Hand-Written Stroke Analysis using ML for Lower KG students

In [6]:

```
!pip install imutils
```

```
Requirement already satisfied: imutils in c:\users\abhij\anaconda3\lib\site-
packages (0.5.4)
```

In [2]:

```
import numpy as np
import pandas as pd
from keras.preprocessing.image import ImageDataGenerator
import os
import random
import cv2
import imutils
import random
import matplotlib.pyplot as plt
import seaborn as sns
```

In [1]:

```
from sklearn.preprocessing import LabelBinarizer
from keras.utils import np_utils
from keras.models import Sequential
from keras import optimizers
from sklearn.preprocessing import LabelBinarizer
from keras import backend as K
from keras.layers import Dense, Activation, Flatten, Dense,MaxPooling2D, Dropout
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
```

In [3]:

```
dir = "./handwritten-characters/Train/"
train_data = []
img_size = 32
non_chars = ["#","$","&","@"]
for i in os.listdir(dir):
    if i in non_chars:
        continue
    count = 0
    sub_directory = os.path.join(dir,i)
    for j in os.listdir(sub_directory):
        count+=1
        if count > 4000:
            break
        img = cv2.imread(os.path.join(sub_directory,j),0)
        img = cv2.resize(img,(img_size,img_size))
        train_data.append([img,i])
```

In [4]:

```
len(train_data)
```

Out[4]:

```
140000
```

In [5]:

```
val_dir = "./handwritten-characters/Validation/"
val_data = []
img_size = 32
for i in os.listdir(val_dir):
    if i in non_chars:
        continue
    count = 0
    sub_directory = os.path.join(val_dir,i)
    for j in os.listdir(sub_directory):
        count+=1
        if count > 1000:
            break
        img = cv2.imread(os.path.join(sub_directory,j),0)
        img = cv2.resize(img,(img_size,img_size))
        val_data.append([img,i])
```

In [6]:

```
len(val_data)
```

Out[6]:

```
15209
```

In [7]:

```
random.shuffle(train_data)
random.shuffle(val_data)
```

In [8]:

```
train_X = []
train_Y = []
for features,label in train_data:
    train_X.append(features)
    train_Y.append(label)
```

In [9]:

```
val_X = []
val_Y = []
for features,label in val_data:
    val_X.append(features)
    val_Y.append(label)
```

In [10]:

```
LB = LabelBinarizer()
train_Y = LB.fit_transform(train_Y)
val_Y = LB.fit_transform(val_Y)
```

In [11]:

```
train_X = np.array(train_X)/255.0
train_X = train_X.reshape(-1,32,32,1)
train_Y = np.array(train_Y)
```

In [12]:

```
val_X = np.array(val_X)/255.0
val_X = val_X.reshape(-1,32,32,1)
val_Y = np.array(val_Y)
```

In [13]:

```
print(train_X.shape,val_X.shape)
```

```
(140000, 32, 32, 1) (15209, 32, 32, 1)
```

In [14]:

```
print(train_Y.shape,val_Y.shape)
```

```
(140000, 35) (15209, 35)
```

In [15]:

```
model = Sequential()

model.add(Conv2D(32, (3, 3), padding = "same", activation='relu', input_shape=(32,32,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(35, activation='softmax'))
```

In [16]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 5, 5, 128)         73856     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0         
 2D)                                                             
                                                                 
 dropout (Dropout)           (None, 2, 2, 128)         0         
                                                                 
 flatten (Flatten)           (None, 512)               0         
                                                                 
 dense (Dense)               (None, 128)               65664     
                                                                 
 dropout_1 (Dropout)         (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 35)                4515      
                                                                 
=================================================================
Total params: 162,851
Trainable params: 162,851
Non-trainable params: 0
_________________________________________________________________
```

In [17]:

```
model.compile(loss='categorical_crossentropy', optimizer="adam",metrics=['accuracy'])
```

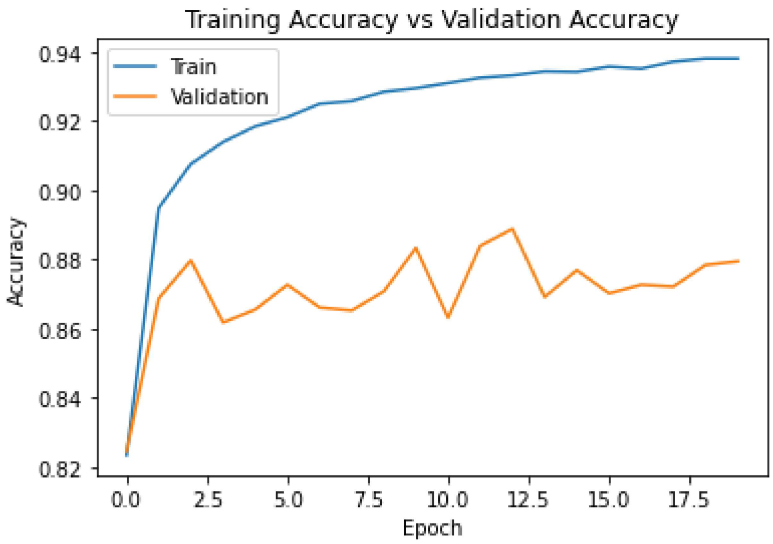
In [18]:

```
history = model.fit(train_X,train_Y, epochs=20, batch_size=32, validation_data = (val_X, va
```

```
Epoch 1/20
4375/4375 [==============================] - 149s 34ms/step - loss: 0.5422
- accuracy: 0.8234 - val_loss: 0.6978 - val_accuracy: 0.8246
Epoch 2/20
4375/4375 [==============================] - 158s 36ms/step - loss: 0.2966
- accuracy: 0.8949 - val_loss: 0.6473 - val_accuracy: 0.8688
Epoch 3/20
4375/4375 [==============================] - 230s 53ms/step - loss: 0.2558
- accuracy: 0.9076 - val_loss: 0.6108 - val_accuracy: 0.8797
Epoch 4/20
4375/4375 [==============================] - 155s 35ms/step - loss: 0.2363
- accuracy: 0.9140 - val_loss: 0.6306 - val_accuracy: 0.8618
Epoch 5/20
4375/4375 [==============================] - 346s 79ms/step - loss: 0.2199
- accuracy: 0.9186 - val_loss: 0.6678 - val_accuracy: 0.8655
Epoch 6/20
4375/4375 [==============================] - 157s 36ms/step - loss: 0.2078
- accuracy: 0.9212 - val_loss: 0.6241 - val_accuracy: 0.8727
Epoch 7/20
```

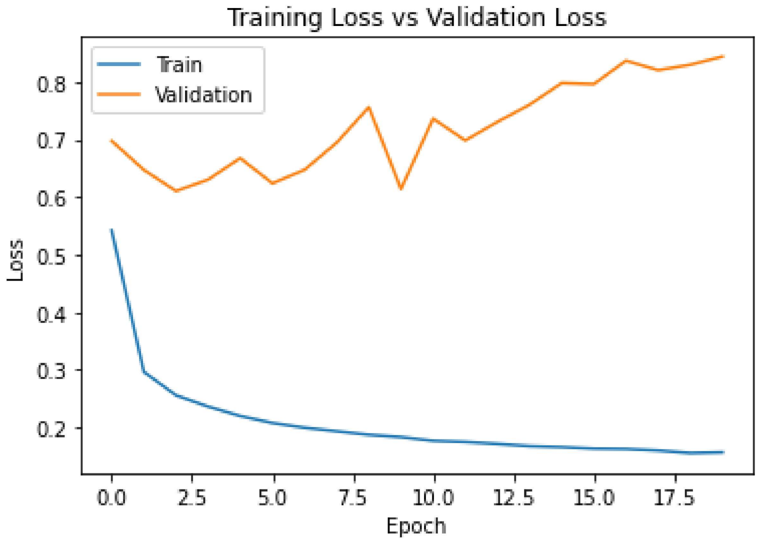
In [24]:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Training Accuracy vs Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



In [25]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training Loss vs Validation Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



In [26]:

```
def sort_contours(cnts, method="left-to-right"):
    reverse = False
    i = 0
    if method == "right-to-left" or method == "bottom-to-top":
        reverse = True
    if method == "top-to-bottom" or method == "bottom-to-top":
        i = 1
    boundingBoxes = [cv2.boundingRect(c) for c in cnts]
    (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes),
    key=lambda b:b[1][i], reverse=reverse))
    # return the list of sorted contours and bounding boxes
    return (cnts, boundingBoxes)
```

In [22]:

```
def get_letters(img):
    letters = []
    image = cv2.imread(img)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    ret,thresh1 = cv2.threshold(gray ,127,255,cv2.THRESH_BINARY_INV)
    dilated = cv2.dilate(thresh1, None, iterations=2)

    cnts = cv2.findContours(dilated.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    cnts = sort_contours(cnts, method="left-to-right")[0]
    # loop over the contours
    for c in cnts:
        if cv2.contourArea(c) > 10:
            (x, y, w, h) = cv2.boundingRect(c)
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
        roi = gray[y:y + h, x:x + w]
        thresh = cv2.threshold(roi, 0, 255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
        thresh = cv2.resize(thresh, (32, 32), interpolation = cv2.INTER_CUBIC)
        thresh = thresh.astype("float32") / 255.0
        thresh = np.expand_dims(thresh, axis=-1)
        thresh = thresh.reshape(1,32,32,1)
        ypred = model.predict(thresh)
        ypred = LB.inverse_transform(ypred)
        [x] = ypred
        letters.append(x)
    return letters, image

#plt.imshow(image)
```

In [23]:

```
def get_word(letter):
    word = "".join(letter)
    return word
```

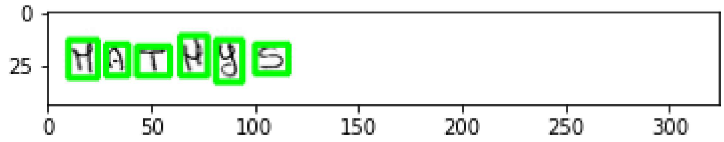
In [27]:

```
letter,image = get_letters("./test/TEST_0191.jpg")
word = get_word(letter)
print(word)
plt.imshow(image)
```

```
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
HATHY5
```

Out[27]:

```
<matplotlib.image.AxesImage at 0x1d7b3544a90>
```



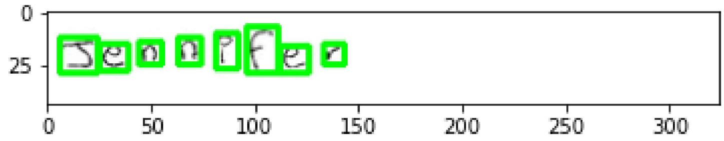
In [28]:

```
letter,image = get_letters("./test/TEST_0268.jpg")
word = get_word(letter)
print(word)
plt.imshow(image)
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 18ms/step
JP01TFPV
```

Out[28]:

```
<matplotlib.image.AxesImage at 0x1d7b35a0910>
```



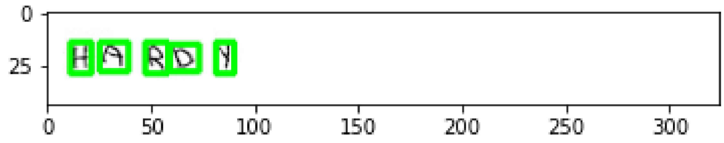
In [29]:

```
letter,image = get_letters("./test/TEST_0061.jpg")
word = get_word(letter)
print(word)
plt.imshow(image)
```

```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
HARDY
```

Out[29]:

```
<matplotlib.image.AxesImage at 0x1d7b3640a00>
```



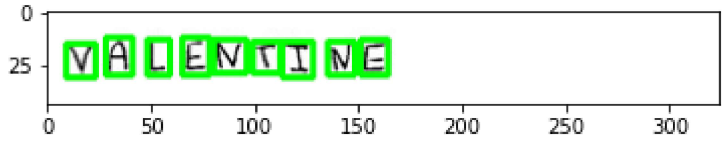
In [40]:

```
letter,image = get_letters("./test/TEST_0007.jpg")
word = get_word(letter)
print(word)
plt.imshow(image)
```

```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
VALENTIVE
```

Out[40]:

```
<matplotlib.image.AxesImage at 0x1d804580940>
```



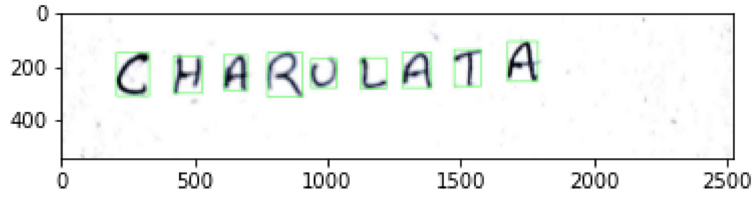
In [34]:

```
letter,image = get_letters("./Test_2.jpg")
word = get_word(letter)
print(word)
plt.imshow(image)
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
CHARULATA
```

Out[34]:

```
<matplotlib.image.AxesImage at 0x1d7b32dbeb0>
```



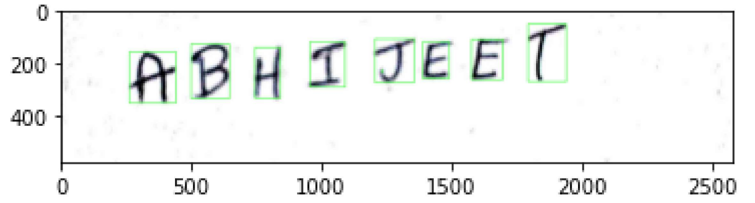
In [38]:

```
letter,image = get_letters("./Test_1.jpg")
word = get_word(letter)
print(word)
plt.imshow(image)
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
ABHIJLEET
```

Out[38]:

```
<matplotlib.image.AxesImage at 0x1d7b36f1310>
```



line segmentation >> word segmentation >> character segmentation >> classification >> post-processing.