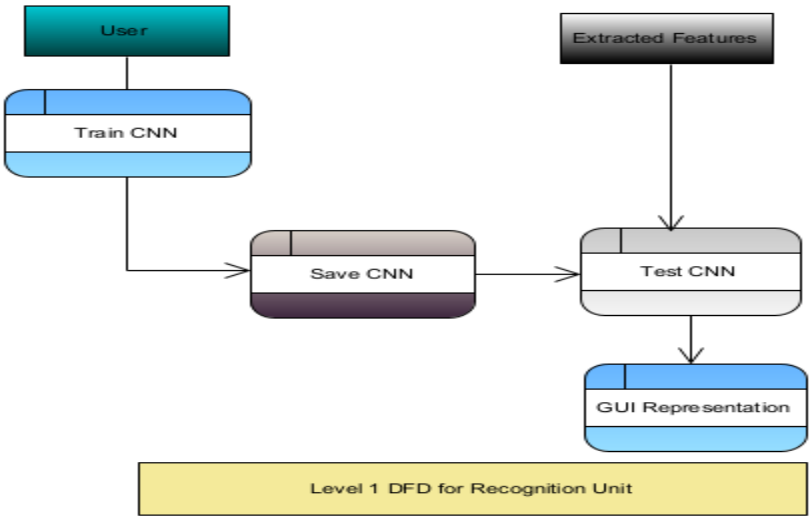


Figure 3.5.2.3 Data Flow Diagram

3.5.3 Class Diagram

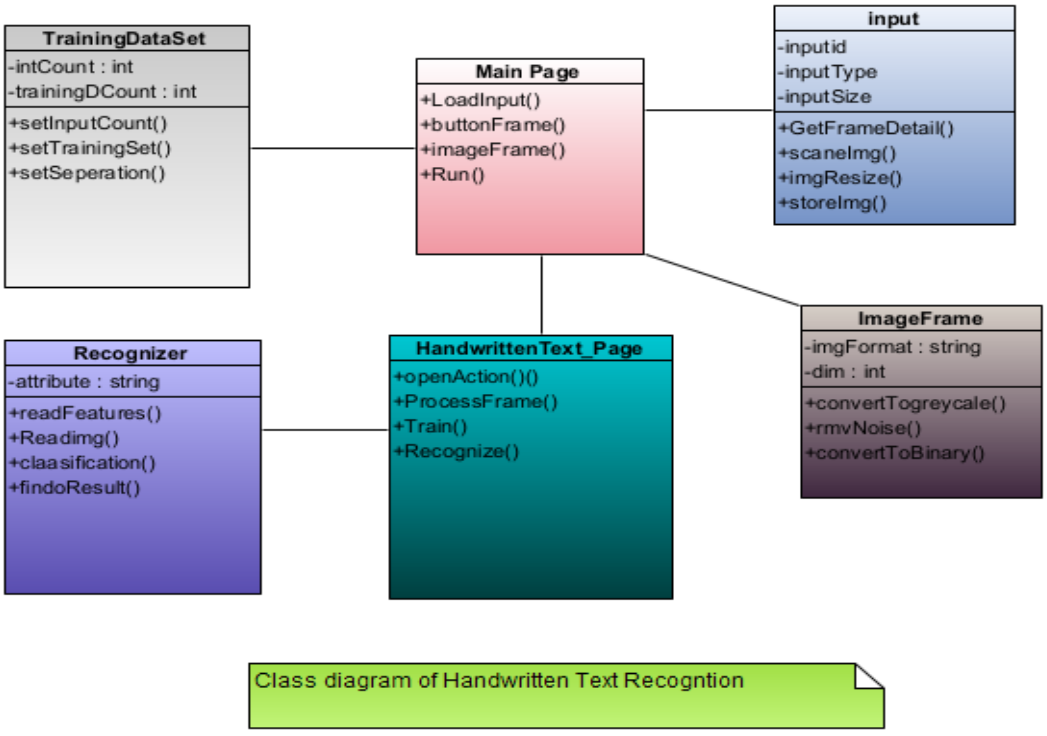


Figure 3.5.3 Class Diagram

3.5.4 Entity Relationship Diagram

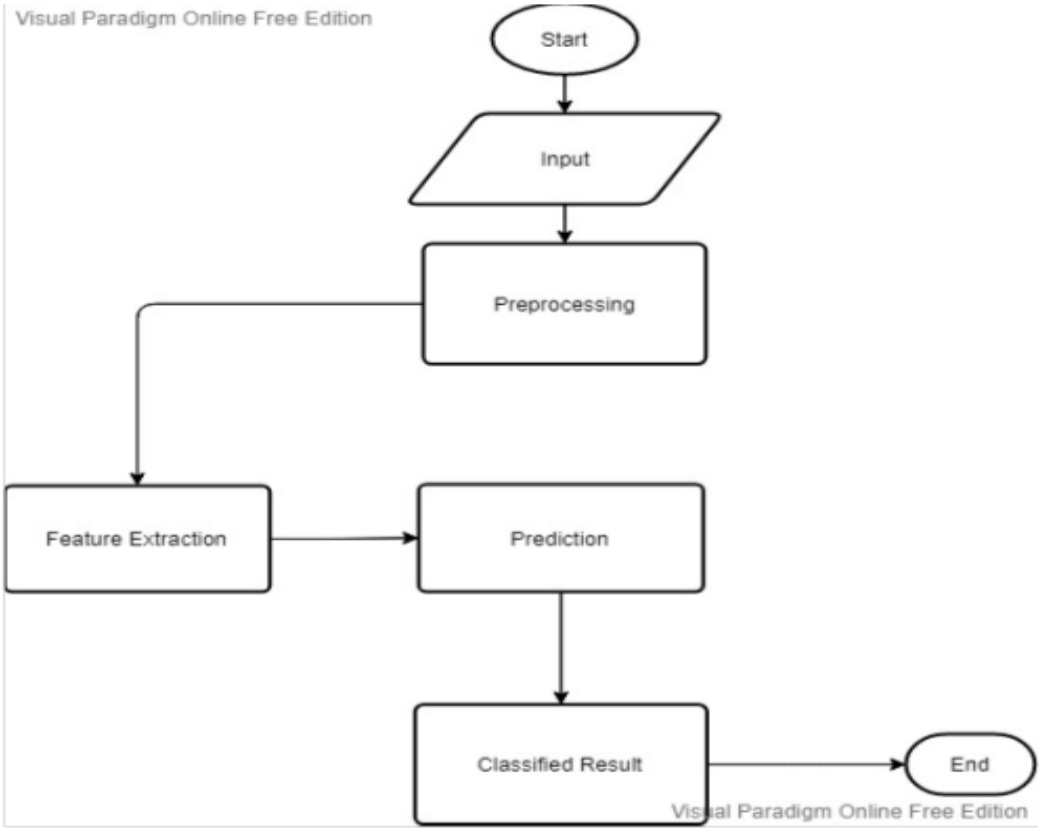


Figure 3.5.4 Entity Relationship Diagram

3.5.5 Sequence Diagram

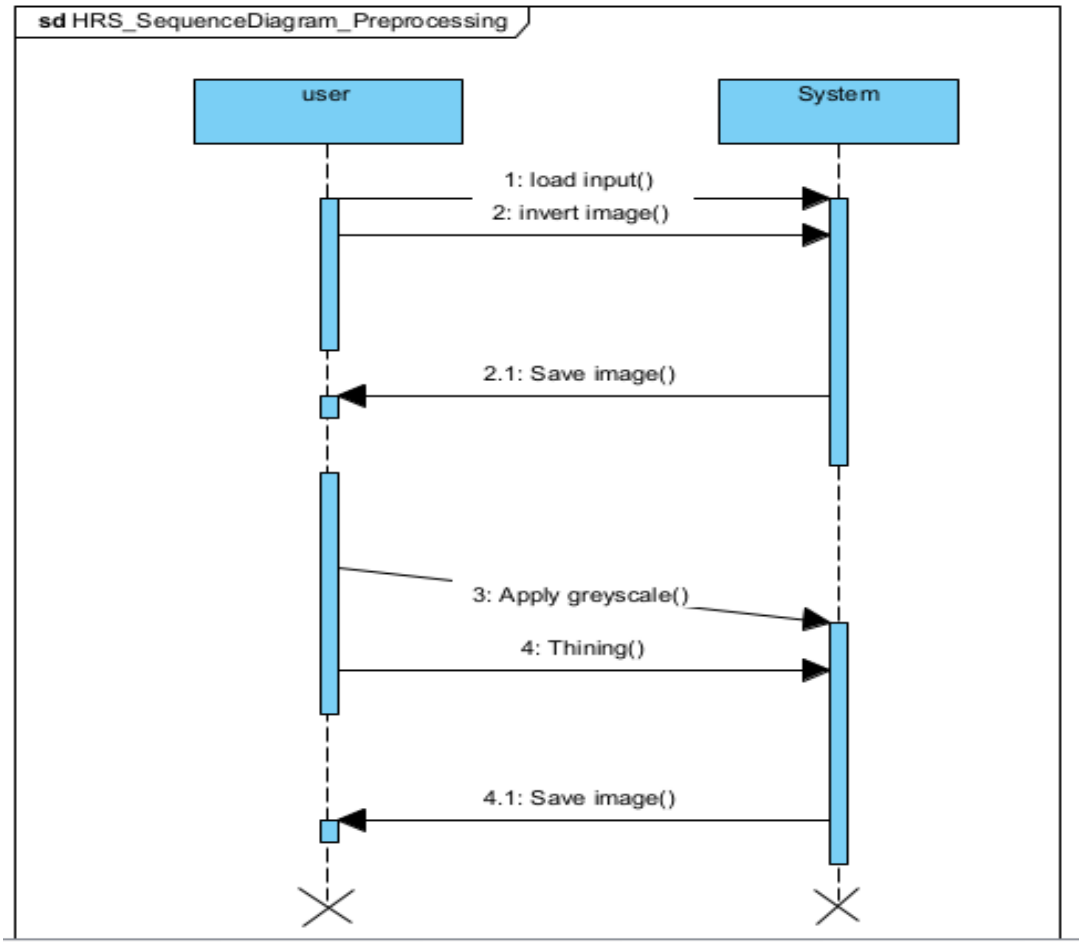


Figure 3.5.5.1 Sequence Diagram

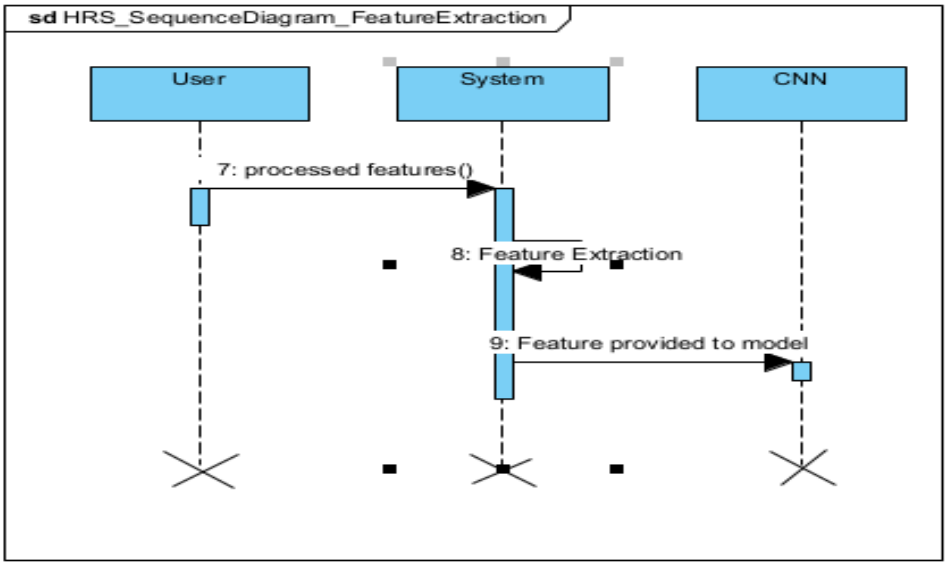


Figure 3.5.5.2 Sequence Diagram

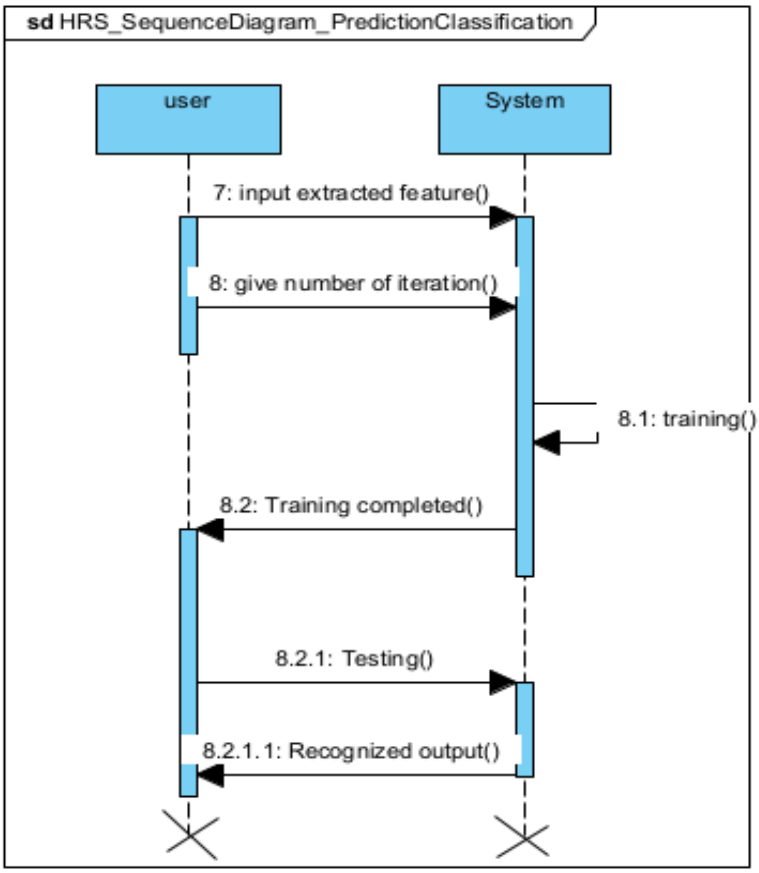


Figure 3.5.5.3 Sequence Diagram

3.5.6 State Diagram

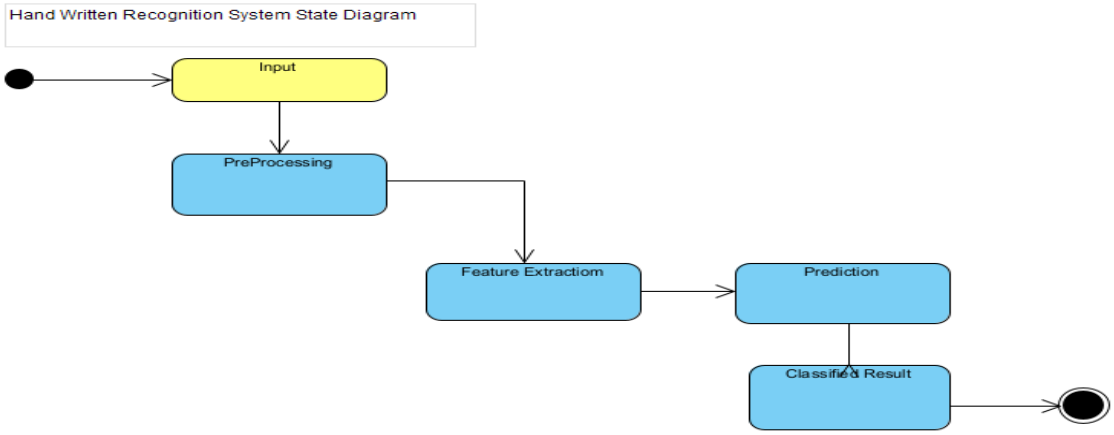


Figure 3.5.6 State Diagram

Chapter 4: Implementation and Testing

4.1 User Interface and Snapshot

At first the dataset is created of 31765 images of 446 gujarati characters including 0-9 Digits and the image information of 785 pixels with their respective label is stored in a csv file named Dataset.csv.

```
In [1]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
        import time
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        import tensorflow as tf
        import os
```

```
In [2]: from google.colab import drive
        drive.mount('/content/gdrive')

        Mounted at /content/gdrive
```

```
In [3]: import pandas as pd
        data = pd.read_csv("gdrive/My Drive/Dataset.csv").astype('float32')
```

```
In [4]: X = data.drop('Label',axis = 1)
        y = data['Label']
```

```
In [5]: from sklearn.model_selection import train_test_split
        train_x, test_x, train_y, test_y = train_test_split(X, y, test_size = 0.2)

        train_x = np.reshape(train_x.values, (train_x.shape[0], 28,28))
        test_x = np.reshape(test_x.values, (test_x.shape[0], 28,28))

        print("Train data shape: ", train_x.shape)
        print("Test data shape: ", test_x.shape)

        Train data shape:  (25412, 28, 28)
        Test data shape:  (6353, 28, 28)
```

```
In [6]: word_dict={'char_0': 0, 'char_1': 1, 'char_100_ક': 2, 'char_101_ખ': 3, 'char_102_ગ': 4, 'char_103_ઘ': 5, 'char_104_ચ': 6, 'ch
```

Figure 4.1.1

First all the required libraries are imported. The data is read using read_csv().Splitting the data read into the images & their corresponding labels. The 'Label' contains the labels, & so we drop the 'Label' column from the data dataframe read & use it in the y to form the labels.

Then, we splitted the data into training and testing set using train_test_split. We reshape the images into 28*28 pixels as they are stored in 784 pixels in the csv file. We create a dictionary word_dict to map the integer values with the characters.

```
In [7]: y_int = np.int0(y)
        count = np.zeros(446, dtype='int')
        for i in y_int:
            count[i] +=1

        alphabets = []
        for i in word_dict.values():
            alphabets.append(i)

        fig, ax = plt.subplots(1,1, figsize=(10,10))
        ax.barh(alphabets, count)

        plt.xlabel("Number of elements ")
        plt.ylabel("Gujarati Characters")
        plt.grid()
        plt.show()
```

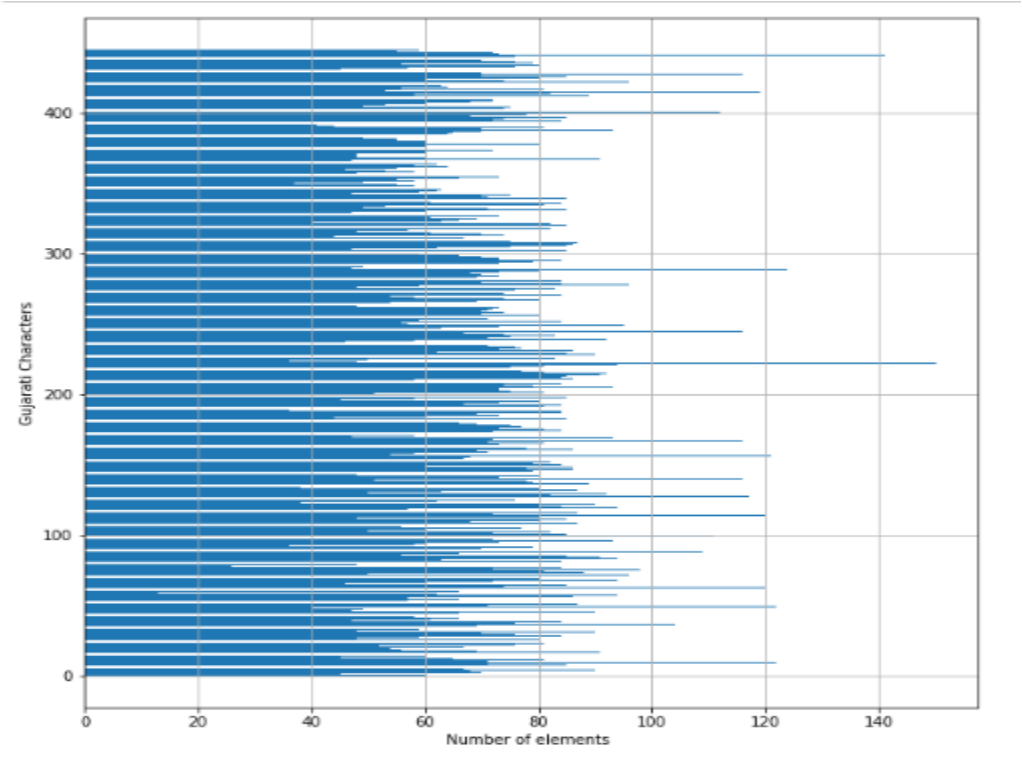


Figure 4.1.2

Here, we are only describing the distribution of the gujarati characters on a horizontal bar plot.

```
In [8]: import cv2
        from sklearn.utils import shuffle
        shuff = shuffle(train_x[:100])

        fig, ax = plt.subplots(3,3, figsize = (10,10))
        axes = ax.flatten()

        for i in range(9):
            _, shu = cv2.threshold(shuff[i], 30, 200, cv2.THRESH_BINARY)
            axes[i].imshow(np.reshape(shuff[i], (28,28)), cmap="gist_gray")
        plt.show()
```

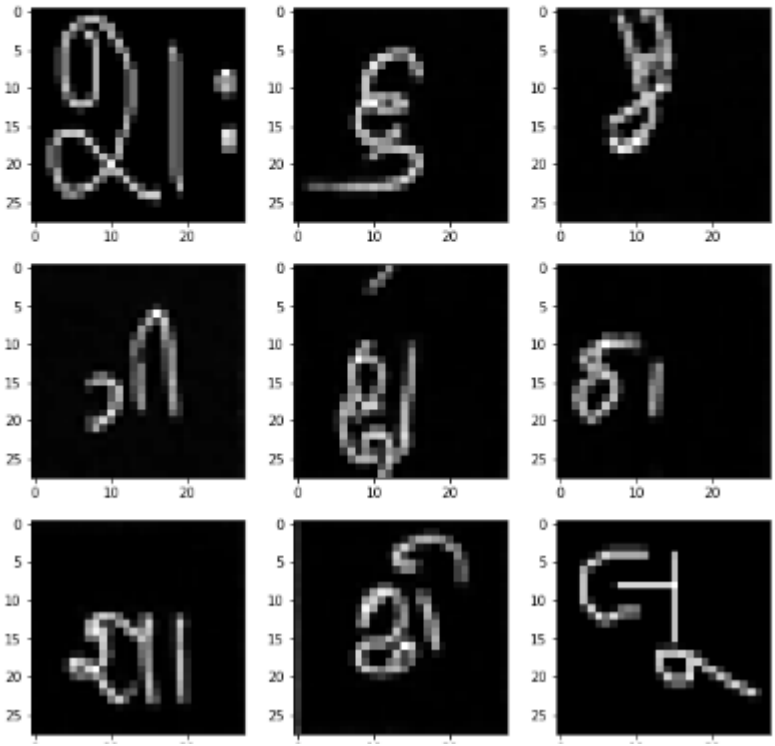


Figure 4.1.3

We shuffle some of the images using shuffle() and display some random images. It shows the grayscale images from the dataset.

```
In [9]: train_X = train_x.reshape(train_x.shape[0],train_x.shape[1],train_x.shape[2],1)
        print("New shape of train data: ", train_X.shape)
        test_X = test_x.reshape(test_x.shape[0], test_x.shape[1], test_x.shape[2],1)
        print("New shape of test data: ", test_X.shape)

        New shape of train data:  (25412, 28, 28, 1)
        New shape of test data:  (6353, 28, 28, 1)

In [10]: from tensorflow.keras.utils import to_categorical
         train_yOHE = to_categorical(train_y, num_classes = 446, dtype='int')
         print("New shape of train labels: ", train_yOHE.shape)
         test_yOHE = to_categorical(test_y, num_classes = 446, dtype='int')
         print("New shape of test labels: ", test_yOHE.shape)

         New shape of train labels:  (25412, 446)
         New shape of test labels:  (6353, 446)
```

Figure 4.1.4

We reshape the train and test image dataset so that they can be put in the model. Then, convert the single float values to categorical values. This is done as the CNN model takes input of labels and generates the output as a vector of probabilities.

```
In [11]: datagen = ImageDataGenerator(
             featurewise_center=False,
             samplewise_center=False,
             featurewise_std_normalization=False,
             samplewise_std_normalization=False,
             zca_whitening=False,
             rotation_range=50,
             width_shift_range=0.01,
             height_shift_range=0.01,
             horizontal_flip=False,
             vertical_flip=False)

         datagen.fit(train_X)

         from matplotlib import pyplot as plt

         gen = datagen.flow(train_X[1:2], batch_size=1)
         for i in range(1, 6):
             plt.subplot(1,5,i)
             plt.axis("off")
             plt.imshow(gen.next().squeeze(), cmap="gist_gray")
             plt.plot()
         plt.show()
```

Figure 4.1.5

Then, we do data augmentation to increase the accuracy.

In [12]:
```
from tensorflow.keras import regularizers
from tensorflow.keras.layers import BatchNormalization
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', kernel_regularizer=regularizers.l2(l=0.01),padding = 'same'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', kernel_regularizer=regularizers.l2(l=0.01),padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', kernel_regularizer=regularizers.l2(l=0.01),padding = 'valid'
model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu', kernel_regularizer=regularizers.l2(l=0.01),padding = 'valid'
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512,activation ="relu"))
model.add(Dropout(0.2))
model.add(Dense(256,activation ="relu"))
model.add(Dropout(0.2))
model.add(Dense(446,activation ="softmax"))

model.summary()
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 conv2d_1 (Conv2D)           (None, 24, 24, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)       0
 )

 batch_normalization (BatchN  (None, 12, 12, 32)       128
 ormalization)

 dropout (Dropout)           (None, 12, 12, 32)        0

 conv2d_2 (Conv2D)           (None, 12, 12, 64)        18496

 conv2d_3 (Conv2D)           (None, 12, 12, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 batch_normalization_1 (Batc  (None, 6, 6, 64)         256
 hNormalization)

 dropout_1 (Dropout)         (None, 6, 6, 64)          0

 conv2d_4 (Conv2D)           (None, 4, 4, 128)         73856

 conv2d_5 (Conv2D)           (None, 2, 2, 256)         295168

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 batch_normalization_2 (Batc  (None, 1, 1, 256)        1024
 hNormalization)

 dropout_2 (Dropout)         (None, 1, 1, 256)         0
```

```
dense (Dense)                (None, 512)               131584

dropout_3 (Dropout)          (None, 512)               0

dense_1 (Dense)              (None, 256)               131328

dropout_4 (Dropout)          (None, 256)               0

dense_2 (Dense)              (None, 446)               114622

=================================================================
Total params: 812,958
Trainable params: 812,254
Non-trainable params: 704
_________________________________________________________________
```

```
In [13]: from tensorflow.keras.optimizers import Adam
         model.compile(loss='categorical_crossentropy',  optimizer= Adam(learning_rate=3e-4), metrics=['accuracy'])
```

```
In [20]: history = model.fit(train_X, train_yOHE,epochs=3, batch_size=150, validation_data = (test_X,test_yOHE))

         Epoch 1/3
         170/170 [==============================] - 6s 31ms/step - loss: 0.4745 - accuracy: 0.9387 - val_loss: 0.5345 - val_accuracy: 0.
         9344
         Epoch 2/3
         170/170 [==============================] - 5s 29ms/step - loss: 0.4352 - accuracy: 0.9459 - val_loss: 0.5262 - val_accuracy: 0.
         9307
         Epoch 3/3
         170/170 [==============================] - 5s 29ms/step - loss: 0.4229 - accuracy: 0.9457 - val_loss: 0.5179 - val_accuracy: 0.
         9340
```


Figure 4.1.6

We created the CNN model that we designed for training the model over the training dataset. Then we compiled the model, where we define the optimizing function & the loss function to be used for fitting. We used Adam() with learning rate of 3e-4 as the optimization function.We then trained it for multiple epochs.

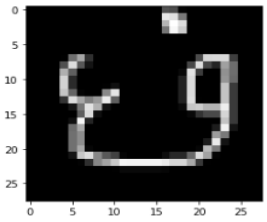
Finally, we get the 94.57% as the training accuracy and 93.40% as the test accuracy.

```
In [21]: import time
         time1 = time.time()
         time2 = time.time()
         score = model.evaluate(test_X,test_yOHE, verbose=0)
         time3 = time.time()
         print("Train time = ", (time2 - time1)/60, "min", "\nTest time = ", (time3 - time2)/ 60, "min", "\n Test loss: ", score[0], " T

         Train time =  3.218650817871094e-07 min
         Test time =  0.02326321601867675 8 min
          Test loss:  0.5179475545883179  Test accuracy:  0.9340469241142273

In [22]: model.save('model6.h5')

In [23]: dict= { 0 : '0', 1 : '1', 2 : '૨', 3 : '૩', 4 : '૪', 5 :'૫', 6 : '૬', 7 : '૭',8 : '૮', 9 : '૯', 10 : '૧૦', 11 : '૧૧', 12 :
               14 : '૧૪', 15 : '૧૫', 16 : '૧૬', 17 : '૧૭', 18 : '૧૮', 19 : '૧૯', 20 : '૨૦', 21 : '૨૧', 22 : '૨', 23 : '૨૩', 24 : '૨૪',25 :
               27 : '૨૭', 28 : '૨૮', 29 : '૨૯', 30 : '૩૦', 31 : '૩૧', 32 : '૩૨', 33 : '૩૩', 34 : '૩૪', 35 : '૩૫', 36 : '૩૬' ,37 : '૩૭', 38 :'૩
               40 : '૪૦', 41 : '૪૧', 42 : '૪૨',43 : '૪૩', 44 : '૪૪',45 : '૪૫', 46 : '૪૬', 47 : '૪૭', 48 : '૪૮', 49 : '૪૯', 50 : '૫૦', 51 : '૫૧',
               53 : '૫૩', 54 : '૫૪', 55 : '૫૫', 56 : '૫૬', 57 : '૫૭',58 : '૫૮',59 : '૫૯', 60 : '૬૦', 61 : '૬૧', 62 : '૬૨', 63 :'૬૩', 64 : '૬૪',
               67 : '૬૭', 68 : '૬૮', 69:  '૬૯', 70 : '૭૦', 71 : '૭૧',72 : '૭૨',73 : '૭૩',74 : '૭૪',75 : '૭૫',76 : '૭૬',77 : '૭૭', 78 : '૭૮',
               80 : '૮૦', 81 : '૮૧',82 : '૮૨',83 : '૮૩', 84 : '૮૪', 85 : '૮૫', 86 : '૮૬', 87 :'૮૭', 88 : '૮૮',89 : '૮૯', 90 : '૯૦', 91 :'૯૧',
               94 : '૯૪', 95 : '૯૫', 96 : '૯૬', 97 : '૯૭', 98 : '૯૮', 99 : '૯૯', 100 : 'એ', 101 : 'બા', 102 : 'બિ',103 : 'બી', 104 : 'બુ', 1
               106 : 'બે',107 : 'બૈ',108 : 'બો',109 : 'બૌ',110 : 'બં', 111 : 'બઁ', 112 :'2', 113 :'બઃ',114 : 'ભ',115: 'ભા',116 : 'ભિ', 117
               119 : 'ભૂ', 120 : 'ભે', 121 : 'ભૈ',122: 'ભો', 123: 'ભૌ',124 : 'ભ્',125: 'મં',126 : 'મઁ',127 : 'મ',128 : 'મા',129: 'મિ',130 :
               132 :'મૂ', 133 : 'મે', 134 : 'મૈ', 135 : 'મૉ', 136:  'મો',137 : 'મૌ', 138: 'મ્', 139 :'મઃ',140 : 'ય', 141 : 'યા',142: 'યિ',14
               145 : 'યઃ',146 : 'યૂ', 147: 'યે',148 : 'યૈ', 149 :'યો',150: 'યૌ',151: 'યં',152 : 'યઁ', 153 :'ર',154: 'રા', 155 :'રિ', 156 :'
               158 :'રુ', 159 :'રૂ', 160 : 'રે',161: 'રૈ',162: 'રો',163: 'રૌ',164 : 'ર્',165 : 'રઃ',166: 'લ',167: 'લા',168: 'લિ',169: 'લી',176
               172 :'લૂ',173 : 'લે',174 : 'લૈ',175 : 'લો',176 : 'લૌ',177 : 'લ્', 178 : 'લઁ', 179 : 'લઃ',180 : 'વ', 181 :'વા', 182 : 'વિ', 18
               185 :'વૂ', 186 : 'વે',187 : 'વૈ',188 : 'વો',189 : 'વૌ', 190 : 'વ્',191 : 'વં',192: 'વઃ', 193 : 'શ',194 : 'શા', 195 : 'શિ',
               197 :'શુ', 198 : 'શૂ', 199 :'શે',200 : 'શૈ', 201 :'શો',202 : 'શૌ', 203 : 'શ્',204 : 'શં',205 : 'શઃ',206: 'ષ',207 : 'ષા', 208
               210 : 'ષુ', 211 : 'ષૂ',212 : 'ષૃ',213 : 'ષે',214 : 'ષૈ',215 : 'ષો',216 : 'ષૌ',217 : 'ષ્',218 : 'ષઃ', 219 : 'સ',220 : 'સા',221
               223 : 'સુ', 224 : 'સૂ',225 : 'સૃ',226 : 'સે',227 : 'સૈ',228 : 'સો',229 : 'સૌ',230 : 'સ્',231 : 'સં', 232 : 'સઃ',233 : 'હ',234 :
               236 : 'હિ',237 : 'હી',238 : 'હુ',239 : 'હૂ',240 : 'હે',241 : 'હૈ',242 : 'હો',243 : 'હૌ',244 : 'હ્',245 : 'હં',246 : 'હઃ',247 :
               250 : 'ળા',251 : 'ળિ',252 : 'ળી',253 : 'ળે',254 : 'ળૈ',255 : 'ળો',256 : 'ળૌ',257 : 'ળ્',258: 'ળં',259 : 'ળઃ',260 : 'ક્ષ',261 :
               264 :'જ્ઞ', 265 : 'શ્ર',266 : 'ત્ર',267 : 'જ્',268 :'દ્ય', 269 : 'ત્ત', 270 : 'ત્ત્ય', 271 : 'ત્ય',272: 'ત્ર',273 : 'દ્ય',274 : 'દ્ય', 275
               277 : 'દ્ય',278 : 'જ્ઞ', 279:  'જ્ઞ',280 : 'જ્ઞ',281 : 'દ્ય',282 : 'જ્ઞ',283: 'દ્ય',284: 'જ્ઞ',285 : 'દ્ય',286 : 'ક્ષ', 287 : 'ક્ષ',288:
               290 : 'ખો', 291 : 'ખુ',292 : 'ખૂ',293 : 'ખે',294 : 'ખૈ',295: 'ખો',296 : 'ખૌ',297: 'ખ્',298: 'ખઃ',299 : 'ગ',300: 'ગા',301 : 'ગ
```

Figure 4.1.7

Then, we save the model using model.save().

```
In [24]: from keras.preprocessing import image
         img = image.load_img("gdrive/My Drive/images/img7.png", color_mode = "grayscale", target_size = (28, 28))
         img = image.img_to_array(img)
         img = np.expand_dims(img, axis = 0)
         prediction = model.predict(img.reshape(1,28,28, 1))
         plt.imshow(img.reshape(28,28), cmap="gray")
         plt.show()

         print('Predicted character is:',dict[np.argmax(prediction)])
```



```
Predicted character is: ૯

In [25]: from google.colab import files
         files.download("model6.h5")

         <IPython.core.display.Javascript object>

         <IPython.core.display.Javascript object>
```

Figure 4.1.8

We then predict the image and it is correctly recognized.

4.2 Testing using Use Cases

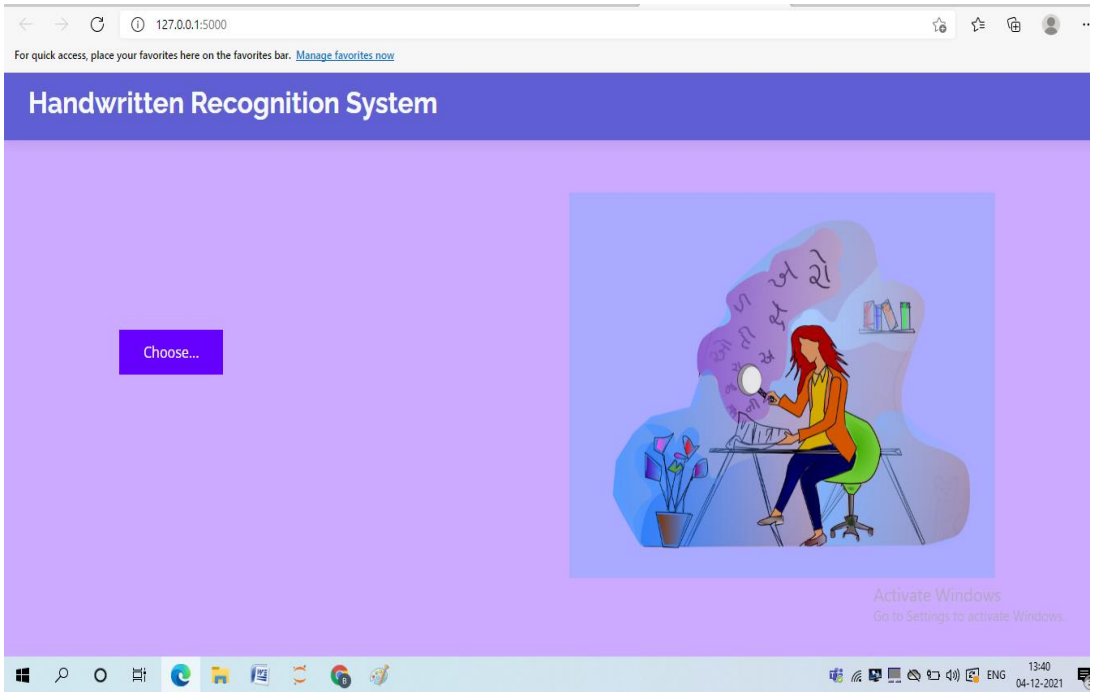


Figure 4.2.1

The above image shows the image of the Handwritten recognition system for gujarati characters. Here, by clicking on choose button you need to select the image from the folder and it is displayed.

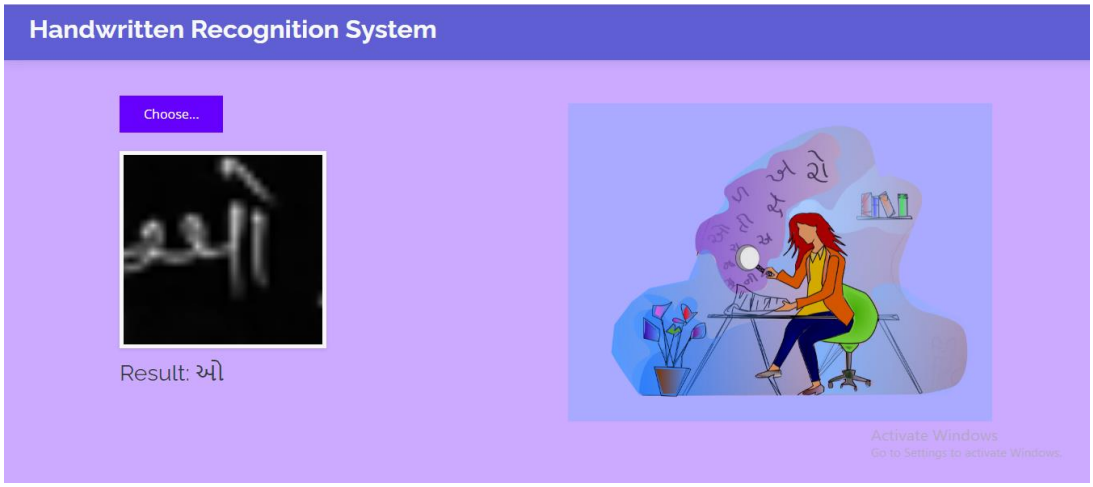


Figure 4.2.2

As shown above, the image is displayed in the website. Here, we test the model using the image from the dataset. It correctly predicted the image of ઓ.

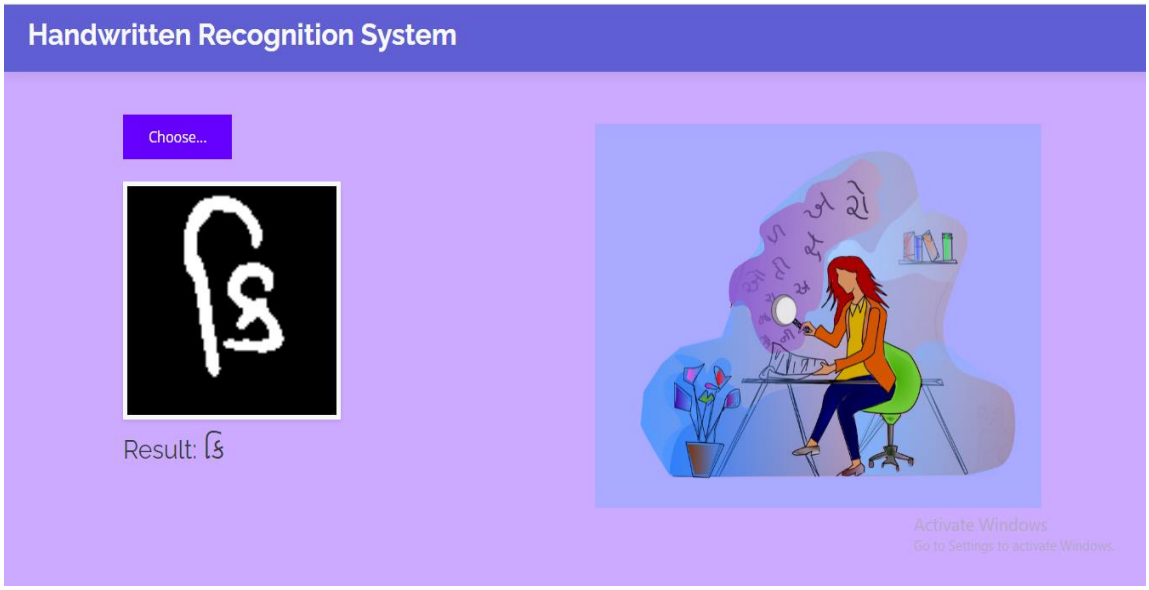


Figure 4.2.3

As shown in above image, here we tested our CNN model for the image outside of the dataset and clicking on predict button we get the correctly recognized result of ઉ.

Chapter 5: Conclusion & Future work

- **Conclusion:**

  We have successfully developed the website for Handwritten recognition system for gujarati characters with Python, Tensorflow, and Machine Learning libraries. Our CNN model get 94.57% of training accuracy and 93.40% test accuracy. Therefore, our model is able to recognize the handwritten gujarati characters with 93.40% test accuracy. The website is able to successfully recognized almost all gujarati characters inside and outside of the dataset.

- **Future work:**

  The handwritten character recognition technique can be further used in word recognization. Word recognization is difficult task in Gujarati Handwritten Words, but first word segmentation is done and after that recognition of one-one character might be possible to achieve whole word recognition. And then this recognition can be done for the whole document written in multiple languages with the further research in this area of topic.

Chapter 6: References

- **Literature References:**

[1] Megha Agarwal, Shalika, Viman Tomar, Priyanka Gupata (2019,April), "HandwrittenCharacter Recognition using Neural Network and Tensor Flow", Vol. 8, pp. 1445-1448
https://www.ijitee.org/wp-content/uploads/papers/v8i6s4/F12940486S419.pdf

[2] I Khandokar et al 2021," Handwritten character recognition using convolutional neural Network", J. Phys.: Conf. Ser. 1918 042152.
https://iopscience.iop.org/article/10.1088/1742-6596/1918/4/042152/pdf

[3] Preetha Sa *, Afrid I M , Karthik Hebbar P , Nishchay S K (2020)," Machine Learning for Handwriting Recognition", Vol. 38(1), pp. 93-101.
https://ijcjournal.org/index.php/InternationalJournalOfComputer/article/view/1637/655

[4] S. Anandh Kishan, J. Clinton David, P.Sharon Femi (2018, September), "Handwritten Character Recognition using CNN",Vol. 5,pp. 241-245.
https://ijrar.org/papers/IJRAR1903931.pdf

[5] Jyoti Pareek, Dimple Singhania, Rashmi Rekha Kumari, Suchit Purohit (2020), "Gujarati Handwritten Character Recognition with Text Images", pp. 514-523. https://www.researchgate.net/publication/341906589_Gujarati_Handwritten_Character_Recognition_from_Text_Images

- **Website References:**

https://www.analyticsvidhya.com/blog/2019/11/4-tricks-improve-deep-learning-model-performance/

https://towardsdatascience.com/how-i-got-1-better-accuracy-by-data-augmentation-2475c509349a

https://data-flair.training/blogs/handwritten-character-recognition-neural-network/

https://analyticsindiamag.com/image-data-augmentation-impacts-performance-of-image-classification-with-codes/