Conversion of HandWritten Text into Text Using Computer Vision

A

PROJECT REPORT

Submitted in partial fulfillment of the

Requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Specialization in

Business Analytics and Optimization

BY

Name	Roll No.	SAPID
ADESH KUMAR GUPTA	R103216006	500053000
SHANKEY GUPTA	R103216089	500054226
TUSHAR SINGH	R103216109	500054191
UTKARSH SANDEEP SINGH	R103216110	500053648

Under the guidance of

Dr. Hitesh Kumar Sharma
AP(SG) and PIC

Department of Informatics

School of Computer Science

UPES

University of Petroleum & Energy Studies

Bidholi, via Prem Nagar, Dehradun, UK

May 2020

CANDIDATES DECLARATION

We hereby certify that the project work entitled **Conversion of Hand Written Text Into Text Using Computer Vision** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering with Specialization in Business Analytics and Optimization and submitted to the Department of Informatics at School of Computer Science, University of Petroleum And Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **January 2020 to April 2020** under the supervision of **Dr. Hitesh Kumar Sharma**, **AP & PIC.**

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

(Name of Student(s))

Adesh, Shankey, Tushar, Utkarsh

Roll No.

R103216006, R103216089,

R103216109, R103216110

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Date: 19 April 2020)

Dr. Hitesh Kumar Sharma
Project Guide

Dr. T.P Singh

Head
Department of Informatics
School of Computer Science
University of Petroleum And Energy Studies
Dehradun - 248 001 (Uttarakhand)

ACKNOWLEDGMENT

We wish to express our deep gratitude to our guide Name, for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank our Head of the Department, Dr. T.P Singh, for his great support in doing our project name at SoCS.

We are also grateful to Dr. Manish Prateek Professor and Director SoCS, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our friends for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our parents who have shown us this world and for every support, they have given us.

Name	Adesh	Shankey	Tushar	Utkarsh
Roll No.	R103216006	R103216089	R103216109	R103216110

Abstract

This application is useful for recognizing all alphabets (English) given as in the input image. Once the input image of the character is given to the proposed system, then it will recognize the input character which is given in the image. 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.

This model will be helpful to avoid manual conversion of handwritten text to the digitized text which is a tough job to be done, so the model overcomes this problem and makes it much easier to identify a particular handwritten character and convert it to its corresponding digitized form. The model takes a handwritten character as an input and processes it and provides the output in the form of its corresponding digitized text on the screen. The model will be based on training and testing data set where it will be trained for each and every set of characters in order to predict the correct outcome and after successful training will be tested for a set of input data to check for accuracy. It can be beneficial in making the search easy for some important information.

Keywords: Handwritten Recognition, Character Recognition, Feature Extraction, Neural Network

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction:

Deep learning is a topic that is making big waves at the moment. It is basically a branch of machine learning (another hot topic) that uses algorithms to e.g. recognize objects and understand human speech. Scientists have used deep learning algorithms with multiple processing layers (hence "deep") to make better models from large quantities of unlabeled data (such as photos with no description, voice recordings or videos on YouTube).

It's one kind of supervised machine learning, in which a computer is provided a training set of examples to learn a function, where each example is a pair of an input and an output from the function.

A great way to use deep learning to classify images is to build a convolutional neural network (CNN). In our scenario, we are developing CNN to classify the English alphabets from images.

A Convolutional Neural Network (CNN) consists of one or more convolutional layers and then followed by one or more fully connected layers as in a standard multilayer neural network. Convolution is a specialized kind of linear operation. The architecture of a CNN is designed to take advantage of the 2D structure of an input image. This is achieved with local connections and tied weights followed by some form of pooling which results in translation-invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

In our real world, **Facebook** has introduced a deep learning system named **Rosetta for scalable optical character recognition (OCR)**. This model extracts text from more than a billion public Facebook and Instagram images and video frames. Then, this extracted text is fed into a text recognition model that has been trained on classifiers, which helps it understand the context of the text and the image together.

Another real-world example is **Google's Web-Based OCR Service Tesseract** which is one of the famous open-source libraries that everyone can leverage to execute OCR. Tesseract is found by HP and development has been sponsored by Google since 2006. The Tesseract 3.x model is an old version while the 4.x version is built by deep learning (LSTM). It has the ability to recognize more than 100 languages out of the box. Also, It can be trained to recognize other languages as well.

2. History:

Deep learning is based on the concept of artificial neural networks, or computational systems that mimic the way the human brain functions. And so, our brief history of deep learning must start with those neural networks.

1943: Warren McCulloch and Walter Pitts create a computational model for neural networks based on mathematics and algorithms called threshold logic.

1958: Frank Rosenblatt creates the perceptron, an algorithm for pattern recognition based on a two-layer computer neural network using simple addition and subtraction. He also proposed additional layers with mathematical notations, but these wouldn't be realized until 1975.

1980: Kunihiko Fukushima proposes the Neoconitron, a hierarchical, multilayered artificial neural network that has been used for handwriting recognition and other pattern recognition problems.

1989: Scientists were able to create algorithms that used deep neural networks, but training times for the systems were measured in days, making them impractical for real-world use.

1992: Juyang Weng publishes Cresceptron, a method for performing 3-D object recognition automatically from cluttered scenes.

Mid-2000s: The term "deep learning" began to gain popularity after a paper by Geoffrey Hinton and Ruslan Salakhutdinov showed how a many-layered neural network could be a pre-trained one layer at a time.

2009: NIPS Workshop on Deep Learning for Speech Recognition discovers that with a large enough data set, the neural networks don't need pre-training, and the error rates drop significantly.

2012: Artificial pattern-recognition algorithms achieve human-level performance on certain tasks. And Google's deep learning algorithm discovers cats.

2014: Google buys UK artificial intelligence startup Deepmind for £400m

2015: Facebook puts deep learning technology - called DeepFace - into operations to automatically tag and identify Facebook users in photographs. Algorithms perform superior face recognition tasks using deep networks that take into account 120 million parameters.

2016: Google DeepMind's algorithm AlphaGo masters the art of the complex board game Go and beats the professional Go player Lee Sedol at a highly publicized tournament in Seoul.

The promise of deep learning is not that computers will start to think like humans. That's a bit like asking an apple to become an orange. Rather, it demonstrates that given a large enough data set, fast enough processors, and a sophisticated enough algorithm, computers can begin to accomplish tasks that used to be completely left in the realm of human perception — like recognizing cat videos on the web (and other, perhaps more useful purposes).

3. Literature Review:

In this paper, [1] the whole framework based on the recognition of offline handwriting digits. This disconnected penmanship number acknowledgment framework has primarily five phases: Preprocessing, Segmentation, Feature Extraction, Classification, and Postprocessing.

The principle point of the proposed work is to effectively perceive the disconnected manually written digits with a higher exactness than past works done in this field. In any case, a troublesome issue in this field is the acknowledgment of somewhat or totally contacting written by hand digits and the current apparatuses are not yet for such conditions. Additionally past transcribed number acknowledgment frameworks depend on just perceiving single digits and they are not equipped for perceiving numerous numbers one after another. So the significant worry in the proposed work is centered around effectively performing segmentation for disconnecting the digits with the goal that various number pictures can be perceived in one stage and improving the exactness of the framework and to make a superior manually written number acknowledgment framework.

Programmed recognition [2] of manually written characters is a troublesome assignment since characters are written in different bent and cursive ways, so they could be of various sizes, direction, thickness, arrangement, and measurement. A disconnected manually written Hindi character recognition framework utilizing a neural network is introduced in this paper. Neural networks are acceptable at perceiving manually written characters as these networks are coldhearted toward the missing information. The paper proposes the way to deal with perceive Hindi characters in four phases—1) Scanning, 2) Preprocessing, 3) Feature Extraction and, 4) Recognition. Preprocessing incorporates clamor decrease, binarization, standardization, and diminishing. Feature extraction incorporates extricating some valuable data out of the diminished picture as a feature vector. The feature vector includes pixels estimations of the standardized character picture. A Backpropagation neural network is utilized for the order. The trial result shows that this methodology gives better outcomes when contrasted with different strategies as far as recognition precision, preparing time and grouping time. The normal exactness of recognition of the framework is 93%.

A neural network [3] approach is proposed to construct a programmed disconnected character recognition framework. Devnagari is an Indo-Aryan language spoken by around 71 million individuals primarily in the Indian province of Maharashtra and neighboring states. One may discover such a great amount of work for Indian dialects like Hindi, Kannada, Tamil, Bangala, Malayalam and so forth yet Devanagari is a language for which scarcely any work is recognizable particularly for character recognition. In this paper, work has been performed to perceive Devnagari characters utilizing a multilayer perceptron with the concealed layer. Different examples of characters are made in the lattice (n*n) with the utilization of a twofold structure and put away in the record. We have utilized the back proliferation neural network for effective recognition and redressed neuron esteems were transmitted by the feed-forward technique in the neural network.

Character recognition [4] is one of the most intriguing and testing research regions in the field of Image handling. These days various procedures are in common use for character recognition. Record confirmation, advanced library, perusing bank store slips, perusing postal locations, extricating data from checks, information section, applications for charge cards, medical coverage, advances, tax documents and so on are application territories of computerized report preparing. This paper gives a summation of research work completed for recognition of written by hand characters. In-Hand composed content there is no limitation on the composing style. Manually written characters are trying to perceive because of various human penmanship styles, variety in edge, size, and state of letters. Different perspectives on manually written character recognition are talked about here alongside their exhibition.

The principle point [5] of this venture is to structure a master framework for, "HCR(English) utilizing Neural Network". that can adequately perceive a specific character of type group utilizing the Artificial Neural Network approach. Neural processing Is a relatively new field, and plan parts are along these lines less all-around determined than those of different structures. Neural PCs execute information parallelism. Neural PC are worked in a way that is totally not quite the same as the activity of ordinary PCs. Neural PC is prepared (not Programmed) with the goal that given a specific beginning state (information input); they either arrange the info information into one of the quantities of classes or cause the first information to develop so that a specific alluring property is advanced.

4. Requirements:

- **Hardware Requirements –**
  Processor – Core i7 2.4 GHz
  Ram – 8 GB
- **Software Requirements –**
  Operating system – Windows 10
  Python version 3.6
  Libraries: Numpy, Pandas, Keras and Tensorflow, OpenCV.

5. Objectives:

Here we are delivering an end-to-end tool that will be used to identify the characters of text given by the end-user and help him/her convert it into text.

1. To construct a suitable neural network and train it properly to recognize characters.
2. Reduce man-power to convert text into digitized form manually.

6. Pert Chart

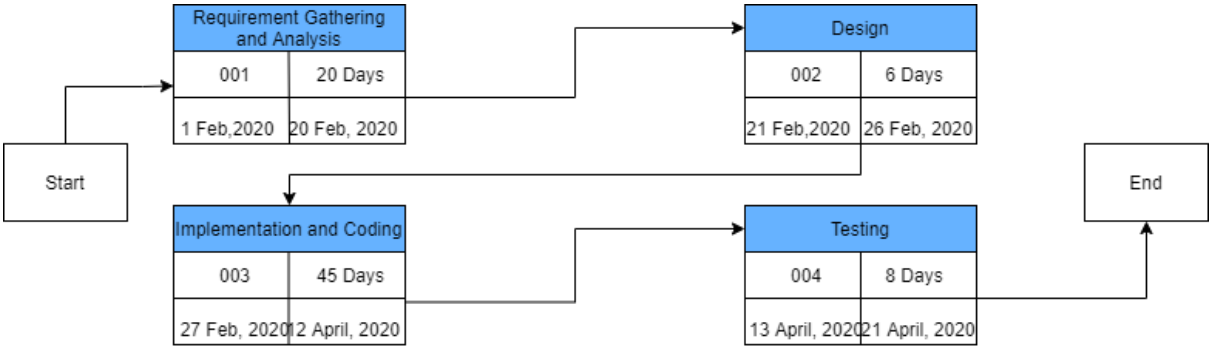

Figure 6.1: Pert Chart

7. System Analysis

- **Existing System**
  Existing handwriting recognition systems use complex architecture which is not beginner-friendly and is modified in a way such that they can be used with only their specific software.

- **Motivation**
  The motivation behind the project was to develop an easy to understand the handwriting recognition system which is easy to use and understand and is compatible in all environments.

- **Proposed System**
  The system will be developed in the python language which is the go-to language when it comes to ML/DL application as it provides an ample number of libraries to develop an end to the end product.

- **Modules**
  - ➔ **Gathering Data: -**
    In this module, we will be gathering the data and store them in files. There will be 26 classes each class will have its own separate folder. This will be created as labeled data for further manipulation.

  - ➔ **Data Preparation: -**
    In this module, we will use the data gathered before and convert each image into a (64x64) grayscale image. Each image will have a label corresponding to it obtained through its data classification.

  - ➔ **Training and Development: -**
    In this module, we will be using CNN technique which has been used for classification.

  - ➔ **Testing and Improvements: -**
    In this module, we will try and improve our model by hyperparameter tuning and changing the architecture of the neural network.

8. Design:

- **Use Case Diagram**

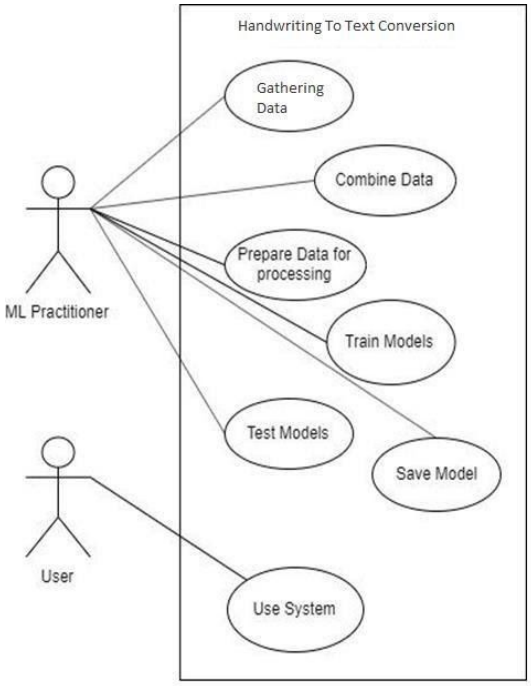

Figure 8.1: Use Case Diagram

- **Flow Chart**

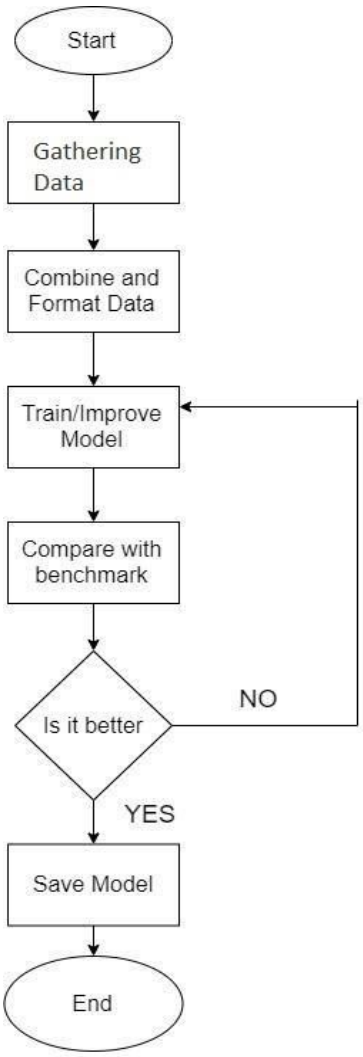

Figure 8.2: Flow Chart

- **Data Flow Diagrams**

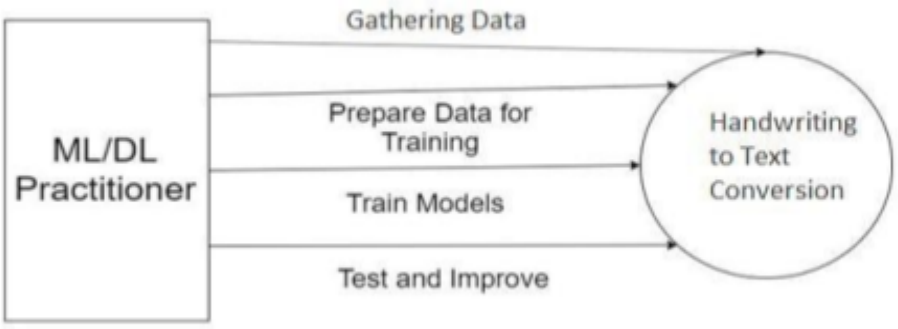

Figure 8.3: Data Flow Diagram

9. Implementation:

- **Interaction**

  The user will interact with the application by giving the input character. The application will then use the model which is stored in the database and use it to recognize the character. The recognized character will be converted to plain text.

- **Architecture**

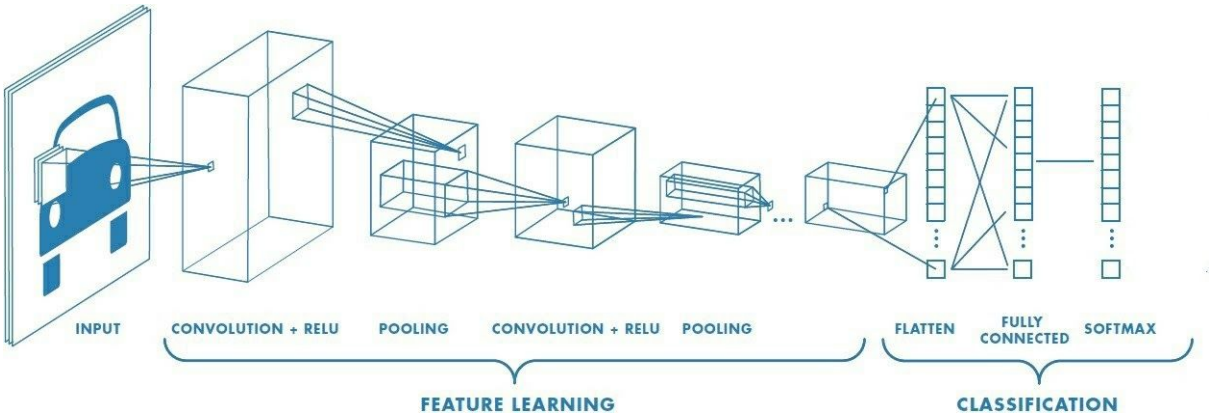

Figure 9.1: Neural network with many convolutional layers.

➢ **Convolution layer:**

The convolution layer is the main building block of a convolutional neural network. The convolution layer comprises of a set of independent filters. We take the filter and slide it over the complete image and along the way take the dot product between the filter and chunks of the input image.

➢ **Pooling layer:**

Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network. The pooling layer operates on each feature map independently. The most common approach used in pooling is max pooling.

➢ **Flatten:**

In between the convolutional layer and the fully connected layer, there is a 'Flatten' layer. Flattening transforms a two-dimensional matrix of features into a 1D vector that can be fed into a fully connected neural network classifier.

➢ **Output layer(FC):**

The convolution and pooling layers would only be able to extract features and reduce the number of parameters from the original images. However, to generate the final output we need to apply a fully connected (FC) layer to generate an output equal to the number of classes we need. Neurons in a fully

connected layer have full connections to all activations in the previous layer, as in regular Neural Networks and work in a similar way.

- **Algorithms**

➢ **CNN**

1. define get_image_define size() to get the image shape and size
2. define get_num_of_classes() to store the number of gestures
3. define cnn_model()
   - 3.1) get the number of classes
   - 3.2) model.add(conv2d(input layer, filters, kernel size, activation function, name))
   - 3.3) model.add (maxpooling2D(poolsize ,strides))
   - 3.4) model.add (conv2d(filters, kernel size, activation function))
   - 3.5) model.add (maxpooling2D(poolsize ,strides))
   - 3.6) model.add (flatten())
   - 3.7) model.add(Dense(128, activation_function=relu))
   - 3.8) model.add(Dropout_function())
   - 3.9) model.add(Dense(num_of_classes, activation='softmax'))
   - 3.10) model.compile(loss='categorical_crossentropy',optimizer,metrics=accuracy)
   - 3.11) callbacks_list ([ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True, mode='max')])
   - 3.12) plot model
   - 3.13) return model, callbacks_list
4. Model Fitting
   - 4.1) model.fit(train_images, train_labels, validation_data=(val_images, val_labels), epochs=3, batch_size=32, callbacks=callbacks_list)
   - 4.2) scores = model.evaluate(val_images, val_labels, verbose=1)
   - 4.3) print(CNN Error)
5. call train()

- **Results**

CNN Score: 89% (after 10 epochs).

```
Train on 7280 samples, validate on 3120 samples
Epoch 1/10
7280/7280 [==============================] - 2s 241us/step - loss: 7.1738 - acc: 0.1647 - val_loss: 3.3952 - val_acc: 0.1939
Epoch 2/10
7280/7280 [==============================] - 2s 226us/step - loss: 2.6953 - acc: 0.2854 - val_loss: 2.0900 - val_acc: 0.4625
Epoch 3/10
7280/7280 [==============================] - 1s 196us/step - loss: 1.8148 - acc: 0.5207 - val_loss: 1.4886 - val_acc: 0.6183
Epoch 4/10
7280/7280 [==============================] - 2s 224us/step - loss: 1.2996 - acc: 0.6713 - val_loss: 1.0711 - val_acc: 0.7186
Epoch 5/10
7280/7280 [==============================] - 2s 237us/step - loss: 0.9494 - acc: 0.7515 - val_loss: 0.8241 - val_acc: 0.7638
Epoch 6/10
7280/7280 [==============================] - 2s 236us/step - loss: 0.7231 - acc: 0.8051 - val_loss: 0.6676 - val_acc: 0.8093
Epoch 7/10
7280/7280 [==============================] - 2s 210us/step - loss: 0.5834 - acc: 0.8408 - val_loss: 0.5717 - val_acc: 0.8423
Epoch 8/10
7280/7280 [==============================] - 2s 247us/step - loss: 0.4919 - acc: 0.8663 - val_loss: 0.5159 - val_acc: 0.8532
Epoch 9/10
7280/7280 [==============================] - 2s 237us/step - loss: 0.4296 - acc: 0.8853 - val_loss: 0.4711 - val_acc: 0.8663
Epoch 10/10
7280/7280 [==============================] - 2s 237us/step - loss: 0.3794 - acc: 0.8993 - val_loss: 0.4361 - val_acc: 0.8772
```

10.Output Screen

- **Data Images**

Figure 10.1: Data Images

- **Model Loss**

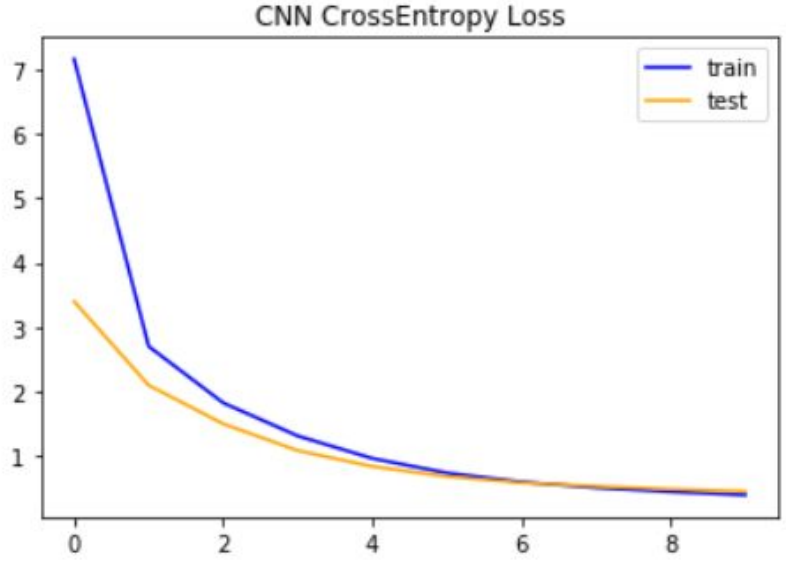

Figure10.2: Model Loss

- **Model Accuracy**

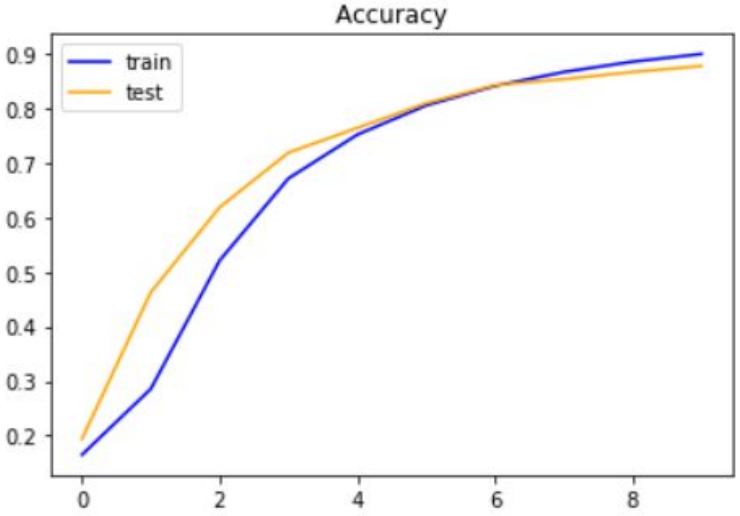

Figure 10.3: Model Accuracy

- **Character Recognition**

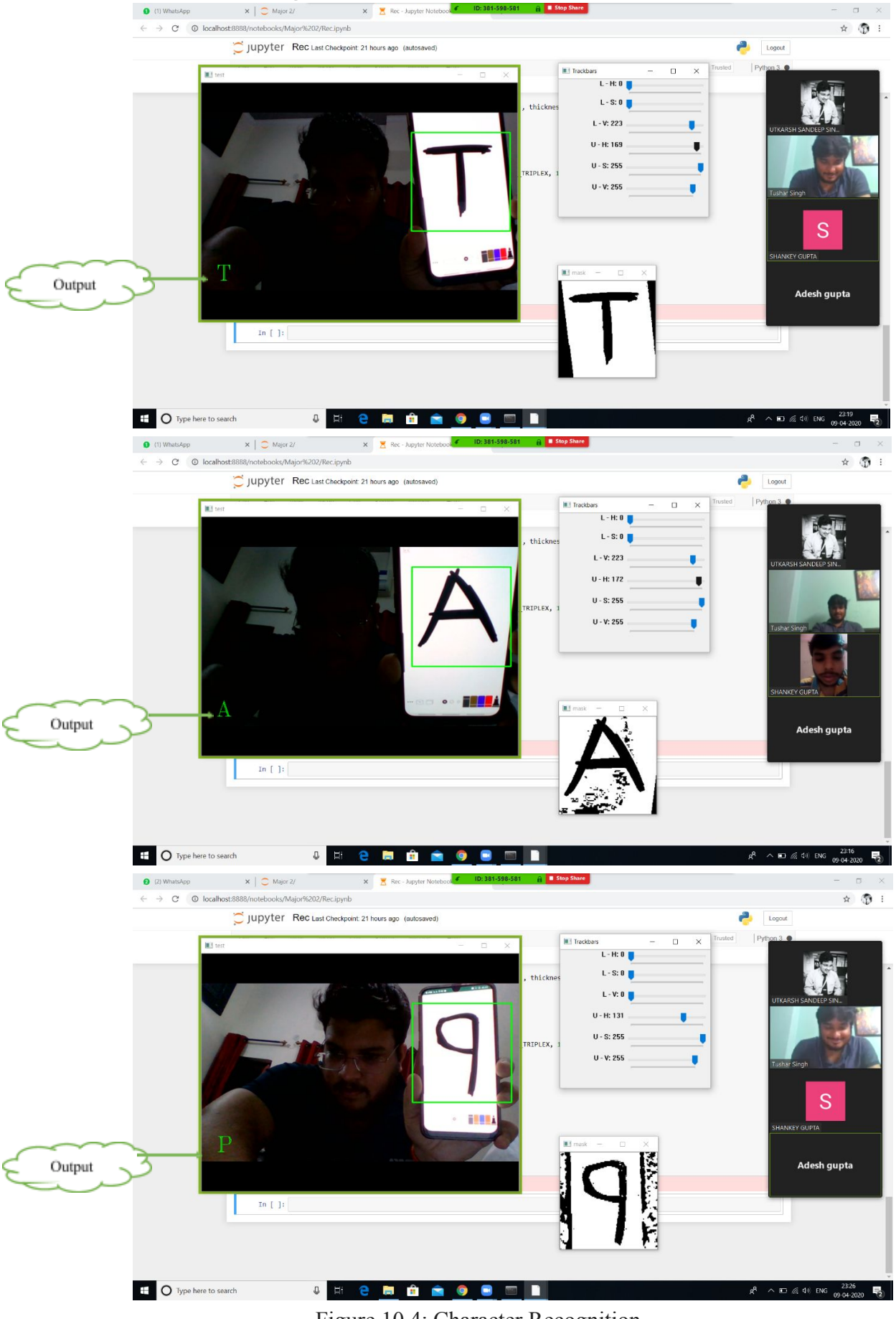

Figure 10.4: Character Recognition

11.References

[1] Isha Vats, Shamandeep Singh, *"Offline Handwritten English Numerals Recognition using Correlation Method"*, International Journal of Engineering Research and Technology (IJERT): ISSN: 2278-0181 Vol. 3 Issue 6, June 2014. Access Date: 09/07/2015.

[2] Gunjan Singh, Sushma Lehri, *" Recognition of Handwritten Hindi Characters using Backpropagation Neural Network"*, International Journal of Computer Science and Information Technologies ISSN 0975-9646, Vol. 3 (4) , 2012,4892-4895. Access Date:09/07/2015.

[3] 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, July 2013. Access Date: 09/07/2015.

[4] Shabana Mehfuz,Gauri katiyar, *'Intelligent Systems for Off-Line Handwritten Character Recognition: A Review"* ,International Journal of Emerging Technology and Advanced Engineering Volume 2, Issue 4, April 2012. Access Date: 09/07/2015.

[5] Prof. Swapna Borde, Ms. Ekta Shah, Ms. Priti Rawat, Ms. Vinaya Patil, *"Fuzzy Based Handwritten Character Recognition System"* ,International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622,VNCET 30 Mar'12. Access Date: 09/07/2015.

Appendix A Project Code

Dataset Generation (Capture File)

```
import cv2
import time
import numpy as np
import os
def nothing(x):
    pass
image_x, image_y = 64, 64
def create_folder(folder_name):
    if not os.path.exists('C:/Users/utkar_znfre1c/Major 2/Images/mydata/training_set/' +
folder_name):
        os.mkdir('C:/Users/utkar_znfre1c/Major 2/Images/mydata/training_set/' +
folder_name)
    if not os.path.exists('C:/Users/utkar_znfre1c/Major 2/Images/mydata/test_set/' +
folder_name):
        os.mkdir('C:/Users/utkar_znfre1c/Major 2/Images/mydata/test_set/' + folder_name)
def capture_images(ges_name):
    create_folder(str(ges_name))
    cam = cv2.VideoCapture(0)
    cv2.namedWindow("test")
    img_counter = 0
    t_counter = 1
    training_set_image_name = 1
    test_set_image_name = 1
    listImage = [1,2,3,4,5]
    cv2.namedWindow("Trackbars")
    cv2.createTrackbar("L - H", "Trackbars", 0, 179, nothing)
    cv2.createTrackbar("L - S", "Trackbars", 0, 255, nothing)
    cv2.createTrackbar("L - V", "Trackbars", 0, 255, nothing)
    cv2.createTrackbar("U - H", "Trackbars", 179, 179, nothing)
    cv2.createTrackbar("U - S", "Trackbars", 255, 255, nothing)
    cv2.createTrackbar("U - V", "Trackbars", 255, 255, nothing)
    for loop in listImage:
        while True:
            ret, frame = cam.read()
            frame = cv2.flip(frame, 1)
            l_h = cv2.getTrackbarPos("L - H", "Trackbars")
            l_s = cv2.getTrackbarPos("L - S", "Trackbars")
            l_v = cv2.getTrackbarPos("L - V", "Trackbars")
            u_h = cv2.getTrackbarPos("U - H", "Trackbars")
            u_s = cv2.getTrackbarPos("U - S", "Trackbars")
            u_v = cv2.getTrackbarPos("U - V", "Trackbars")
            """l_h = 0
            l_s = 40
            l_v = 0
            u_h = 179
            u_s = 225
            u_v = 225"""
```

```
            img = cv2.rectangle(frame, (425, 100), (625, 300), (0, 255, 0), thickness=2,
    lineType=8, shift=0)
            lower_blue = np.array([l_h, l_s, l_v])
            upper_blue = np.array([u_h, u_s, u_v])
            imcrop = img[102:298, 427:623]
            hsv = cv2.cvtColor(imcrop, cv2.COLOR_BGR2HSV)
            mask = cv2.inRange(hsv, lower_blue, upper_blue)
            result = cv2.bitwise_and(imcrop, imcrop, mask=mask)
            cv2.putText(frame, str(img_counter), (30, 400),
    cv2.FONT_HERSHEY_TRIPLEX, 1.5, (127, 127, 255))
            cv2.imshow("test", frame)
            cv2.imshow("mask", mask)
            cv2.imshow("result", result)
            if cv2.waitKey(1) == ord('c'):
                if t_counter <= 350:
                    img_name = "C:/Users/utkar_znfre1c/Major 2/Images/mydata/training_set/"
    + str(ges_name) + "/{}.png".format(training_set_image_name)
                    save_img = cv2.resize(mask, (image_x, image_y))
                    cv2.imwrite(img_name, save_img)
                    print("{} written!".format(img_name))
                    training_set_image_name += 1
                if t_counter > 350 and t_counter <= 400:
                    img_name = "C:/Users/utkar_znfre1c/Major 2/Images/mydata/test_set/" +
    str(ges_name) + "/{}.png".format(test_set_image_name)
                    save_img = cv2.resize(mask, (image_x, image_y))
                    cv2.imwrite(img_name, save_img)
                    print("{} written!".format(img_name))
                    test_set_image_name += 1
                    if test_set_image_name > 250:
                        break
                t_counter += 1
                if t_counter == 401:
                    t_counter = 1
                img_counter += 1
            #elif cv2.waitKey(1) == 27:
                #break
            elif(cv2.waitKey(1) & 0xFF == ord('q')):
                break
        if test_set_image_name > 250:
            break
    cam.release()
    cv2.destroyAllWindows()
ges_name = input("Enter gesture name: ")
capture_images(ges_name)
```

Training Model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
```

```
import keras
from keras.models import Sequential
from keras.layers import Conv2D,MaxPool2D,Dense,Flatten
from keras.utils import to_categorical
from keras.optimizers import Adam,RMSprop
import pickle
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import auc, roc_auc_score, roc_curve
from sklearn.model_selection import cross_val_score,StratifiedKFold, train_test_split
import cv2
from PIL import Image
import warnings
warnings.filterwarnings("ignore")
train_df = pd.read_pickle("Train.pkl")
test_df  = pd.read_pickle("Test.pkl")
feature_columns = [i for i in range(784)]
trainX = train_df[feature_columns]
trainY = train_df['label']
testX  = test_df[feature_columns]
testY  = test_df['label']
print(trainX.shape, trainY.shape)
print(testX.shape, testY.shape)
trainX = np.array(trainX).reshape((trainX.shape[0], 28, 28, 1))
testX = np.array(testX).reshape((testX.shape[0], 28, 28, 1))
print(trainX.shape, trainY.shape)
print(testX.shape, testY.shape)
trainY = to_categorical(trainY)
testY = to_categorical(testY)
print(trainX.shape, trainY.shape)
print(testX.shape, testY.shape)
# Scaling:
trainX = trainX/255.0
testX = testX/255.0
img = trainX[0]
print(img.shape)
plt.imshow(img[:,:,0], cmap='gray')
plt.show()
np.save('TrainX',trainX)
np.save('TestX',testX)
np.save('trainY',trainY)
np.save('testY',testY)
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',
input_shape=(28, 28, 1) ))
model.add(MaxPool2D((2,2)))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(26, activation='softmax'))
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
print(model.summary())
history = model.fit(trainX, trainY, epochs=10, batch_size=512, validation_data=(testX,
testY), verbose=1)
plt.title(" CNN CrossEntropy Loss ")
plt.plot(history.history['loss'], color='blue', label='train')
plt.plot(history.history['val_loss'], color='orange', label='test')
plt.legend()
plt.show()
plt.title(" Accuracy ")
plt.plot(history.history['acc'], color='blue', label='train')
plt.plot(history.history['val_acc'], color='orange', label='test')
plt.legend()
plt.show()
model.save('CNNSG13.h5')
```

Testing Model

```
from keras.models import load_model
classifier = load_model('Trained_model.h5')
classifier.evaluate()
#Prediction of single image
import numpy as np
from keras.preprocessing import image
img_name = input('Enter Image Name: ')
image_path = './predicting_data/{}'.format(img_name)
print('')
test_image = image.load_img(image_path, target_size=(200, 200))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = classifier.predict(test_image)
#training_set.class_indices
print('Predicted Sign is:')
print('')
if result[0][0] == 1:
    return 'A'
elif result[0][1] == 1:
    return 'B'
elif result[0][2] == 1:
    return 'C'
elif result[0][3] == 1:
    return 'D'
elif result[0][4] == 1:
    return 'E'
elif result[0][5] == 1:
    return 'F'
elif result[0][6] == 1:
    return 'G'
elif result[0][7] == 1:
    return 'H'
elif result[0][8] == 1:
    return 'I'
```

```
    elif result[0][9] == 1:
        return 'J'
    elif result[0][10] == 1:
        return 'K'
    elif result[0][11] == 1:
        return 'L'
    elif result[0][12] == 1:
        return 'M'
    elif result[0][13] == 1:
        return 'N'
    elif result[0][14] == 1:
        return 'O'
    elif result[0][15] == 1:
        return 'P'
    elif result[0][16] == 1:
        return 'Q'
    elif result[0][17] == 1:
        return 'R'
    elif result[0][18] == 1:
        return 'S'
    elif result[0][19] == 1:
        return 'T'
    elif result[0][20] == 1:
        return 'U'
    elif result[0][21] == 1:
        return 'V'
    elif result[0][22] == 1:
        return 'W'
    elif result[0][23] == 1:
        return 'X'
    elif result[0][24] == 1:
        return 'Y'
    elif result[0][25] == 1:
        return 'Z'
```

Realtime Monitoring

```
import cv2
import numpy as np
#import pyttsx3
def nothing(x):
   pass
image_x, image_y = 64,64
from keras.models import load_model
classifier = load_model('Trained_model.h5')
def predictor():
       import numpy as np
       from keras.preprocessing import image
       test_image = image.load_img('1.png', target_size=(64, 64))
       test_image = image.img_to_array(test_image)
       test_image = np.expand_dims(test_image, axis = 0)
       result = classifier.predict(test_image)
```

```
if result[0][0] == 1:
        return 'A'
elif result[0][1] == 1:
        return 'B'
elif result[0][2] == 1:
        return 'C'
elif result[0][3] == 1:
        return 'D'
elif result[0][4] == 1:
        return 'E'
elif result[0][5] == 1:
        return 'F'
elif result[0][6] == 1:
        return 'G'
elif result[0][7] == 1:
        return 'H'
elif result[0][8] == 1:
        return 'I'
elif result[0][9] == 1:
        return 'J'
elif result[0][10] == 1:
        return 'K'
elif result[0][11] == 1:
        return 'L'
elif result[0][12] == 1:
        return 'M'
elif result[0][13] == 1:
        return 'N'
elif result[0][14] == 1:
        return 'O'
elif result[0][15] == 1:
        return 'P'
elif result[0][16] == 1:
        return 'Q'
elif result[0][17] == 1:
        return 'R'
elif result[0][18] == 1:
        return 'S'
elif result[0][19] == 1:
        return 'T'
elif result[0][20] == 1:
        return 'U'
elif result[0][21] == 1:
        return 'V'
elif result[0][22] == 1:
        return 'W'
elif result[0][23] == 1:
        return 'X'
elif result[0][24] == 1:
        return 'Y'
elif result[0][25] == 1:
```

```
            return 'Z'
cam = cv2.VideoCapture(0)
cv2.namedWindow("Trackbars")
cv2.createTrackbar("L - H", "Trackbars", 0, 179, nothing)
cv2.createTrackbar("L - S", "Trackbars", 0, 255, nothing)
cv2.createTrackbar("L - V", "Trackbars", 0, 255, nothing)
cv2.createTrackbar("U - H", "Trackbars", 179, 179, nothing)
cv2.createTrackbar("U - S", "Trackbars", 255, 255, nothing)
cv2.createTrackbar("U - V", "Trackbars", 255, 255, nothing)
cv2.namedWindow("test")
img_counter = 0
img_text = ''
while True:
    ret, frame = cam.read()
    frame = cv2.flip(frame,1)
    l_h = cv2.getTrackbarPos("L - H", "Trackbars")
    l_s = cv2.getTrackbarPos("L - S", "Trackbars")
    l_v = cv2.getTrackbarPos("L - V", "Trackbars")
    u_h = cv2.getTrackbarPos("U - H", "Trackbars")
    u_s = cv2.getTrackbarPos("U - S", "Trackbars")
    u_v = cv2.getTrackbarPos("U - V", "Trackbars")
    """l_h = 0
    l_s = 40
    l_v = 0
    u_h = 179
    u_s = 255
    u_v = 255"""
    img = cv2.rectangle(frame, (425,100),(625,300), (0,255,0), thickness=2, lineType=8,
shift=0)
    lower_blue = np.array([l_h, l_s, l_v])
    upper_blue = np.array([u_h, u_s, u_v])
    imcrop = img[102:298, 427:623]
    hsv = cv2.cvtColor(imcrop, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, lower_blue, upper_blue)
    cv2.putText(frame, img_text, (30, 400), cv2.FONT_HERSHEY_TRIPLEX, 1.5, (0,
255, 0))
    cv2.imshow("test", frame)
    cv2.imshow("mask", mask)
    img_name = "1.png"
    save_img = cv2.resize(mask, (image_x, image_y))
    cv2.imwrite(img_name, save_img)
    #print("{} written!".format(img_name))
    img_text = predictor()
    if(cv2.waitKey(1) & 0xFF == ord('q')):
        break
cam.release()
cv2.destroyAllWindows()
```