Deep Learning and Neural Networks - Industry Assignment 1

Handwritten Letter Classification Model - Train a Deep Learning model to classify any ten letters from any of the Indian Languages.

Importing Required Libraries

In [1]:
```
import cv2
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from keras.models import load_model
from keras.models import Sequential
from sklearn.utils import shuffle
from keras.utils import to_categorical
from keras.optimizers import SGD, Adam
from sklearn.model_selection import train_test_split
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

Importing the Dataset from Kaggle

In [2]:
```
data = pd.read_csv(r"A_Z Handwritten Data.csv").astype('float32')

data.head(5)
```

Out[2]:

	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	0.642	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	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	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	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	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 785 columns

Training and Testing the Data

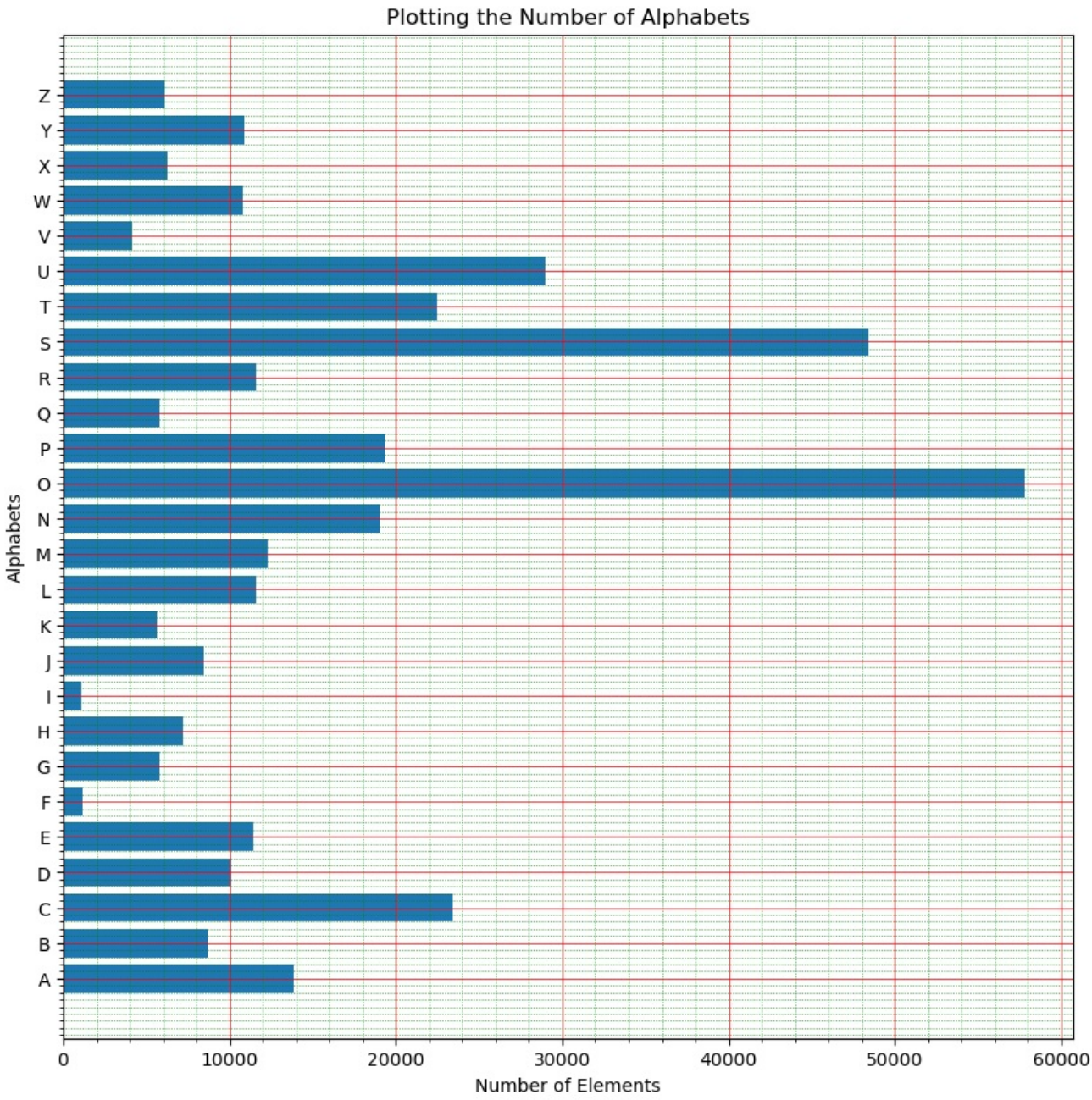
In [3]:
```
x = data.drop('0', axis=1)
y = data['0']

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("Training Data: ",train_x.shape)

print("Testing Data: ",test_x.shape)
```

```
Training Data:  (297960, 28, 28)
Testing Data:  (74490, 28, 28)
```

Plotting the Number of Alphabets

In [4]:
```
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:
            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')
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)

#Naming the x axis

plt.xlabel("Number of Elements")
```

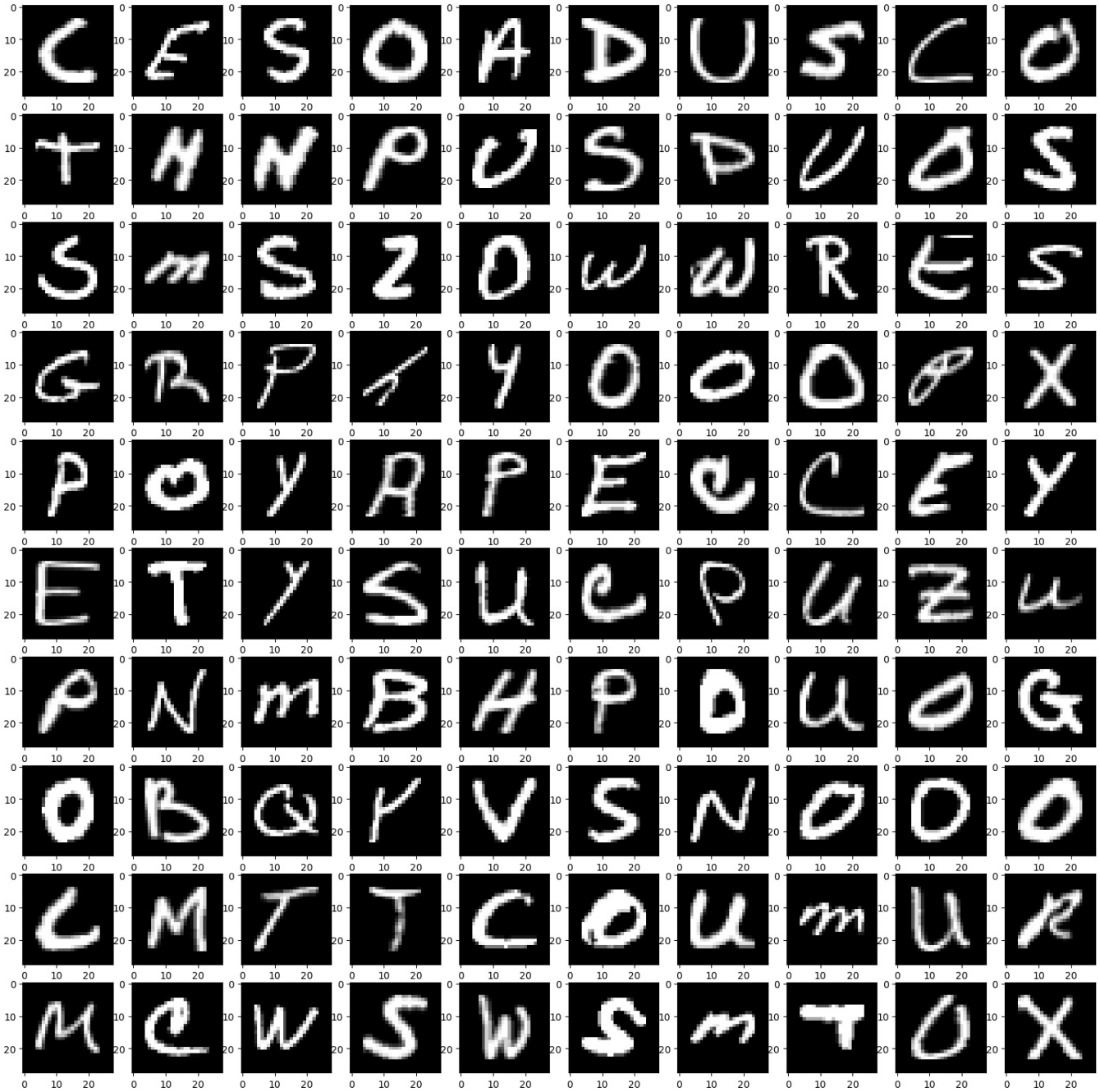
```
#Naming the y axis

plt.ylabel("Alphabets")

#Giving a Title

plt.title("Plotting the Number of Alphabets")

#Turning on the minor ticks required for minor grid

plt.minorticks_on()

#Customize the major grid

plt.grid(which='major',linestyle='-',linewidth='0.5',color='red')

#Customize the minor grid

plt.grid(which='minor',linestyle=':', linewidth='0.5', color='green')

plt.show()
```



Shuffling the Data into Images with 10 rows and 10 columns

In [5]:
```
#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=plt.get_cmap('gray'))

#plt.show()

shuff= shuffle(train_x[:100])
```

```
rows, cols = 10,10

plt.figure(figsize=(20,20))

for i in range(rows*cols):
    plt.subplot(cols,rows, i+1)
    plt.imshow(shuff[i].reshape(28,28),interpolation="nearest", cmap="gray")

plt.show()
```



Reshaping the Data for Model Creation and Model Building

In [6]:
```
#Reshaping the Data for Model Creation

train_x = train_x.reshape(train_x.shape[0], train_x.shape[1],train_x.shape[2],1)
print("New Shape of Training 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 Testing Data: ", test_x.shape)

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)
```

```
New Shape of Training Data:  (297960, 28, 28, 1)
New Shape of Testing Data:  (74490, 28, 28, 1)
New Shape of Train Labels:  (297960, 26)
New Shape of Test Labels:  (74490, 26)
```

Model Building

In [7]:
```
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPool2D(pool_size=(2,2), strides=2))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2,2), strides=2))

model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu', padding='valid'))
model.add(MaxPool2D(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"))

model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(train_x, train_yOHE, epochs=1, validation_data=(test_x,test_yOHE))

model.summary()
model.save(r'model_hand.h5')
```

```
9312/9312 [==============================] - 166s 18ms/step - loss: 0.1543 - accuracy: 0.9582 - val_loss: 0.080
3 - val_accuracy: 0.9770
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

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

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 64)                32832

 dense_1 (Dense)             (None, 128)               8320

 dense_2 (Dense)             (None, 26)                3354

=================================================================
Total params: 137,178
Trainable params: 137,178
Non-trainable params: 0
_________________________________________________________________
```

In [8]:
```
print("Validation Accuracy: ", history.history['val_accuracy'])

print("Validation Loss: ", history.history['val_loss'])

print("Training Accuracy: ", history.history['accuracy'])

print("Training Loss: ", history.history['loss'])
```

```
Validation Accuracy:  [0.976976752281189]
Validation Loss:  [0.08025951683521271]
Training Accuracy:  [0.9582158923149109]
Training Loss:  [0.15428753197193146]
```

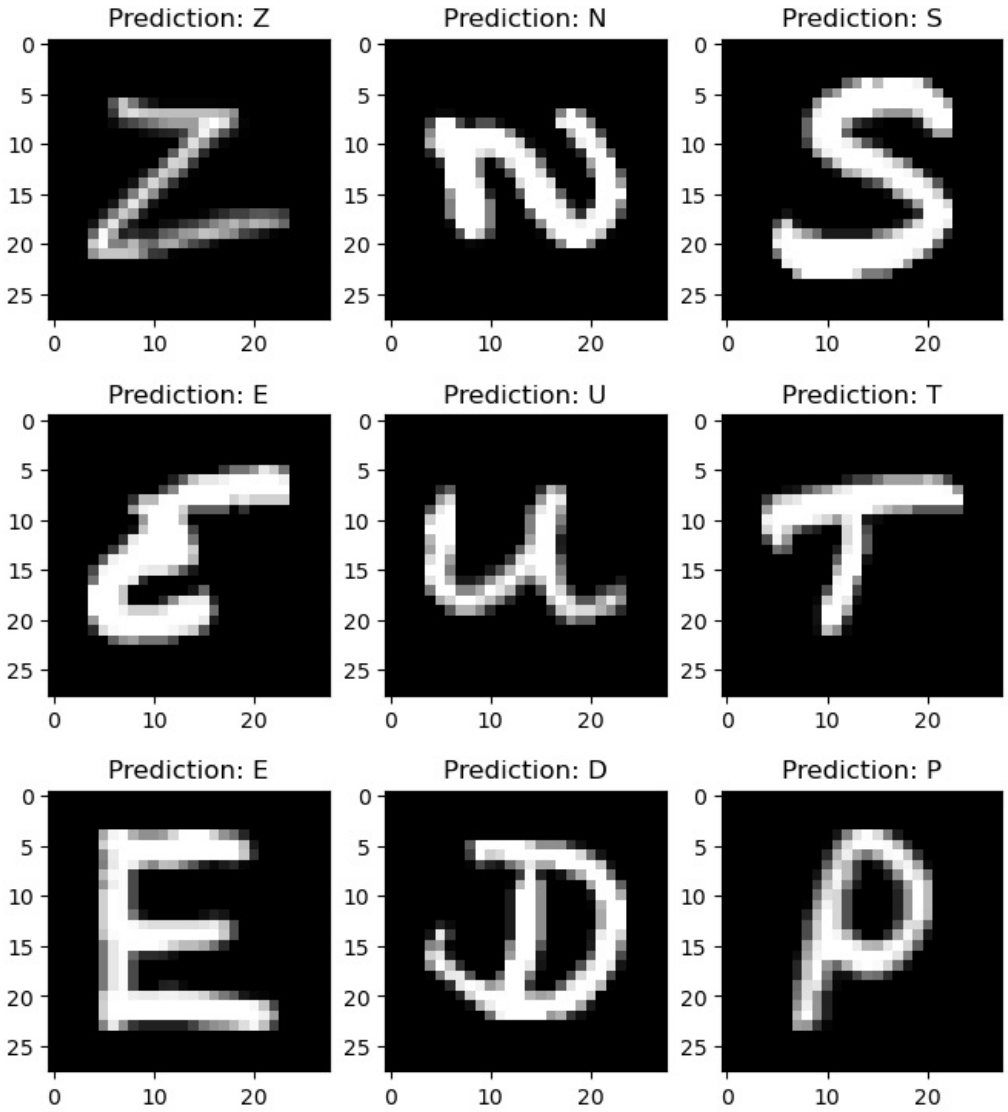
In [9]:
```
model = load_model('model_hand.h5')

model.summary()
```

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

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

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

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 64)                32832

 dense_1 (Dense)             (None, 128)               8320

 dense_2 (Dense)             (None, 26)                3354

=================================================================
Total params: 137,178
Trainable params: 137,178
Non-trainable params: 0
_________________________________________________________________
```

Prediction on the Test Data

In [10]:

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

for i, ax in enumerate(axes):
    img = np.reshape(test_x[i],(28,28))
    ax.imshow(img, cmap=plt.get_cmap('gray'))

    pred = word_dict[np.argmax(test_yOHE[i])]
    ax.set_title("Prediction: "+pred)
```



In [11]:
```
#Prediction on external image

img = cv2.imread('image.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)

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, "Image Data", (100,25), cv2.FONT_HERSHEY_DUPLEX, fontScale=1, thickness=2, color=(255,0,0))
cv2.putText(img, "Character Prediction: ", +img_pred, (10,410), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, thicknes
            color=(0,0,255))
cv2.imshow('Character Recognition: ',img)

while(1):
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break
        cv2.destroyAllWindows()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[11], line 4
      1 #Prediction on external image
      3 img = cv2.imread('image.jpg')
----> 4 img_copy = img.copy()
      6 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      7 img = cv2.resize(img(400,440))

AttributeError: 'NoneType' object has no attribute 'copy'
```