A Course Based Project Report on

Handwriting Character Recognition using Machine Learning

Submitted to the

Department of Information Technology

in partial fulfilment of the requirements for the completion of course

Machine Learning Lab (19PC2CS04)

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by

Akhil Nalla (19071A12C1)

Akshay Nedunuri (19071A12C2)

Araveeti Sai Shreya (19071A12C3)

Aturi Jabilli (19071A12C4)

Bathini Vyooha (19071A12C5)

DEPARTMENT OF INFORMATION TECHNOLOGY

VNR Vignana Jyothi Institute of Engineering & Technology

(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA)

Bachupally, Nizampet (S.O.) Hyderabad- 500 090

VNR Vignana Jyothi Institute of Engineering & Technology
Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA) Bachupally,
Nizampet (S.O.) Hyderabad- 500 090

Department of Information Technology

CERTIFICATE

This is to certify that the course based project work entitled **"Handwriting Character Recognition using Machine Learning"** is being submitted by Akhil Nalla (19071A12C1), Akshay Nedunuri (19071A12C2), Araveeti Sai Shreya (19071A12C3), Aturi Jabilli (19071A12C4), Bathini Vyooha (19071A12C5) in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **INFORMATION TECHNOLOGY** to the Department of Information Technology, Hyderabad during the academic year 2021-22 is a record of bonafide work carried out by them under our guidance and supervision.

The results embodied in this report have not been submitted by the students to any other University or Institution for the award of any degree or diploma.

Project Guide
Mrs. B. Mathurabai
Assistant Professor,
IT Department,
VNRVJIET,
Hyderabad.

Head of Department
Dr. D. Srinivasa Rao,
HOD,
IT Department,
VNRVJIET,
Hyderabad.

VNR Vignana Jyothi Institute of Engineering & Technology
Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA) Bachupally,
Nizampet (S.O.) Hyderabad- 500090.

Department of Information Technology

DECLARATION

I hereby declare that the course-based project entitled "**Handwriting Character Recognition**" submitted for the year III semester II as part of B. Tech Degree in IT, is my original work and the project has not formed the basis for the award of any degree, associate ship, fellowship, or any other similar titles.

Signature of the Student:

Akhil Nalla (19071A12C1)	Akshay Nedunuri (19071A12C2)	Araveeti Sai Shreya (19071A12C3)	Aturi Jabilli (19071A12C4)	Bathini Vyooha (19071A12C5)

Place: Hyderabad

Date: 14th July 2022

ACKNOWLEDGEMENT

We express our deep gratitude to our beloved President, Dr.D.Suresh Babu, VNR Vignana Jyothi Institute of Engineering and Technology for the valuable guidance and for permitting us to carry out for this project.

With immense pleasure, we record our deep sense of gratitude to our beloved Principal, Dr.C.D.Naidu for permitting us to carry out for this project.

We express our deep sense of gratitude to our beloved professor **Dr. D. Srinivasa Rao, Associate Professor and Head, Department of Information Technology, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad-90** for the valuable guidance and suggestions, keen interest, and encouragement extended throughout the project work.

We take immense pleasure to express our deep sense of gratitude to our beloved Guide **Mrs. B. Mathurabai, and Mrs.B.Srivani, Assistant Professor in Information Technology, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad,** for her valuable suggestions and rare insights, for a constant source of encouragement and inspiration throughout my project work.

We express our thanks to all those who contributed to the successful completion of our project work.

ABSTRACT

Handwritten character recognition has been one of the active and challenging research areas in the field of image processing and pattern recognition. It has numerous advantages such as reading aid for bank cheques, recognizing character from form applications etc. An attempt is made to recognize handwritten characters for English alphabets using CNN. EMNIST dataset which consists of English alphabets and numbers are made use of to train the neural network.

EMNIST balanced dataset consist of 131,600images of characters and 47 classes. The feature extraction technique is obtained by normalizing the pixel values. Pixel values will range from 0 to 255 which represents the intensity of each pixel in the image, and they are normalized to represent value between 0 and 1. Convolutional neural network is used as a classifier which trains the EMNIST dataset.

The work is extended by adding some more dataset to EMNIST dataset of characters from English language and training the model. The prediction for the given input image is obtained from the trained classifier.

PROBLEM DEFINITION

Given a handwritten character, the system needs to predict the type of the character. In other words, if we can write the character “A” the system predicts the character that it is truly “A” or the input character is nearer to “A” or something else. The purpose of this project is to take the handwritten characters as an input process the character, train the neural network effectively by using the algorithm to recognize the pattern.

METHODOLOGY

Convolutional Neural Networks

CNN are made up of many interconnected neurons that have learnable weights and biases. In the architecture of CNN, the neurons are organized as layers. It consists of an input layer, many hidden layers, and an output layer. If the network has many hidden layers the same are referred as deep neural networks. The neurons in the hidden layers of CNN are connected to a small region (receptive field) of the input space generated from the previous layer instead of connecting to all, as in the fully connected network like Multi Layered Perceptron (MLP) networks. This approach reduces the number of connection weights (parameters) in CNN compared to MLP.

Therefore, CNN takes less time to train for the networks of similar size [8]. The input to the typical CNN is two dimensions (2D) array of data such as images. Unlike the regular neural network, the layers of a CNN are arranged in three dimensions (width, height, and depth).

The followings are the type of layers which as commonly found in the CNNs.

• Input Layer is a buffer to hold the input and pass it on to the next layer.

• Convolution Layer performs the core operation of feature extraction. It performs Convolution operation of the input data. The convolution operation is executed by sliding the kernel over the input and performs sum of the product at every location. The step size with which the kernel slides are known as stride. Numerous convolution operations are performed on the input by using different kernel, which results in different feature maps, the number of feature map produced in a convolutional layer is also known as the depth of the layer.

• Rectified Linear Unit (ReLU) is an activation function used to introduce nonlinearity. It replaces negative value with zero, which can speed up the learning process. The output of every convolution layer is passed through the activation function.

• Pooling layer reduces the spatial size of each feature map, which in turn reduces the computation in the network. Pooling also uses a sliding window that moves in stride across the feature map and transform it into representative values. Min pooling, average pooling and max pooling are commonly used.

• Fully connected layer connects every neuron in the layer to all the neurons in the previous layer. It learns the non-linear combination of the features and used for classifying or predicting the output. For classification problems, the fully connected layer is followed by a soft-max layer, it produces the probability of each class for the given input. And for regression problems, it is followed by a regression layer to predict the output.

CODE

Import Statements:

```
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from keras.utils.np_utils import to_categorical

from sklearn.model_selection import train_test_split

from sklearn.utils import shuffle #For shuffling the Images data

import cv2

import pandas as pd

import csv

data = pd.read_csv("A_Z Handwritten Data.csv").astype('float32')

print(data.head(10))
```

O/P :

```
     0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  ...  0.639  0.640  0.641  \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
6  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
7  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
8  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
9  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0

   0.642  0.643  0.644  0.645  0.646  0.647  0.648
0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
1    0.0    0.0    0.0    0.0    0.0    0.0    0.0
2    0.0    0.0    0.0    0.0    0.0    0.0    0.0
3    0.0    0.0    0.0    0.0    0.0    0.0    0.0
4    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5    0.0    0.0    0.0    0.0    0.0    0.0    0.0
6    0.0    0.0    0.0    0.0    0.0    0.0    0.0
7    0.0    0.0    0.0    0.0    0.0    0.0    0.0
8    0.0    0.0    0.0    0.0    0.0    0.0    0.0
9    0.0    0.0    0.0    0.0    0.0    0.0    0.0

[10 rows x 785 columns]
```

Dropping first column:

```
X = data.drop('0',axis = 1)

y = data['0']
```

Splitting the dataset to training and testing:

```
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)

# to reshape the Image and Label data according to our requirement

#shape size would be [28,28]

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)
```

O/P:

```
Train data shape:  (297960, 28, 28)
Test data shape:  (74490, 28, 28)
```

Word_dict list:

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

y_int = np.int0(y)

count = np.zeros(26, dtype='int')

print(count)

for i in y_int:

  count[i] +=1

alphabets = []

for i in word_dict.values():

   alphabets.append(i)
```

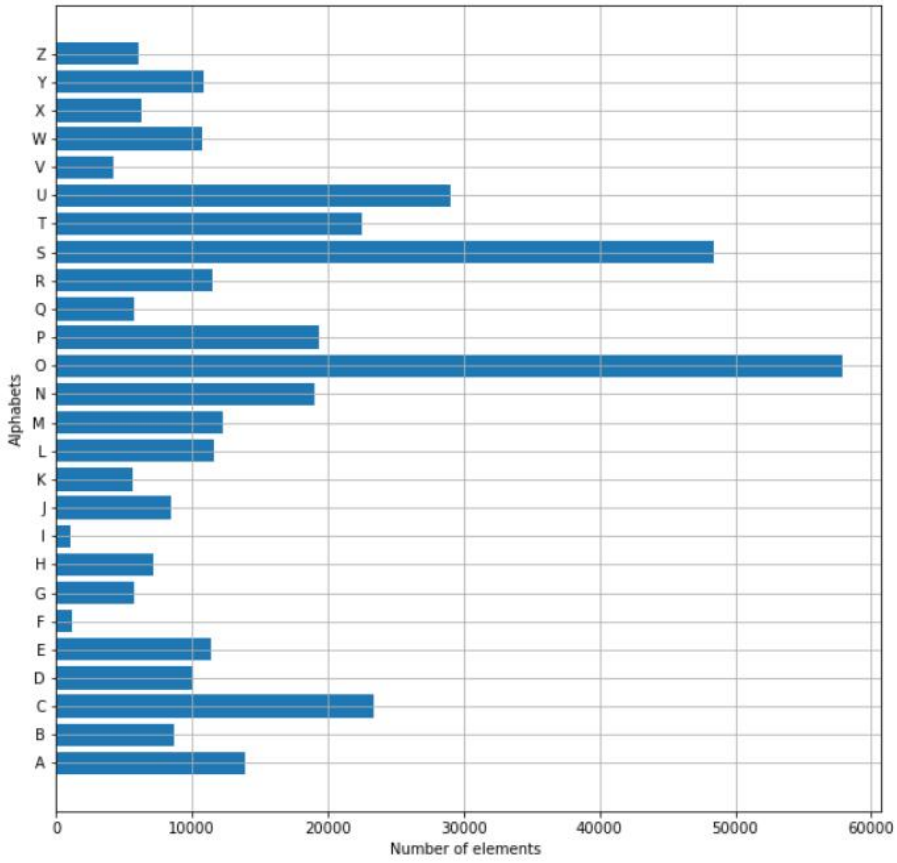
O/P:

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Plotting of alphabets from the dataset:

```
alphabets[1]

#We are ploting the bar graph of size 10 , 10

#In which we see that Alphabets "o" frequency is greater than 50,000

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

ax.barh(alphabets, count)

plt.xlabel("Number of elements ")

plt.ylabel("Alphabets")

plt.grid()

plt.show()
```

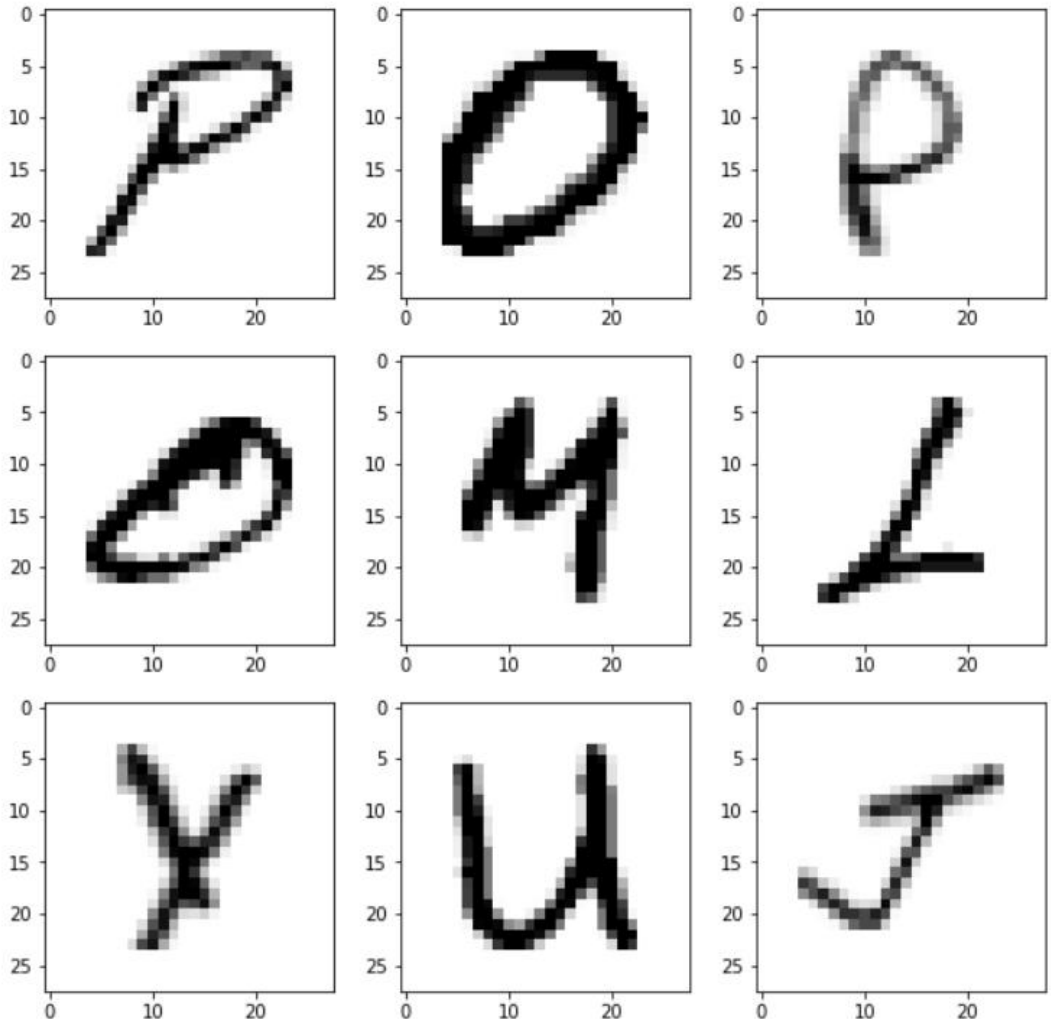
O/P:



Shuffling image data to get random alphabets for accuracy value:

```
shuff = shuffle(train_x[:10])

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="Greys")

plt.show()
```

O/P:



Reshaping and converting to categorical values:

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 train data: ", test_X.shape)

O/P:

```
New shape of train data:  (297960, 28, 28, 1)
New shape of train data:  (74490, 28, 28, 1)
```

#Here we convert the single float values to categorical values.
#This is done as the CNN model takes input of labels & generates
#the output as a vector of probabilities.

train_yOHE = to_categorical(train_y, num_classes = 26, dtype='int')

print("New shape of train labels: ", train_yOHE.shape)

test_yOHE = to_categorical(test_y, num_classes = 26, dtype='int')

print("New shape of test labels: ", test_yOHE.shape)

O/P:

```
New shape of train labels:  (297960, 26)
New shape of test labels:  (74490, 26)
```

Importing packages for CNN:

#CNN stands for Convolutional Neural Networks that are used to extract
#the features of the images using several layers of filters.
from __future__ **import** print_function
import tensorflow **as** tf
from tensorflow.keras.datasets **import** mnist
from tensorflow.keras.models **import** Sequential
from tensorflow.keras.layers **import** Dense, Dropout, Flatten
from tensorflow.keras.layers **import** Conv2D
from tensorflow.keras.layers **import** MaxPooling2D
from tensorflow.keras **import** backend **as** k
import tensorflow

CNN layer convolution:

#CNN gets the data as a input
 # Then its convolute the images with many different layers
 # After Convoluation there are pooling layers
 # The convolution 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

```
model = tf.keras.Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'valid'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Flatten())
model.add(Dense(64,activation ="relu"))
model.add(Dense(128,activation ="relu"))
model.add(Dense(26,activation ="softmax"))
```

Compile and fitting the model:

```
from tensorflow import keras
from tensorflow.keras import layers
opt = keras.optimizers.Adam(lr=0.001)
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_X, train_yOHE, epochs=1,  validation_data = (test_X,test_yOHE))
```

O/P:

```
Train on 297960 samples, validate on 74490 samples
Epoch 1/1
297960/297960 [==============================] - 44s 148us/step - loss: 0.2964 - acc: 0.9514 - val_loss: 0.0756 - val_acc: 0.97
91
```

Model summary:

model.summary()

model.save(r'model_hand.h5')

O/P:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                32832
_________________________________________________________________
dense_4 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_5 (Dense)              (None, 26)                3354
=================================================================
Total params: 137,178
Trainable params: 137,178
Non-trainable params: 0
```

Accuracy and Losses:

print("The validation accuracy is :", history.history['val_acc'])

print("The training accuracy is :", history.history['acc'])

print("The validation loss is :", history.history['val_loss'])

print("The training loss is :", history.history['loss'])

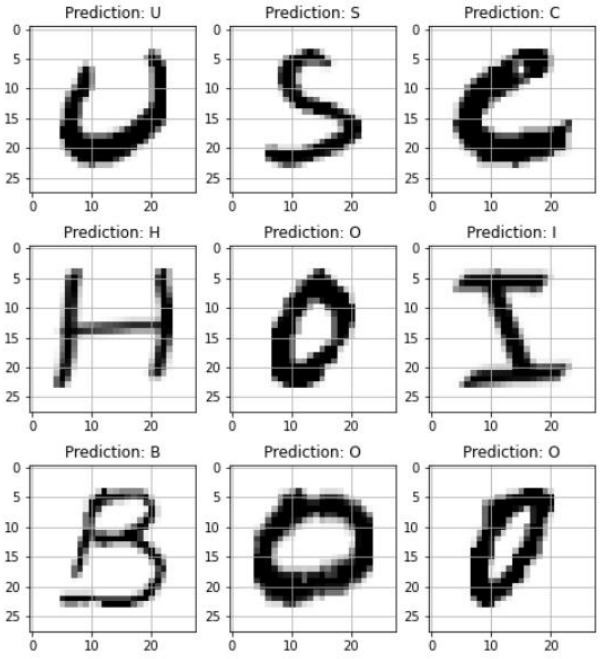
O/P:

```
The validation accuracy is : [0.9790844408645456]
The training accuracy is : [0.951433078265539]
The validation loss is : [0.07556949973212407]
The training loss is : [0.29639921997451135]
```

Predicting images based on the model:

```
fig, axes = plt.subplots(3,3, figsize=(8,9))

axes = axes.flatten()

# To showing the 9 images we will loop though the data from test model and will predict the

# images on the basis of CNN model of keras/tensorflow.

for i,ax in enumerate(axes):

    img = np.reshape(test_X[i], (28,28))

    ax.imshow(img, cmap="Greys")


    pred = word_dict[np.argmax(test_yOHE[i])]

    ax.set_title("Prediction: "+pred)

    ax.grid()
```

O/P:

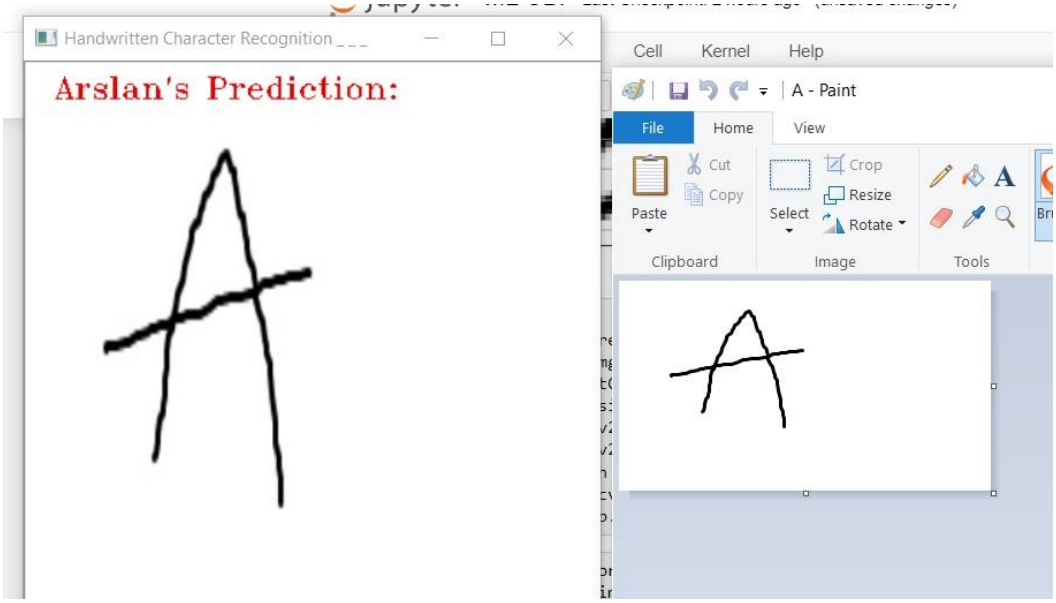


Importing image and checking whether the model predicted correctly or not:

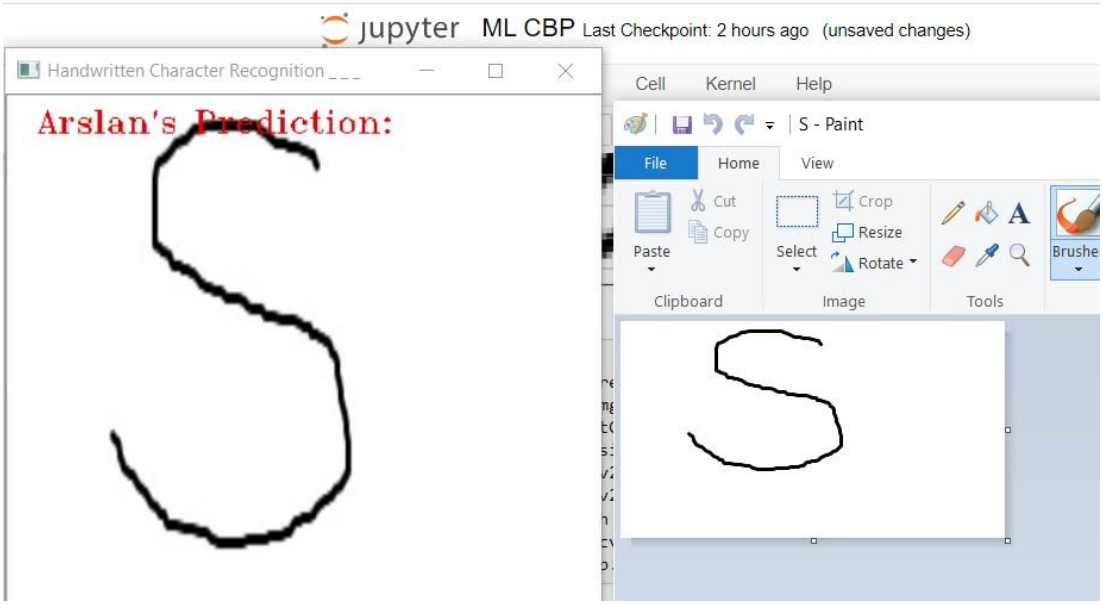
```
import cv2

img = cv2.imread('S.jpg')

img_copy = img.copy()

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

img = cv2.resize(img, (400,440))

img_copy = cv2.GaussianBlur(img_copy, (7,7), 0)

img_gray = cv2.cvtColor(img_copy, cv2.COLOR_BGR2GRAY)

_, img_thresh = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)

img_final = cv2.resize(img_thresh, (28,28))

img_final =np.reshape(img_final, (1,28,28,1))


img_pred = word_dict[np.argmax(model.predict(img_final))]

cv2.putText(img, "Arslan's Prediction: ", (20,25), cv2.FONT_HERSHEY_TRIPLEX, 0.7,
color = (0,0,230))

cv2.imshow('Handwritten character recognition _ _ _ ', img)


while (1):

    k = cv2.waitKey(1) & 0xFF

    if k == 27:

        break

cv2.destroyAllWindows()
```

Results:

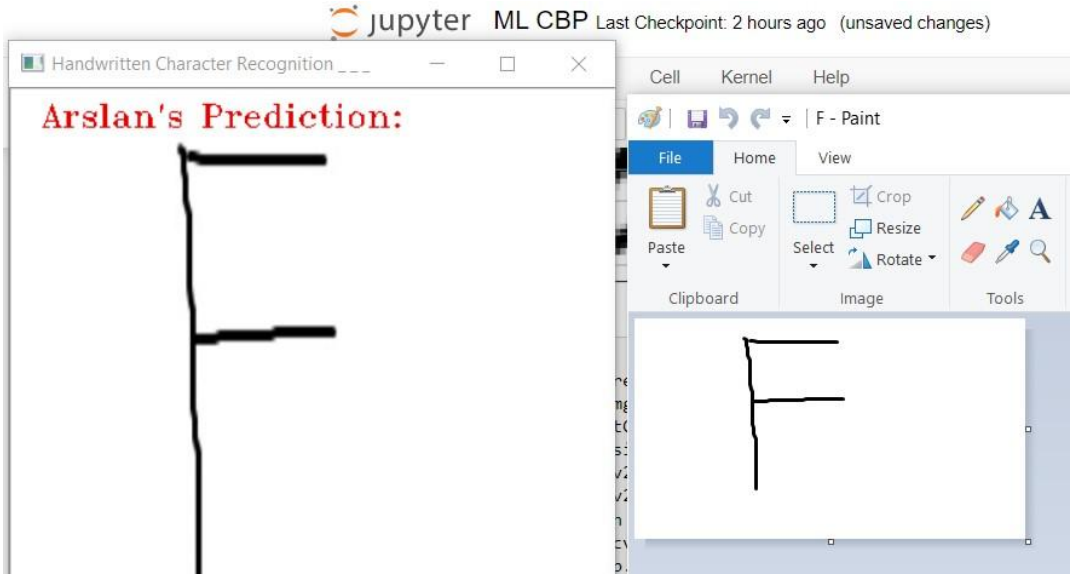
1) **From A.jpg :**



2) **From S.jpg:**



3) **From F.jpg:**



4) **From AKS.jpg:**