UCS2612 Machine Learning Laboratory

A3 – Handwritten Character Recognition using Neural Networks

Name: C B Ananya
Reg No: 3122215001010

Question:

Download the handwritten character recognition dataset from the link given below:

https://www.kaggle.com/datasets/dhruvildave/english-handwritten-characters-dataset

This dataset contains 3,410 images of handwritten characters in English. This is a classification dataset that can be used for Computer Vision tasks. It contains 62 classes with 55 images of each class. The 62 classes are 0-9, A-Z and a-z.

Develop a python program to recognize handwritten characters using Neural Network (NN) Model. Visualize the features from the dataset and interpret the results obtained by the model using Matplotlib library.

Use the following steps to do implementation:

1. Loading the dataset.
2. Pre-Processing the data (Image Enhancement techniques)
3. Exploratory Data Analysis.
4. Feature Engineering techniques. (Image segmentation / Image Extraction)
5. Split the data into training, testing and validation sets.
6. Train the model.
7. Test the model.
8. Measure the performance of the trained model.
9. Represent the results using graphs.

Learning Objectives:

- Analyze dataset features to understand their relevance.
- Implement diverse classification algorithms for optimal performance.
- Enhance model discrimination through feature engineering.
- Evaluate hyperparameters and optimization techniques.
- Assess trade-offs between model complexity and performance.

Github Link: https://github.com/CB-Ananya/ML-Lab

Import necessary libraries

In [ ]:
```
!pip install opendatasets
```

```
Requirement already satisfied: opendatasets in c:\users\ananya\appdata\local\prog
rams\python\python311\lib\site-packages (0.1.22)
Requirement already satisfied: tqdm in c:\users\ananya\appdata\local\programs\pyt
hon\python311\lib\site-packages (from opendatasets) (4.66.1)
Requirement already satisfied: kaggle in c:\users\ananya\appdata\local\programs\p
ython\python311\lib\site-packages (from opendatasets) (1.6.6)
Requirement already satisfied: click in c:\users\ananya\appdata\local\programs\py
thon\python311\lib\site-packages (from opendatasets) (8.1.7)
Requirement already satisfied: colorama in c:\users\ananya\appdata\local\programs
\python\python311\lib\site-packages (from click->opendatasets) (0.4.6)
Requirement already satisfied: six>=1.10 in c:\users\ananya\appdata\local\program
s\python\python311\lib\site-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in c:\users\ananya\appdata\local\programs
\python\python311\lib\site-packages (from kaggle->opendatasets) (2023.7.22)
Requirement already satisfied: python-dateutil in c:\users\ananya\appdata\local\p
rograms\python\python311\lib\site-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in c:\users\ananya\appdata\local\programs
\python\python311\lib\site-packages (from kaggle->opendatasets) (2.31.0)
Requirement already satisfied: python-slugify in c:\users\ananya\appdata\local\pr
ograms\python\python311\lib\site-packages (from kaggle->opendatasets) (8.0.4)
Requirement already satisfied: urllib3 in c:\users\ananya\appdata\local\programs
\python\python311\lib\site-packages (from kaggle->opendatasets) (1.26.16)
Requirement already satisfied: bleach in c:\users\ananya\appdata\local\programs\p
ython\python311\lib\site-packages (from kaggle->opendatasets) (6.0.0)
Requirement already satisfied: webencodings in c:\users\ananya\appdata\local\prog
rams\python\python311\lib\site-packages (from bleach->kaggle->opendatasets) (0.5.
1)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\ananya\appdata\loc
al\programs\python\python311\lib\site-packages (from python-slugify->kaggle->open
datasets) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\ananya\appdat
a\local\programs\python\python311\lib\site-packages (from requests->kaggle->opend
atasets) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ananya\appdata\local\prog
rams\python\python311\lib\site-packages (from requests->kaggle->opendatasets) (3.
4)
```

In [ ]:
```
import cv2
import matplotlib.pyplot as plt
```

In [ ]:
```
import opendatasets as od
import pandas as pd

#od.download('https://www.kaggle.com/datasets/dhruvildave/english-handwritten-ch
```

Load the dataset from CSV file

In [ ]:
```
df = pd.read_csv('archive/english.csv')
df
```

Out[ ]:

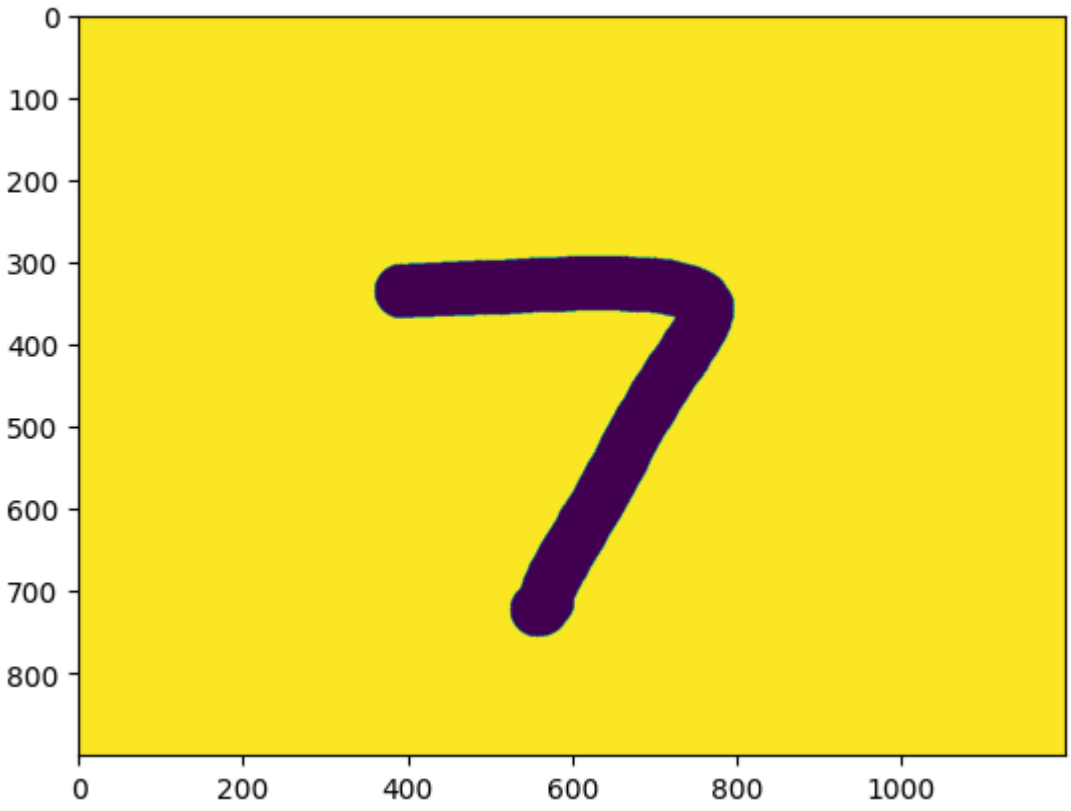
	image	label
0	Img/img001-001.png	0
1	Img/img001-002.png	0
2	Img/img001-003.png	0
3	Img/img001-004.png	0
4	Img/img001-005.png	0
...	...	...
3405	Img/img062-051.png	z
3406	Img/img062-052.png	z
3407	Img/img062-053.png	z
3408	Img/img062-054.png	z
3409	Img/img062-055.png	z

3410 rows × 2 columns

Read an image and display it

In [ ]:
```
img = cv2.imread('archive/Img/img008-025.png', cv2.IMREAD_GRAYSCALE)
plt.imshow(img)
```

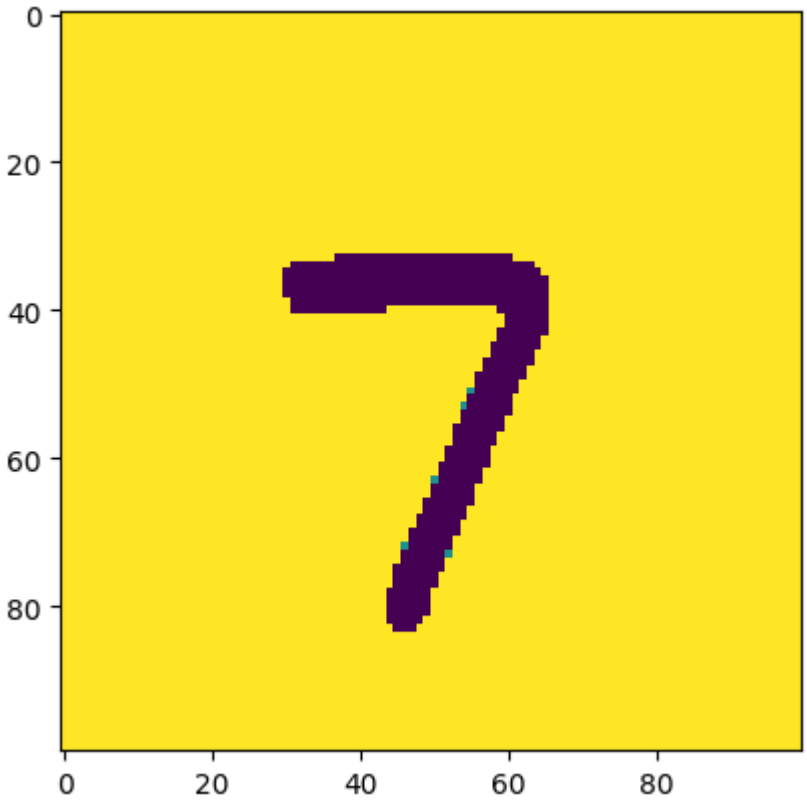
Out[ ]:
```
<matplotlib.image.AxesImage at 0x2caae1ba410>
```



Resize the image and display it again

In [ ]:
```
img = cv2.resize(img,(100,100))
plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x2caae240a10>
```



In [ ]:
```
img.shape
```

Out[ ]:
```
(100, 100)
```

Add the image path to the dataset

In [ ]:
```
df['image'] = 'archive/' + df['image']
```

Preprocess the labels

In [ ]:
```
from sklearn import preprocessing

# label_encoder object knows
# how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['label']= label_encoder.fit_transform(df['label'])
```

In [ ]:
```
df
```

Out[ ]:

	image	label
0	archive/Img/img001-001.png	0
1	archive/Img/img001-002.png	0
2	archive/Img/img001-003.png	0
3	archive/Img/img001-004.png	0
4	archive/Img/img001-005.png	0
...	...	...
3405	archive/Img/img062-051.png	61
3406	archive/Img/img062-052.png	61
3407	archive/Img/img062-053.png	61
3408	archive/Img/img062-054.png	61
3409	archive/Img/img062-055.png	61

3410 rows × 2 columns

Load the images and labels into lists

In [ ]:
```
images = []
labels = []

def imageLoad(row):
  path = row['image']
  img = cv2.imread(path)
  img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  img = cv2.resize(img,(100,100))
  images.append(img)
  labels.append(row['label'])
```

In [ ]:
```
df.apply(imageLoad,axis=1)
```

Out[ ]:
```
0       None
1       None
2       None
3       None
4       None
        ...
3405    None
3406    None
3407    None
3408    None
3409    None
Length: 3410, dtype: object
```

Split the dataset into training and testing sets

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(images,labels , random_state
```

Convert the datasets to tensors

In [ ]:
```
import tensorflow as tf

X_train = tf.convert_to_tensor(X_train)

X_test = tf.convert_to_tensor(X_test)
y_train = tf.convert_to_tensor(y_train)
y_test = tf.convert_to_tensor(y_test)
```

Normalize the pixel values

In [ ]:
```
X_train = X_train/255
X_test = X_test/255
```

Define the neural network model

In [ ]:
```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

In [ ]:
```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(100,100,1)))
model.add(layers.Dense(728,activation='relu'))
model.add(layers.Dense(500,activation='relu'))
model.add(layers.Dense(200,activation='relu'))
model.add(layers.Dense(100,activation='relu'))
model.add(layers.Dense(60,activation='relu'))
model.add(layers.Dense(62,activation='softmax'))
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 10000)             0

 dense (Dense)               (None, 728)               7280728

 dense_1 (Dense)             (None, 500)               364500

 dense_2 (Dense)             (None, 200)               100200

 dense_3 (Dense)             (None, 100)               20100

 dense_4 (Dense)             (None, 60)                6060

 dense_5 (Dense)             (None, 62)                3782

=================================================================
Total params: 7775370 (29.66 MB)
Trainable params: 7775370 (29.66 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Compile the model

In [ ]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='sgd', metrics=[
```

Train the model

In [ ]:
```
history = model.fit(X_train, y_train, epochs=150,
                    validation_data=(X_test, y_test))
```

```
Epoch 1/150
86/86 [==============================] - 4s 33ms/step - loss: 4.1398 - accuracy:
0.0202 - val_loss: 4.1282 - val_accuracy: 0.0220
Epoch 2/150
86/86 [==============================] - 3s 30ms/step - loss: 4.1293 - accuracy:
0.0224 - val_loss: 4.1335 - val_accuracy: 0.0235
Epoch 3/150
86/86 [==============================] - 2s 28ms/step - loss: 4.1276 - accuracy:
0.0169 - val_loss: 4.1257 - val_accuracy: 0.0161
Epoch 4/150
86/86 [==============================] - 3s 29ms/step - loss: 4.1197 - accuracy:
0.0231 - val_loss: 4.1176 - val_accuracy: 0.0220
Epoch 5/150
86/86 [==============================] - 3s 29ms/step - loss: 4.1160 - accuracy:
0.0257 - val_loss: 4.1179 - val_accuracy: 0.0220
Epoch 6/150
86/86 [==============================] - 3s 29ms/step - loss: 4.1096 - accuracy:
0.0224 - val_loss: 4.1000 - val_accuracy: 0.0103
Epoch 7/150
86/86 [==============================] - 3s 29ms/step - loss: 4.0989 - accuracy:
0.0275 - val_loss: 4.0961 - val_accuracy: 0.0205
Epoch 8/150
86/86 [==============================] - 3s 30ms/step - loss: 4.0699 - accuracy:
0.0323 - val_loss: 4.1589 - val_accuracy: 0.0191
Epoch 9/150
86/86 [==============================] - 3s 30ms/step - loss: 4.0528 - accuracy:
0.0334 - val_loss: 4.1805 - val_accuracy: 0.0191
Epoch 10/150
86/86 [==============================] - 3s 29ms/step - loss: 4.0275 - accuracy:
0.0341 - val_loss: 4.0932 - val_accuracy: 0.0235
Epoch 11/150
86/86 [==============================] - 3s 30ms/step - loss: 3.9946 - accuracy:
0.0374 - val_loss: 4.1295 - val_accuracy: 0.0161
Epoch 12/150
86/86 [==============================] - 3s 30ms/step - loss: 3.9512 - accuracy:
0.0517 - val_loss: 3.9452 - val_accuracy: 0.0235
Epoch 13/150
86/86 [==============================] - 3s 30ms/step - loss: 3.8858 - accuracy:
0.0572 - val_loss: 4.1248 - val_accuracy: 0.0440
Epoch 14/150
86/86 [==============================] - 3s 30ms/step - loss: 3.8541 - accuracy:
0.0663 - val_loss: 3.7862 - val_accuracy: 0.0836
Epoch 15/150
86/86 [==============================] - 3s 30ms/step - loss: 3.8019 - accuracy:
0.0737 - val_loss: 4.2007 - val_accuracy: 0.0117
Epoch 16/150
86/86 [==============================] - 3s 32ms/step - loss: 3.7661 - accuracy:
0.0762 - val_loss: 3.9056 - val_accuracy: 0.0323
Epoch 17/150
86/86 [==============================] - 3s 30ms/step - loss: 3.6871 - accuracy:
0.0887 - val_loss: 4.0003 - val_accuracy: 0.0674
Epoch 18/150
86/86 [==============================] - 3s 30ms/step - loss: 3.6586 - accuracy:
0.0894 - val_loss: 4.2397 - val_accuracy: 0.0367
Epoch 19/150
86/86 [==============================] - 3s 30ms/step - loss: 3.6168 - accuracy:
0.0953 - val_loss: 3.9081 - val_accuracy: 0.0601
Epoch 20/150
86/86 [==============================] - 3s 31ms/step - loss: 3.5465 - accuracy:
0.1096 - val_loss: 4.1869 - val_accuracy: 0.0425
```

```
Epoch 21/150
86/86 [==============================] - 3s 30ms/step - loss: 3.5223 - accuracy:
0.1103 - val_loss: 3.5221 - val_accuracy: 0.1056
Epoch 22/150
86/86 [==============================] - 3s 30ms/step - loss: 3.4611 - accuracy:
0.1224 - val_loss: 3.6566 - val_accuracy: 0.0836
Epoch 23/150
86/86 [==============================] - 3s 31ms/step - loss: 3.4236 - accuracy:
0.1301 - val_loss: 3.6118 - val_accuracy: 0.0909
Epoch 24/150
86/86 [==============================] - 3s 30ms/step - loss: 3.3715 - accuracy:
0.1422 - val_loss: 4.3091 - val_accuracy: 0.0455
Epoch 25/150
86/86 [==============================] - 3s 30ms/step - loss: 3.3213 - accuracy:
0.1525 - val_loss: 4.0704 - val_accuracy: 0.0484
Epoch 26/150
86/86 [==============================] - 3s 32ms/step - loss: 3.2627 - accuracy:
0.1620 - val_loss: 3.4400 - val_accuracy: 0.0953
Epoch 27/150
86/86 [==============================] - 3s 32ms/step - loss: 3.2165 - accuracy:
0.1800 - val_loss: 4.0521 - val_accuracy: 0.0587
Epoch 28/150
86/86 [==============================] - 3s 32ms/step - loss: 3.1718 - accuracy:
0.1774 - val_loss: 3.3710 - val_accuracy: 0.1158
Epoch 29/150
86/86 [==============================] - 3s 31ms/step - loss: 3.1477 - accuracy:
0.1877 - val_loss: 3.5032 - val_accuracy: 0.1085
Epoch 30/150
86/86 [==============================] - 3s 31ms/step - loss: 3.0904 - accuracy:
0.2031 - val_loss: 3.2707 - val_accuracy: 0.1466
Epoch 31/150
86/86 [==============================] - 3s 36ms/step - loss: 3.0197 - accuracy:
0.2097 - val_loss: 3.3154 - val_accuracy: 0.1217
Epoch 32/150
86/86 [==============================] - 3s 31ms/step - loss: 3.0372 - accuracy:
0.2152 - val_loss: 4.0587 - val_accuracy: 0.0616
Epoch 33/150
86/86 [==============================] - 3s 32ms/step - loss: 2.9381 - accuracy:
0.2309 - val_loss: 3.4804 - val_accuracy: 0.1129
Epoch 34/150
86/86 [==============================] - 3s 33ms/step - loss: 2.9189 - accuracy:
0.2386 - val_loss: 3.0531 - val_accuracy: 0.1994
Epoch 35/150
86/86 [==============================] - 3s 32ms/step - loss: 2.9219 - accuracy:
0.2262 - val_loss: 3.0998 - val_accuracy: 0.1935
Epoch 36/150
86/86 [==============================] - 3s 31ms/step - loss: 2.8449 - accuracy:
0.2507 - val_loss: 3.4858 - val_accuracy: 0.1173
Epoch 37/150
86/86 [==============================] - 3s 31ms/step - loss: 2.7852 - accuracy:
0.2691 - val_loss: 3.6151 - val_accuracy: 0.1026
Epoch 38/150
86/86 [==============================] - 3s 31ms/step - loss: 2.7274 - accuracy:
0.2867 - val_loss: 2.9975 - val_accuracy: 0.2331
Epoch 39/150
86/86 [==============================] - 3s 32ms/step - loss: 2.7379 - accuracy:
0.2650 - val_loss: 2.8751 - val_accuracy: 0.2155
Epoch 40/150
86/86 [==============================] - 3s 31ms/step - loss: 2.6755 - accuracy:
0.2870 - val_loss: 3.4319 - val_accuracy: 0.1481
```

```
Epoch 41/150
86/86 [==============================] - 3s 31ms/step - loss: 2.6171 - accuracy:
0.3006 - val_loss: 4.2460 - val_accuracy: 0.1026
Epoch 42/150
86/86 [==============================] - 3s 30ms/step - loss: 2.6244 - accuracy:
0.2933 - val_loss: 3.3947 - val_accuracy: 0.1525
Epoch 43/150
86/86 [==============================] - 3s 31ms/step - loss: 2.5804 - accuracy:
0.3079 - val_loss: 3.6054 - val_accuracy: 0.1378
Epoch 44/150
86/86 [==============================] - 3s 33ms/step - loss: 2.5519 - accuracy:
0.3076 - val_loss: 2.8019 - val_accuracy: 0.2361
Epoch 45/150
86/86 [==============================] - 3s 31ms/step - loss: 2.5086 - accuracy:
0.3156 - val_loss: 3.1829 - val_accuracy: 0.1701
Epoch 46/150
86/86 [==============================] - 3s 32ms/step - loss: 2.4804 - accuracy:
0.3391 - val_loss: 3.0382 - val_accuracy: 0.2009
Epoch 47/150
86/86 [==============================] - 3s 30ms/step - loss: 2.4464 - accuracy:
0.3391 - val_loss: 3.2072 - val_accuracy: 0.1745
Epoch 48/150
86/86 [==============================] - 3s 31ms/step - loss: 2.3567 - accuracy:
0.3625 - val_loss: 3.1787 - val_accuracy: 0.2053
Epoch 49/150
86/86 [==============================] - 3s 31ms/step - loss: 2.3449 - accuracy:
0.3545 - val_loss: 3.9311 - val_accuracy: 0.1026
Epoch 50/150
86/86 [==============================] - 3s 32ms/step - loss: 2.3141 - accuracy:
0.3691 - val_loss: 4.3240 - val_accuracy: 0.0997
Epoch 51/150
86/86 [==============================] - 3s 32ms/step - loss: 2.3160 - accuracy:
0.3713 - val_loss: 2.6698 - val_accuracy: 0.2757
Epoch 52/150
86/86 [==============================] - 3s 31ms/step - loss: 2.2473 - accuracy:
0.3746 - val_loss: 3.2634 - val_accuracy: 0.1598
Epoch 53/150
86/86 [==============================] - 3s 34ms/step - loss: 2.2457 - accuracy:
0.3812 - val_loss: 2.5210 - val_accuracy: 0.3314
Epoch 54/150
86/86 [==============================] - 3s 30ms/step - loss: 2.1838 - accuracy:
0.3937 - val_loss: 2.9612 - val_accuracy: 0.2185
Epoch 55/150
86/86 [==============================] - 3s 32ms/step - loss: 2.1110 - accuracy:
0.4062 - val_loss: 3.6061 - val_accuracy: 0.1496
Epoch 56/150
86/86 [==============================] - 3s 31ms/step - loss: 2.1536 - accuracy:
0.4003 - val_loss: 2.9849 - val_accuracy: 0.2361
Epoch 57/150
86/86 [==============================] - 3s 31ms/step - loss: 2.0463 - accuracy:
0.4399 - val_loss: 2.6732 - val_accuracy: 0.2830
Epoch 58/150
86/86 [==============================] - 3s 32ms/step - loss: 2.1184 - accuracy:
0.4058 - val_loss: 4.3034 - val_accuracy: 0.1026
Epoch 59/150
86/86 [==============================] - 3s 34ms/step - loss: 2.0756 - accuracy:
0.4201 - val_loss: 2.6085 - val_accuracy: 0.3167
Epoch 60/150
86/86 [==============================] - 3s 31ms/step - loss: 1.9564 - accuracy:
0.4483 - val_loss: 3.1792 - val_accuracy: 0.2155
```

```
Epoch 61/150
86/86 [==============================] - 3s 31ms/step - loss: 1.9567 - accuracy:
0.4428 - val_loss: 3.2416 - val_accuracy: 0.2229
Epoch 62/150
86/86 [==============================] - 3s 31ms/step - loss: 1.9168 - accuracy:
0.4523 - val_loss: 2.6679 - val_accuracy: 0.3196
Epoch 63/150
86/86 [==============================] - 3s 31ms/step - loss: 1.8610 - accuracy:
0.4666 - val_loss: 2.7036 - val_accuracy: 0.2859
Epoch 64/150
86/86 [==============================] - 3s 31ms/step - loss: 1.8731 - accuracy:
0.4534 - val_loss: 3.9003 - val_accuracy: 0.1906
Epoch 65/150
86/86 [==============================] - 3s 32ms/step - loss: 1.8251 - accuracy:
0.4784 - val_loss: 3.3120 - val_accuracy: 0.1642
Epoch 66/150
86/86 [==============================] - 3s 31ms/step - loss: 1.7972 - accuracy:
0.4879 - val_loss: 3.5212 - val_accuracy: 0.2126
Epoch 67/150
86/86 [==============================] - 3s 32ms/step - loss: 1.7884 - accuracy:
0.4938 - val_loss: 3.4576 - val_accuracy: 0.1994
Epoch 68/150
86/86 [==============================] - 3s 31ms/step - loss: 1.7248 - accuracy:
0.5084 - val_loss: 3.5214 - val_accuracy: 0.2053
Epoch 69/150
86/86 [==============================] - 3s 32ms/step - loss: 1.6656 - accuracy:
0.5220 - val_loss: 2.7967 - val_accuracy: 0.3196
Epoch 70/150
86/86 [==============================] - 3s 31ms/step - loss: 1.6342 - accuracy:
0.5238 - val_loss: 3.2141 - val_accuracy: 0.2229
Epoch 71/150
86/86 [==============================] - 3s 30ms/step - loss: 1.6079 - accuracy:
0.5319 - val_loss: 2.2730 - val_accuracy: 0.4326
Epoch 72/150
86/86 [==============================] - 3s 31ms/step - loss: 1.5692 - accuracy:
0.5370 - val_loss: 2.5639 - val_accuracy: 0.3328
Epoch 73/150
86/86 [==============================] - 3s 30ms/step - loss: 1.5566 - accuracy:
0.5407 - val_loss: 2.5149 - val_accuracy: 0.3666
Epoch 74/150
86/86 [==============================] - 3s 30ms/step - loss: 1.5310 - accuracy:
0.5436 - val_loss: 2.7843 - val_accuracy: 0.2889
Epoch 75/150
86/86 [==============================] - 3s 31ms/step - loss: 1.5057 - accuracy:
0.5513 - val_loss: 3.2242 - val_accuracy: 0.2346
Epoch 76/150
86/86 [==============================] - 3s 30ms/step - loss: 1.4150 - accuracy:
0.5847 - val_loss: 3.0998 - val_accuracy: 0.3343
Epoch 77/150
86/86 [==============================] - 3s 31ms/step - loss: 1.4983 - accuracy:
0.5587 - val_loss: 2.4198 - val_accuracy: 0.3328
Epoch 78/150
86/86 [==============================] - 3s 31ms/step - loss: 1.3897 - accuracy:
0.5850 - val_loss: 3.1071 - val_accuracy: 0.2551
Epoch 79/150
86/86 [==============================] - 3s 31ms/step - loss: 1.3780 - accuracy:
0.5898 - val_loss: 3.2071 - val_accuracy: 0.2830
Epoch 80/150
86/86 [==============================] - 3s 31ms/step - loss: 1.3987 - accuracy:
0.5821 - val_loss: 2.2866 - val_accuracy: 0.4120
```

```
Epoch 81/150
86/86 [==============================] - 3s 31ms/step - loss: 1.3038 - accuracy:
0.6092 - val_loss: 2.9664 - val_accuracy: 0.3152
Epoch 82/150
86/86 [==============================] - 3s 31ms/step - loss: 1.2737 - accuracy:
0.6092 - val_loss: 2.5701 - val_accuracy: 0.3710
Epoch 83/150
86/86 [==============================] - 3s 30ms/step - loss: 1.2907 - accuracy:
0.6136 - val_loss: 2.1712 - val_accuracy: 0.4076
Epoch 84/150
86/86 [==============================] - 3s 33ms/step - loss: 1.2829 - accuracy:
0.6096 - val_loss: 2.2144 - val_accuracy: 0.4252
Epoch 85/150
86/86 [==============================] - 3s 31ms/step - loss: 1.2351 - accuracy:
0.6294 - val_loss: 2.4923 - val_accuracy: 0.3695
Epoch 86/150
86/86 [==============================] - 3s 31ms/step - loss: 1.2278 - accuracy:
0.6316 - val_loss: 2.8972 - val_accuracy: 0.2859
Epoch 87/150
86/86 [==============================] - 3s 31ms/step - loss: 1.1775 - accuracy:
0.6422 - val_loss: 2.2766 - val_accuracy: 0.4208
Epoch 88/150
86/86 [==============================] - 3s 31ms/step - loss: 1.1356 - accuracy:
0.6492 - val_loss: 2.8342 - val_accuracy: 0.3651
Epoch 89/150
86/86 [==============================] - 3s 32ms/step - loss: 1.1854 - accuracy:
0.6408 - val_loss: 2.1095 - val_accuracy: 0.4296
Epoch 90/150
86/86 [==============================] - 3s 32ms/step - loss: 1.0953 - accuracy:
0.6624 - val_loss: 2.1086 - val_accuracy: 0.4545
Epoch 91/150
86/86 [==============================] - 3s 31ms/step - loss: 1.0912 - accuracy:
0.6631 - val_loss: 3.2080 - val_accuracy: 0.3196
Epoch 92/150
86/86 [==============================] - 3s 31ms/step - loss: 1.0484 - accuracy:
0.6796 - val_loss: 7.1022 - val_accuracy: 0.1041
Epoch 93/150
86/86 [==============================] - 3s 31ms/step - loss: 1.2375 - accuracy:
0.6433 - val_loss: 2.2658 - val_accuracy: 0.4604
Epoch 94/150
86/86 [==============================] - 3s 31ms/step - loss: 0.9943 - accuracy:
0.6880 - val_loss: 3.4600 - val_accuracy: 0.3167
Epoch 95/150
86/86 [==============================] - 3s 31ms/step - loss: 1.0203 - accuracy:
0.6917 - val_loss: 2.2082 - val_accuracy: 0.4560
Epoch 96/150
86/86 [==============================] - 3s 31ms/step - loss: 0.9294 - accuracy:
0.7056 - val_loss: 4.0039 - val_accuracy: 0.2551
Epoch 97/150
86/86 [==============================] - 3s 30ms/step - loss: 0.9833 - accuracy:
0.6979 - val_loss: 2.4309 - val_accuracy: 0.4135
Epoch 98/150
86/86 [==============================] - 3s 31ms/step - loss: 0.9263 - accuracy:
0.7199 - val_loss: 3.1840 - val_accuracy: 0.3211
Epoch 99/150
86/86 [==============================] - 3s 31ms/step - loss: 0.8656 - accuracy:
0.7353 - val_loss: 2.1505 - val_accuracy: 0.4809
Epoch 100/150
86/86 [==============================] - 3s 31ms/step - loss: 0.8673 - accuracy:
0.7269 - val_loss: 2.3038 - val_accuracy: 0.4413
```

```
Epoch 101/150
86/86 [==============================] - 3s 31ms/step - loss: 0.8224 - accuracy:
0.7434 - val_loss: 2.0906 - val_accuracy: 0.4736
Epoch 102/150
86/86 [==============================] - 3s 31ms/step - loss: 0.8642 - accuracy:
0.7298 - val_loss: 6.5675 - val_accuracy: 0.1716
Epoch 103/150
86/86 [==============================] - 3s 31ms/step - loss: 1.0198 - accuracy:
0.7214 - val_loss: 2.4224 - val_accuracy: 0.4355
Epoch 104/150
86/86 [==============================] - 3s 31ms/step - loss: 1.0085 - accuracy:
0.7155 - val_loss: 2.2942 - val_accuracy: 0.4604
Epoch 105/150
86/86 [==============================] - 3s 31ms/step - loss: 0.7349 - accuracy:
0.7621 - val_loss: 2.1612 - val_accuracy: 0.4663
Epoch 106/150
86/86 [==============================] - 3s 31ms/step - loss: 0.7394 - accuracy:
0.7801 - val_loss: 5.6617 - val_accuracy: 0.1642
Epoch 107/150
86/86 [==============================] - 3s 30ms/step - loss: 1.0307 - accuracy:
0.7203 - val_loss: 3.0148 - val_accuracy: 0.3519
Epoch 108/150
86/86 [==============================] - 3s 31ms/step - loss: 0.7103 - accuracy:
0.7793 - val_loss: 2.7989 - val_accuracy: 0.3798
Epoch 109/150
86/86 [==============================] - 3s 30ms/step - loss: 0.7879 - accuracy:
0.7540 - val_loss: 2.5551 - val_accuracy: 0.4399
Epoch 110/150
86/86 [==============================] - 3s 32ms/step - loss: 0.7071 - accuracy:
0.7764 - val_loss: 2.4797 - val_accuracy: 0.4545
Epoch 111/150
86/86 [==============================] - 3s 31ms/step - loss: 0.6333 - accuracy:
0.7955 - val_loss: 3.9926 - val_accuracy: 0.2742
Epoch 112/150
86/86 [==============================] - 3s 32ms/step - loss: 0.8890 - accuracy:
0.7427 - val_loss: 3.7355 - val_accuracy: 0.2918
Epoch 113/150
86/86 [==============================] - 3s 30ms/step - loss: 0.7859 - accuracy:
0.7687 - val_loss: 2.2741 - val_accuracy: 0.4736
Epoch 114/150
86/86 [==============================] - 3s 31ms/step - loss: 0.5969 - accuracy:
0.8079 - val_loss: 2.3119 - val_accuracy: 0.4487
Epoch 115/150
86/86 [==============================] - 3s 31ms/step - loss: 0.5530 - accuracy:
0.8262 - val_loss: 2.0649 - val_accuracy: 0.5308
Epoch 116/150
86/86 [==============================] - 3s 31ms/step - loss: 0.5772 - accuracy:
0.8204 - val_loss: 2.5847 - val_accuracy: 0.4619
Epoch 117/150
86/86 [==============================] - 3s 31ms/step - loss: 0.5293 - accuracy:
0.8343 - val_loss: 2.6310 - val_accuracy: 0.4560
Epoch 118/150
86/86 [==============================] - 3s 31ms/step - loss: 0.5786 - accuracy:
0.8163 - val_loss: 3.0971 - val_accuracy: 0.4340
Epoch 119/150
86/86 [==============================] - 3s 31ms/step - loss: 0.5898 - accuracy:
0.8229 - val_loss: 2.3542 - val_accuracy: 0.4853
Epoch 120/150
86/86 [==============================] - 3s 32ms/step - loss: 0.4117 - accuracy:
0.8735 - val_loss: 2.1593 - val_accuracy: 0.5249
```

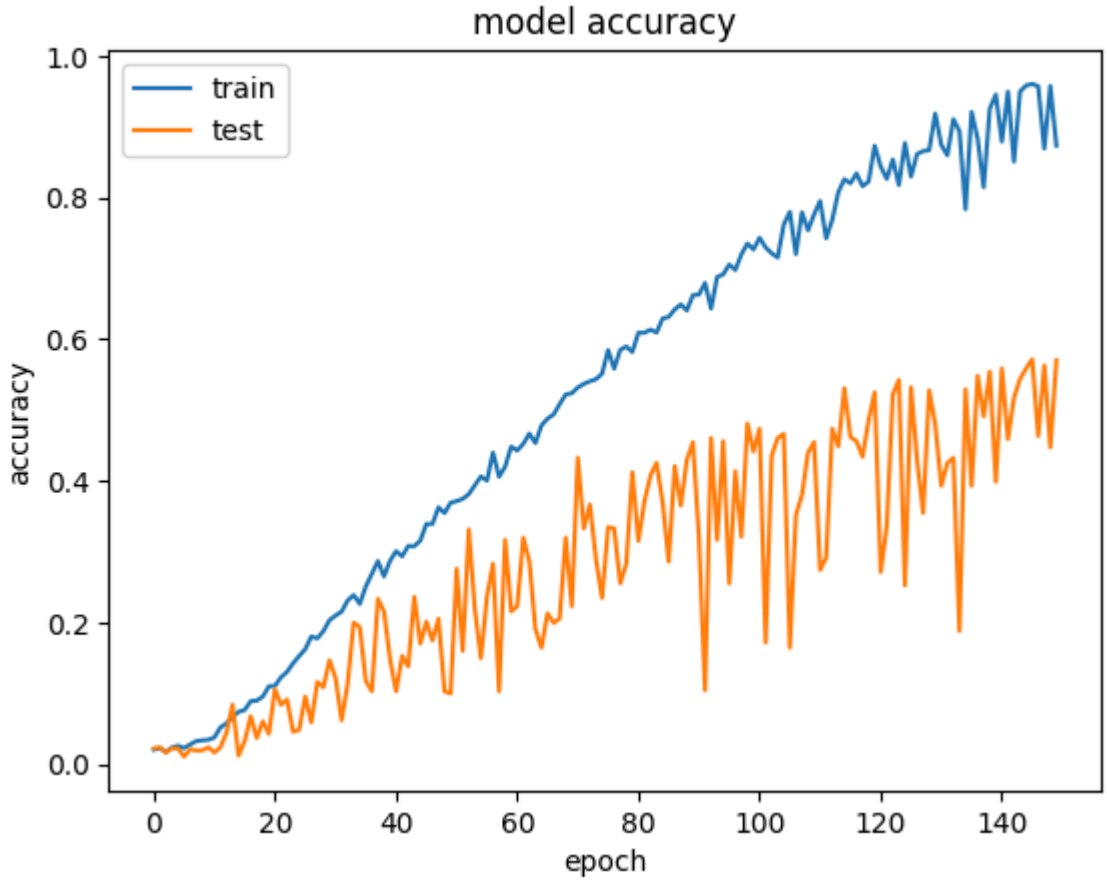
```
Epoch 121/150
86/86 [==============================] - 3s 31ms/step - loss: 0.5108 - accuracy:
0.8442 - val_loss: 4.7562 - val_accuracy: 0.2713
Epoch 122/150
86/86 [==============================] - 3s 32ms/step - loss: 0.5993 - accuracy:
0.8266 - val_loss: 3.6787 - val_accuracy: 0.3358
Epoch 123/150
86/86 [==============================] - 3s 32ms/step - loss: 0.4795 - accuracy:
0.8541 - val_loss: 2.2448 - val_accuracy: 0.5220
Epoch 124/150
86/86 [==============================] - 3s 31ms/step - loss: 0.6873 - accuracy:
0.8178 - val_loss: 2.0232 - val_accuracy: 0.5425
Epoch 125/150
86/86 [==============================] - 3s 31ms/step - loss: 0.3937 - accuracy:
0.8772 - val_loss: 5.6609 - val_accuracy: 0.2522
Epoch 126/150
86/86 [==============================] - 3s 32ms/step - loss: 0.6407 - accuracy:
0.8299 - val_loss: 2.2325 - val_accuracy: 0.5323
Epoch 127/150
86/86 [==============================] - 3s 31ms/step - loss: 0.4662 - accuracy:
0.8618 - val_loss: 2.7854 - val_accuracy: 0.4296
Epoch 128/150
86/86 [==============================] - 3s 31ms/step - loss: 0.4288 - accuracy:
0.8658 - val_loss: 3.4640 - val_accuracy: 0.3548
Epoch 129/150
86/86 [==============================] - 3s 32ms/step - loss: 0.5114 - accuracy:
0.8673 - val_loss: 2.1963 - val_accuracy: 0.5279
Epoch 130/150
86/86 [==============================] - 3s 31ms/step - loss: 0.2731 - accuracy:
0.9190 - val_loss: 2.8301 - val_accuracy: 0.4795
Epoch 131/150
86/86 [==============================] - 3s 32ms/step - loss: 0.3916 - accuracy:
0.8746 - val_loss: 3.2845 - val_accuracy: 0.3930
Epoch 132/150
86/86 [==============================] - 3s 31ms/step - loss: 0.4558 - accuracy:
0.8600 - val_loss: 3.1374 - val_accuracy: 0.4252
Epoch 133/150
86/86 [==============================] - 3s 32ms/step - loss: 0.3047 - accuracy:
0.9109 - val_loss: 2.8763 - val_accuracy: 0.4326
Epoch 134/150
86/86 [==============================] - 3s 31ms/step - loss: 0.3333 - accuracy:
0.8941 - val_loss: 7.1154 - val_accuracy: 0.1877
Epoch 135/150
86/86 [==============================] - 3s 33ms/step - loss: 0.8380 - accuracy:
0.7837 - val_loss: 2.1221 - val_accuracy: 0.5293
Epoch 136/150
86/86 [==============================] - 3s 31ms/step - loss: 0.2697 - accuracy:
0.9212 - val_loss: 4.1337 - val_accuracy: 0.3930
Epoch 137/150
86/86 [==============================] - 3s 31ms/step - loss: 0.3748 - accuracy:
0.8834 - val_loss: 2.2416 - val_accuracy: 0.5484
Epoch 138/150
86/86 [==============================] - 3s 31ms/step - loss: 0.7172 - accuracy:
0.8149 - val_loss: 2.4296 - val_accuracy: 0.4912
Epoch 139/150
86/86 [==============================] - 3s 31ms/step - loss: 0.2512 - accuracy:
0.9256 - val_loss: 2.2887 - val_accuracy: 0.5543
Epoch 140/150
86/86 [==============================] - 3s 31ms/step - loss: 0.2005 - accuracy:
0.9457 - val_loss: 4.0899 - val_accuracy: 0.3988
```

```
Epoch 141/150
86/86 [==============================] - 3s 31ms/step - loss: 0.5026 - accuracy:
0.8794 - val_loss: 2.2177 - val_accuracy: 0.5587
Epoch 142/150
86/86 [==============================] - 3s 31ms/step - loss: 0.1771 - accuracy:
0.9498 - val_loss: 3.3524 - val_accuracy: 0.4589
Epoch 143/150
86/86 [==============================] - 3s 32ms/step - loss: 0.5292 - accuracy:
0.8512 - val_loss: 2.2892 - val_accuracy: 0.5176
Epoch 144/150
86/86 [==============================] - 3s 31ms/step - loss: 0.1859 - accuracy:
0.9501 - val_loss: 2.4274 - val_accuracy: 0.5440
Epoch 145/150
86/86 [==============================] - 3s 31ms/step - loss: 0.1556 - accuracy:
0.9586 - val_loss: 2.5070 - val_accuracy: 0.5587
Epoch 146/150
86/86 [==============================] - 3s 31ms/step - loss: 0.1495 - accuracy:
0.9608 - val_loss: 2.3368 - val_accuracy: 0.5718
Epoch 147/150
86/86 [==============================] - 3s 32ms/step - loss: 0.1551 - accuracy:
0.9575 - val_loss: 3.1436 - val_accuracy: 0.4633
Epoch 148/150
86/86 [==============================] - 3s 34ms/step - loss: 0.4842 - accuracy:
0.8695 - val_loss: 2.2966 - val_accuracy: 0.5630
Epoch 149/150
86/86 [==============================] - 3s 31ms/step - loss: 0.1592 - accuracy:
0.9578 - val_loss: 3.1963 - val_accuracy: 0.4472
Epoch 150/150
86/86 [==============================] - 3s 32ms/step - loss: 0.5890 - accuracy:
0.8735 - val_loss: 2.3265 - val_accuracy: 0.5704
```

Plot the accuracy and loss of the model

In [ ]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2cac9b9e450>
```

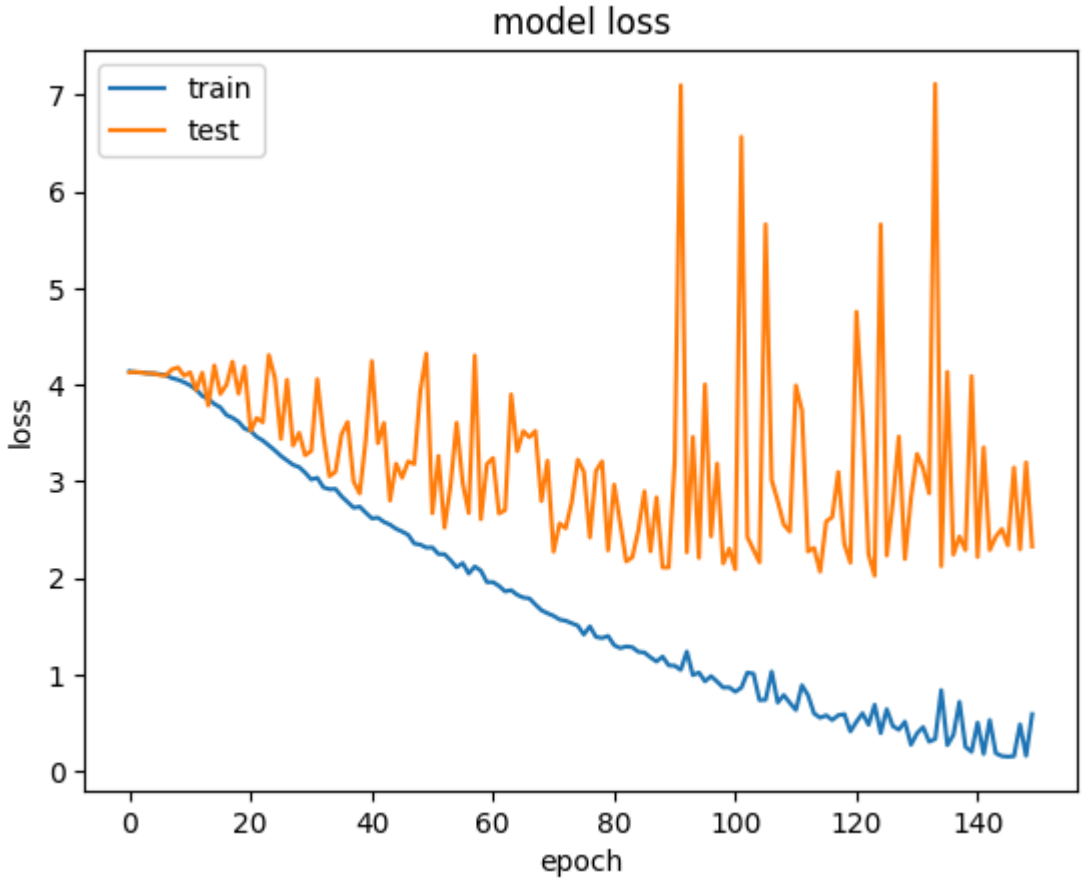


In [ ]:
```
print("Accuracy at 150 epochs: {:.2f}%".format(history.history['val_accuracy'][-
```

```
Accuracy at 150 epochs: 57.04%
```

In [ ]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2cacc845050>
```



In [ ]:
```
print("Loss at 150 epochs: {:.2f}".format(history.history['loss'][-1]))
```

```
Loss at 150 epochs: 0.59
```

Classify English Handwritten Characters through CNN

In [ ]:
```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('archive'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
archive\english.csv
archive\Img\img001-001.png
archive\Img\img001-002.png
archive\Img\img001-003.png
archive\Img\img001-004.png
archive\Img\img001-005.png
archive\Img\img001-006.png
archive\Img\img001-007.png
archive\Img\img001-008.png
archive\Img\img001-009.png
archive\Img\img001-010.png
archive\Img\img001-011.png
archive\Img\img001-012.png
archive\Img\img001-013.png
archive\Img\img001-014.png
archive\Img\img001-015.png
archive\Img\img001-016.png
archive\Img\img001-017.png
archive\Img\img001-018.png
archive\Img\img001-019.png
archive\Img\img001-020.png
archive\Img\img001-021.png
archive\Img\img001-022.png
archive\Img\img001-023.png
archive\Img\img001-024.png
archive\Img\img001-025.png
archive\Img\img001-026.png
archive\Img\img001-027.png
archive\Img\img001-028.png
archive\Img\img001-029.png
archive\Img\img001-030.png
archive\Img\img001-031.png
archive\Img\img001-032.png
archive\Img\img001-033.png
archive\Img\img001-034.png
archive\Img\img001-035.png
archive\Img\img001-036.png
archive\Img\img001-037.png
archive\Img\img001-038.png
archive\Img\img001-039.png
archive\Img\img001-040.png
archive\Img\img001-041.png
archive\Img\img001-042.png
archive\Img\img001-043.png
archive\Img\img001-044.png
archive\Img\img001-045.png
archive\Img\img001-046.png
archive\Img\img001-047.png
archive\Img\img001-048.png
archive\Img\img001-049.png
archive\Img\img001-050.png
archive\Img\img001-051.png
archive\Img\img001-052.png
archive\Img\img001-053.png
archive\Img\img001-054.png
archive\Img\img001-055.png
archive\Img\img002-001.png
archive\Img\img002-002.png
archive\Img\img002-003.png
archive\Img\img002-004.png
```

```
archive\Img\img062-005.png
archive\Img\img062-006.png
archive\Img\img062-007.png
archive\Img\img062-008.png
archive\Img\img062-009.png
archive\Img\img062-010.png
archive\Img\img062-011.png
archive\Img\img062-012.png
archive\Img\img062-013.png
archive\Img\img062-014.png
archive\Img\img062-015.png
archive\Img\img062-016.png
archive\Img\img062-017.png
archive\Img\img062-018.png
archive\Img\img062-019.png
archive\Img\img062-020.png
archive\Img\img062-021.png
archive\Img\img062-022.png
archive\Img\img062-023.png
archive\Img\img062-024.png
archive\Img\img062-025.png
archive\Img\img062-026.png
archive\Img\img062-027.png
archive\Img\img062-028.png
archive\Img\img062-029.png
archive\Img\img062-030.png
archive\Img\img062-031.png
archive\Img\img062-032.png
archive\Img\img062-033.png
archive\Img\img062-034.png
archive\Img\img062-035.png
archive\Img\img062-036.png
archive\Img\img062-037.png
archive\Img\img062-038.png
archive\Img\img062-039.png
archive\Img\img062-040.png
archive\Img\img062-041.png
archive\Img\img062-042.png
archive\Img\img062-043.png
archive\Img\img062-044.png
archive\Img\img062-045.png
archive\Img\img062-046.png
archive\Img\img062-047.png
archive\Img\img062-048.png
archive\Img\img062-049.png
archive\Img\img062-050.png
archive\Img\img062-051.png
archive\Img\img062-052.png
archive\Img\img062-053.png
archive\Img\img062-054.png
archive\Img\img062-055.png
```

Problem statement

The dataset contains 3410 images containing handwritten letters (0-9 numbers, a-z alphabets small and in caps) The goal is to train the model to recognize and predict the characters efficiently and categorize between 62 unique characters

I'm trying the classification through CNN

import the libraries

In [ ]:
```
import pandas
import random
import tensorflow as tf
from keras_preprocessing.image import ImageDataGenerator
import matplotlib.image as img
import matplotlib.pyplot as plt
```

Split the dataset

In this step, we'll split the data into 3 datasets - training set, validation test and test set
Out of total 3410 images, 2910 to training set, 490 added to validation set, 5 to test set
Removed the images added to validation, test set from training set to test its accuracy

In [ ]:
```
data_path = r"archive"

dataset = pandas.read_csv(data_path + '/english.csv')
rand = random.sample(range(len(dataset)), 500)
validation_set = pandas.DataFrame(dataset.iloc[rand, :].values, columns=['image'
# remove the added data
dataset.drop(rand, inplace=True)

rand = random.sample(range(len(validation_set)), 12)
test_set = pandas.DataFrame(validation_set.iloc[rand, :].values, columns=['image
# remove the added data
validation_set.drop(rand, inplace=True)

print(test_set)
```

```
                image label
0   Img/img004-043.png     3
1   Img/img029-039.png     S
2   Img/img023-048.png     M
3   Img/img006-017.png     5
4   Img/img020-027.png     J
5   Img/img020-043.png     J
6   Img/img023-046.png     M
7   Img/img059-028.png     w
8   Img/img060-029.png     x
9   Img/img009-020.png     8
10  Img/img027-050.png     Q
11  Img/img029-053.png     S
```

Data preprocessing

Now that the data is split, lets start with preprocessing step

Load the images through **flow_from_dataframe** method This method is convinient since the data file (english.csv) contains the image names along with the classification class details

In [ ]:
```
train_data_generator = ImageDataGenerator(rescale=1/255, shear_range=0.2, zoom_r
data_generator = ImageDataGenerator(rescale=1/255)
training_data_frame = train_data_generator.flow_from_dataframe(dataframe=dataset
                                                               target_size=(64,
validation_data_frame = data_generator.flow_from_dataframe(dataframe=validation_
                                                           target_size=(64, 64),
test_data_frame = data_generator.flow_from_dataframe(dataframe=test_set, directo
                                                     target_size=(64, 64), class
```

```
Found 2910 validated image filenames belonging to 62 classes.
Found 488 validated image filenames belonging to 62 classes.
Found 12 validated image filenames belonging to 9 classes.
```

Building the CNN model

We are about to build CNN model using libraries provided through **TensorFlow**

Code block breakdown:

- Create Convolution layer: to read/process the image, one feature or one part at a time
- Create Pooling layer: used to reduce the spatial size of convolved image
- Create Flattening layer: used to flatten the result, whose output would be the input for the neural network

We can create multiple convolution and pooling layer depending upon the need/complexity of the dataset

In [ ]:
```
cnn = tf.keras.models.Sequential()

# add convolutional and pooling layer
cnn.add(tf.keras.layers.Conv2D(filters=30, kernel_size=3, activation='relu', inp
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Conv2D(filters=30, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Conv2D(filters=30, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Flatten())
```

Building, compiling and training the neural network

From the above step we have received the flattened matrix of the images that we processed We are going to feed it to our neural network and train it

In this section, created fully connected Neural network aka Dense network, chosen sigmoid function for activation type In below the model will learn from the training set and predicts the data from validation set

The model accuracy improves as the epochs iteration progresses

In [ ]:
```
# add full connection, output layer
cnn.add(tf.keras.layers.Dense(units=600, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=62, activation='sigmoid'))

# compile cnn
cnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accurac
cnn.fit(x=training_data_frame, validation_data=validation_data_frame, epochs=30)
```

```
Epoch 1/30
91/91 [==============================] - 49s 523ms/step - loss: 3.7126 - accurac
y: 0.1058 - val_loss: 2.6056 - val_accuracy: 0.3525
Epoch 2/30
91/91 [==============================] - 35s 382ms/step - loss: 1.8850 - accurac
y: 0.5055 - val_loss: 1.5906 - val_accuracy: 0.5799
Epoch 3/30
91/91 [==============================] - 34s 370ms/step - loss: 1.1412 - accurac
y: 0.6845 - val_loss: 1.3657 - val_accuracy: 0.6639
Epoch 4/30
91/91 [==============================] - 35s 380ms/step - loss: 0.8307 - accurac
y: 0.7491 - val_loss: 1.3678 - val_accuracy: 0.6455
Epoch 5/30
91/91 [==============================] - 34s 376ms/step - loss: 0.6413 - accurac
y: 0.7997 - val_loss: 1.2448 - val_accuracy: 0.6783
Epoch 6/30
91/91 [==============================] - 36s 399ms/step - loss: 0.4814 - accurac
y: 0.8546 - val_loss: 1.2550 - val_accuracy: 0.6885
Epoch 7/30
91/91 [==============================] - 36s 393ms/step - loss: 0.3842 - accurac
y: 0.8777 - val_loss: 1.2418 - val_accuracy: 0.7234
Epoch 8/30
91/91 [==============================] - 37s 407ms/step - loss: 0.2803 - accurac
y: 0.9117 - val_loss: 1.2328 - val_accuracy: 0.7070
Epoch 9/30
91/91 [==============================] - 36s 399ms/step - loss: 0.2674 - accurac
y: 0.9137 - val_loss: 1.3054 - val_accuracy: 0.7090
Epoch 10/30
91/91 [==============================] - 34s 378ms/step - loss: 0.2088 - accurac
y: 0.9357 - val_loss: 1.2232 - val_accuracy: 0.7418
Epoch 11/30
91/91 [==============================] - 36s 396ms/step - loss: 0.2107 - accurac
y: 0.9344 - val_loss: 1.3556 - val_accuracy: 0.7070
Epoch 12/30
91/91 [==============================] - 39s 435ms/step - loss: 0.1824 - accurac
y: 0.9419 - val_loss: 1.4141 - val_accuracy: 0.6967
Epoch 13/30
91/91 [==============================] - 46s 502ms/step - loss: 0.1518 - accurac
y: 0.9522 - val_loss: 1.5065 - val_accuracy: 0.7275
Epoch 14/30
91/91 [==============================] - 52s 573ms/step - loss: 0.1732 - accurac
y: 0.9385 - val_loss: 1.2945 - val_accuracy: 0.7131
Epoch 15/30
91/91 [==============================] - 63s 692ms/step - loss: 0.1509 - accurac
y: 0.9519 - val_loss: 1.3216 - val_accuracy: 0.7172
Epoch 16/30
91/91 [==============================] - 52s 570ms/step - loss: 0.1179 - accurac
y: 0.9605 - val_loss: 1.6675 - val_accuracy: 0.7029
Epoch 17/30
91/91 [==============================] - 38s 415ms/step - loss: 0.1285 - accurac
y: 0.9581 - val_loss: 1.6104 - val_accuracy: 0.7070
Epoch 18/30
91/91 [==============================] - 43s 471ms/step - loss: 0.1058 - accurac
y: 0.9653 - val_loss: 1.5495 - val_accuracy: 0.7295
Epoch 19/30
91/91 [==============================] - 43s 479ms/step - loss: 0.1103 - accurac
y: 0.9643 - val_loss: 1.6781 - val_accuracy: 0.7193
Epoch 20/30
91/91 [==============================] - 34s 371ms/step - loss: 0.1255 - accurac
y: 0.9625 - val_loss: 1.4071 - val_accuracy: 0.7336
```

```
Epoch 21/30
91/91 [==============================] - 43s 471ms/step - loss: 0.1093 - accurac
y: 0.9656 - val_loss: 1.7305 - val_accuracy: 0.7254
Epoch 22/30
91/91 [==============================] - 50s 556ms/step - loss: 0.0947 - accurac
y: 0.9722 - val_loss: 1.5441 - val_accuracy: 0.7439
Epoch 23/30
91/91 [==============================] - 42s 465ms/step - loss: 0.1103 - accurac
y: 0.9649 - val_loss: 1.7748 - val_accuracy: 0.7172
Epoch 24/30
91/91 [==============================] - 47s 514ms/step - loss: 0.1093 - accurac
y: 0.9663 - val_loss: 1.7726 - val_accuracy: 0.7254
Epoch 25/30
91/91 [==============================] - 50s 548ms/step - loss: 0.0823 - accurac
y: 0.9711 - val_loss: 1.7765 - val_accuracy: 0.6988
Epoch 26/30
91/91 [==============================] - 48s 528ms/step - loss: 0.0872 - accurac
y: 0.9759 - val_loss: 1.5370 - val_accuracy: 0.7377
Epoch 27/30
91/91 [==============================] - 46s 506ms/step - loss: 0.0770 - accurac
y: 0.9766 - val_loss: 1.5058 - val_accuracy: 0.7459
Epoch 28/30
91/91 [==============================] - 38s 422ms/step - loss: 0.0713 - accurac
y: 0.9753 - val_loss: 1.7650 - val_accuracy: 0.7172
Epoch 29/30
91/91 [==============================] - 44s 482ms/step - loss: 0.0555 - accurac
y: 0.9804 - val_loss: 1.6092 - val_accuracy: 0.7254
Epoch 30/30
91/91 [==============================] - 46s 512ms/step - loss: 0.0646 - accurac
y: 0.9804 - val_loss: 1.7664 - val_accuracy: 0.7131
```

Out[ ]: <keras.src.callbacks.History at 0x2cac9a80150>

Predicting the testset images

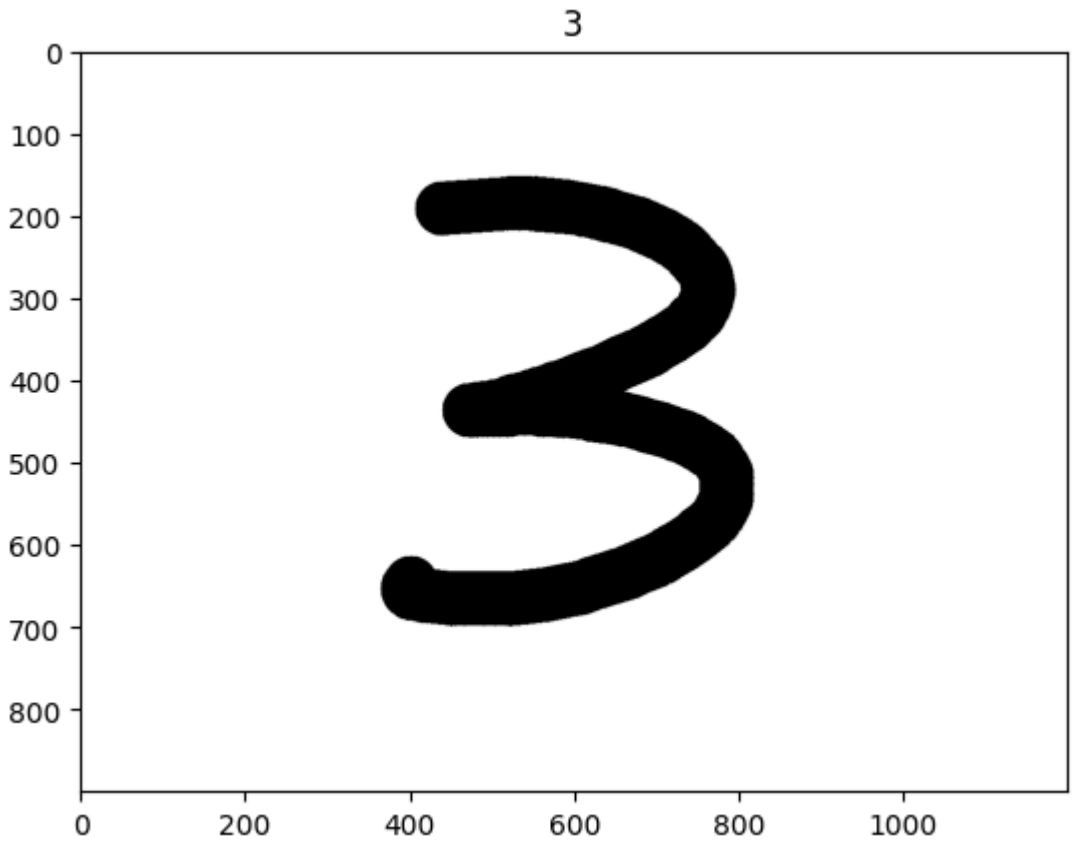
Since the model is trained, lets pass the testset images and see how well our model predicts class_indices function gives us the neural network mapping for our 62 characters

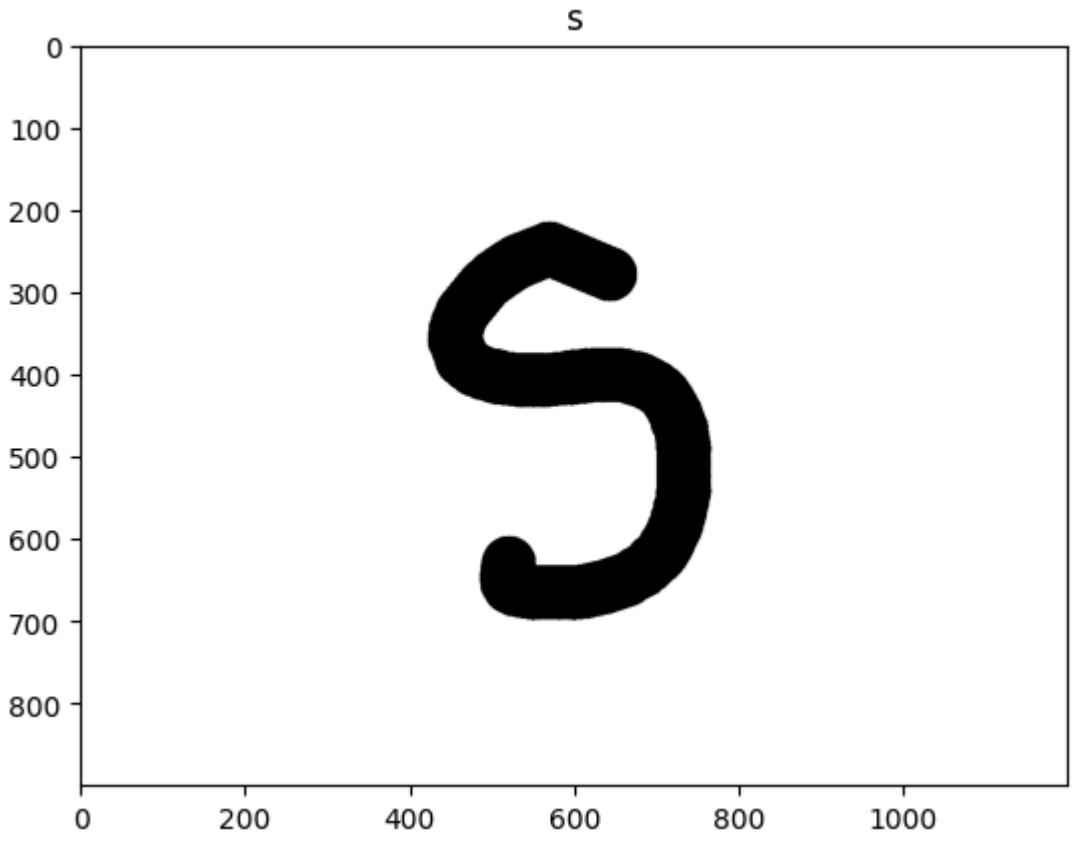
The result image's name is the predicted character by our model

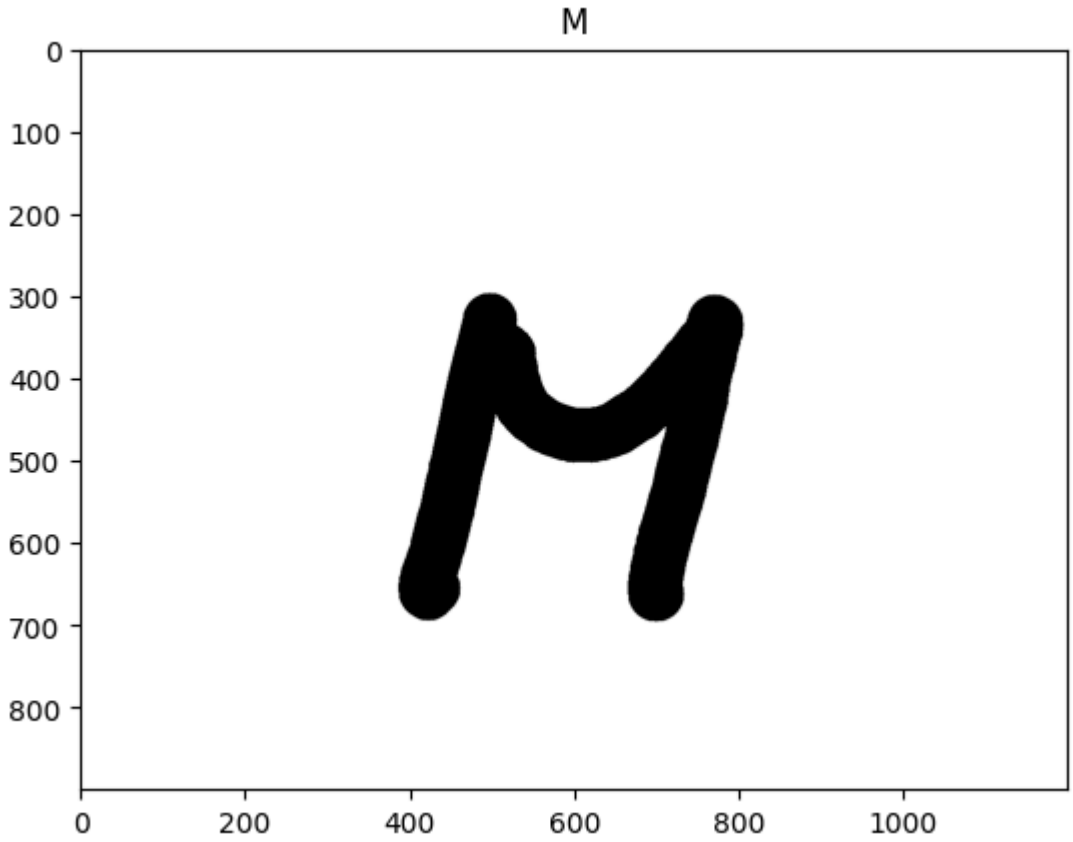
In [ ]:
```
print("Prediction mapping: ", training_data_frame.class_indices)
pred = cnn.predict(test_data_frame)

# switcher shows our network mapping to the prediction
switcher = {
            0: "0", 1: "1", 2: "2", 3: "3", 4: "4", 5: "5", 6: "6", 7: "7", 8: "
            11: "B", 12: "C", 13: "D", 14: "E", 15: "F", 16: "G", 17: "H", 18: "
            21: "L", 22: "M", 23: "N", 24: "O", 25: "P", 26: "Q", 27: "R", 28: "
            31: "V", 32: "W", 33: "X", 34: "Y", 35: "Z", 36: "a", 37: "b", 38: "
            41: "f", 42: "g", 43: "h", 44: "i", 45: "j", 46: "k", 47: "l", 48: "
            51: "p", 52: "q", 53: "r", 54: "s", 55: "t", 56: "u", 57: "v", 58: "
            61: "z"}

outputDf = pandas.DataFrame(pred)
maxIndex = list(outputDf.idxmax(axis=1))
print("Max index: ", maxIndex)
for i in range(len(test_set)):
```

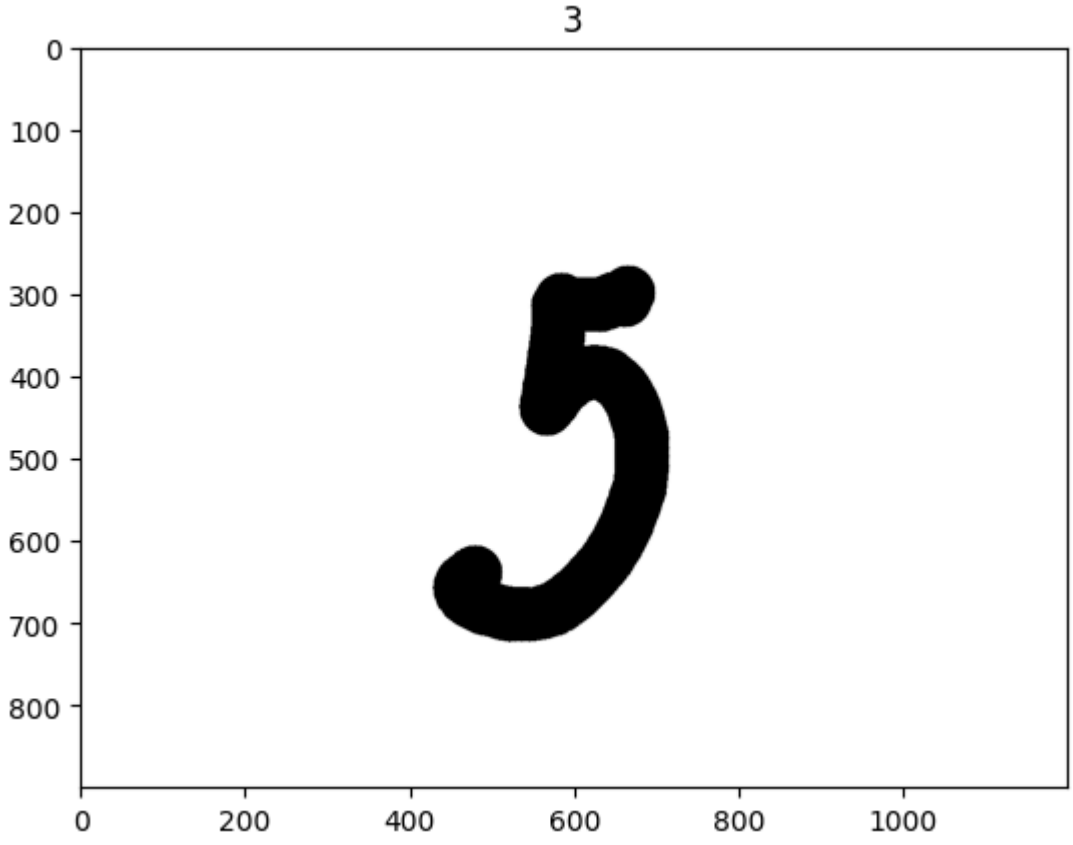
```
    image = img.imread(data_path + '/' + test_set.at[i, 'image'])
    plt.title(switcher.get(maxIndex[i], "error"))
    plt.imshow(image)
    plt.show()
```

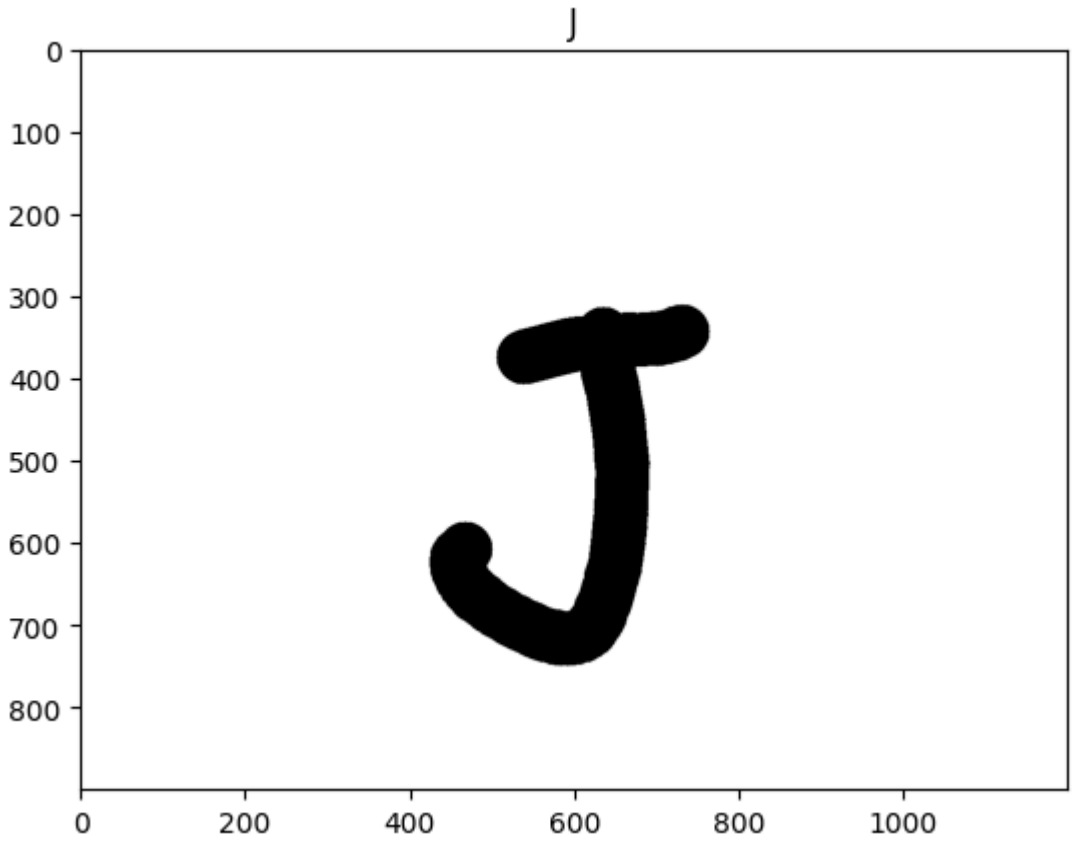
```
Prediction mapping:  {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6,
'7': 7, '8': 8, '9': 9, 'A': 10, 'B': 11, 'C': 12, 'D': 13, 'E': 14, 'F': 15,
'G': 16, 'H': 17, 'I': 18, 'J': 19, 'K': 20, 'L': 21, 'M': 22, 'N': 23, 'O': 24,
'P': 25, 'Q': 26, 'R': 27, 'S': 28, 'T': 29, 'U': 30, 'V': 31, 'W': 32, 'X': 33,
'Y': 34, 'Z': 35, 'a': 36, 'b': 37, 'c': 38, 'd': 39, 'e': 40, 'f': 41, 'g': 42,
'h': 43, 'i': 44, 'j': 45, 'k': 46, 'l': 47, 'm': 48, 'n': 49, 'o': 50, 'p': 51,
'q': 52, 'r': 53, 's': 54, 't': 55, 'u': 56, 'v': 57, 'w': 58, 'x': 59, 'y': 60,
'z': 61}
1/1 [==============================] - 0s 246ms/step
Max index:  [3, 54, 22, 3, 19, 19, 22, 58, 59, 8, 26, 28]
```

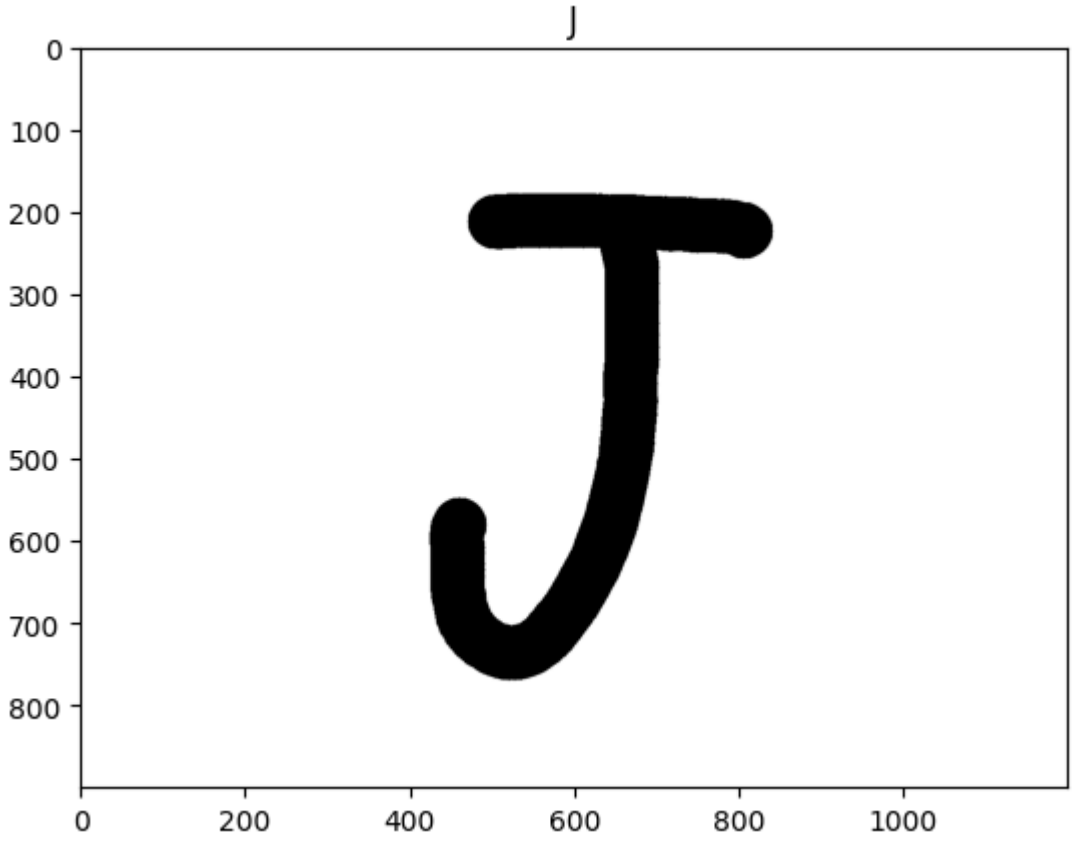


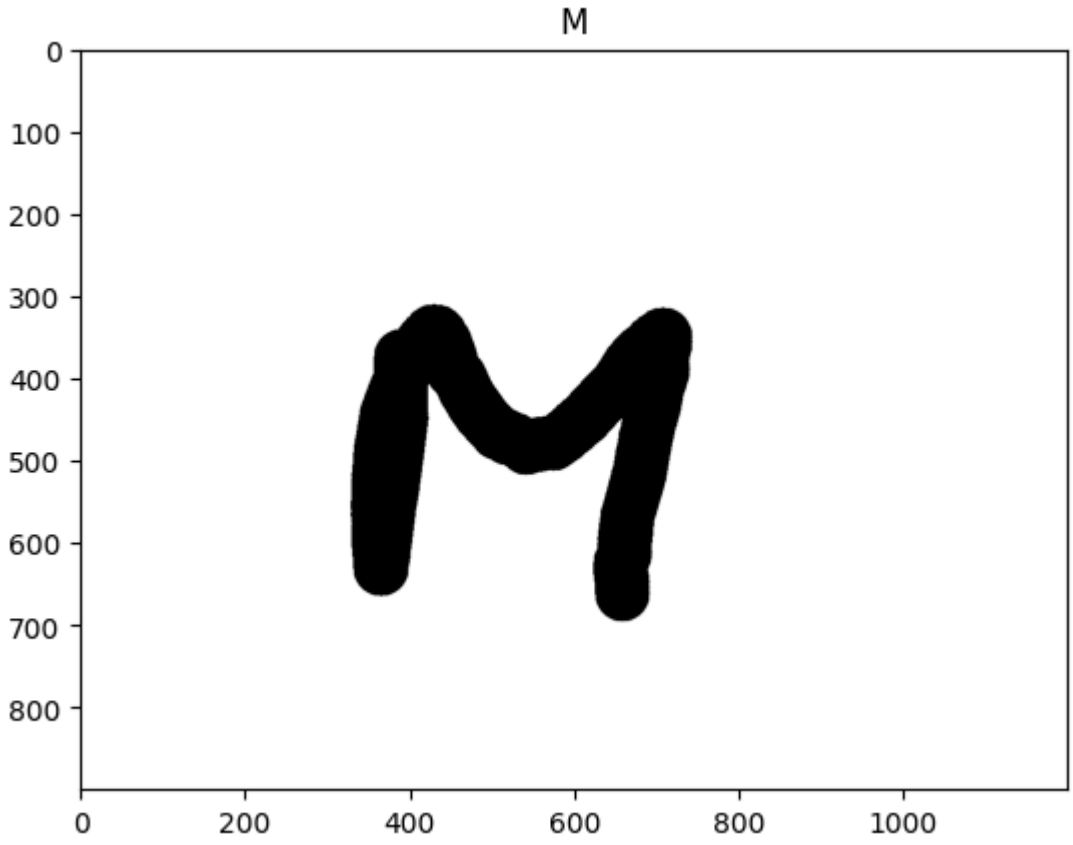

S
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

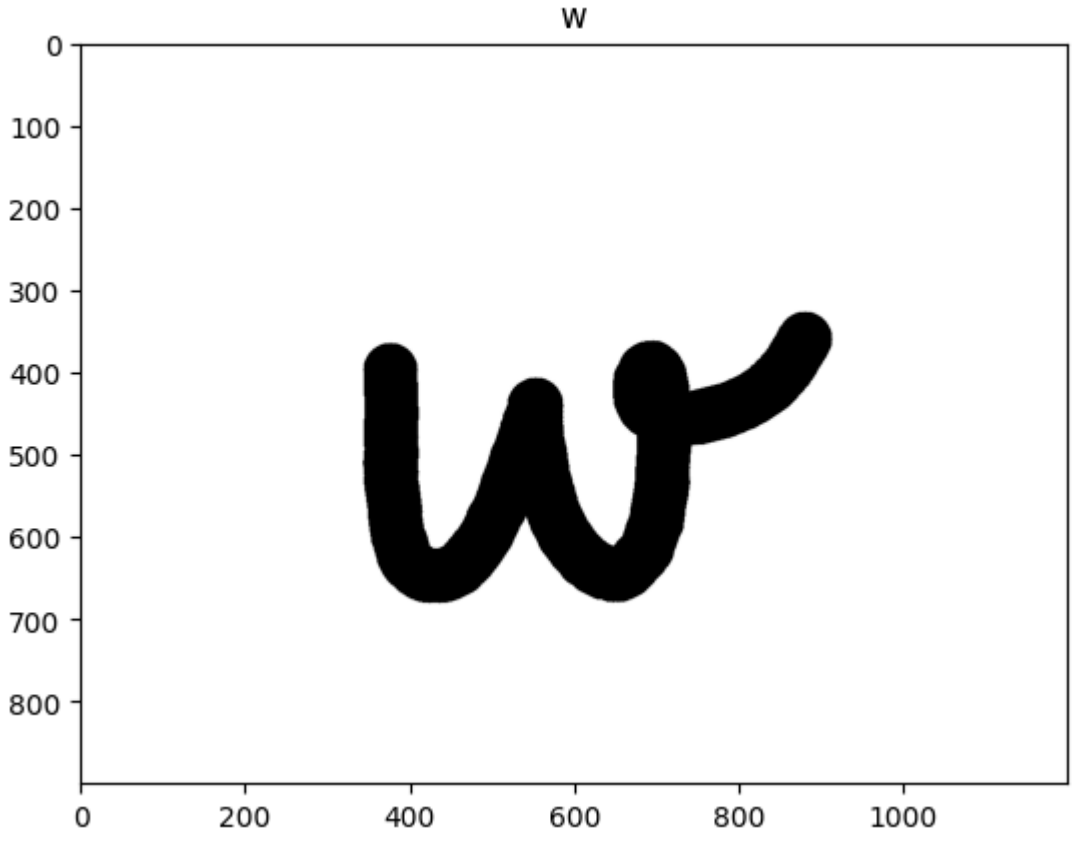

M
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

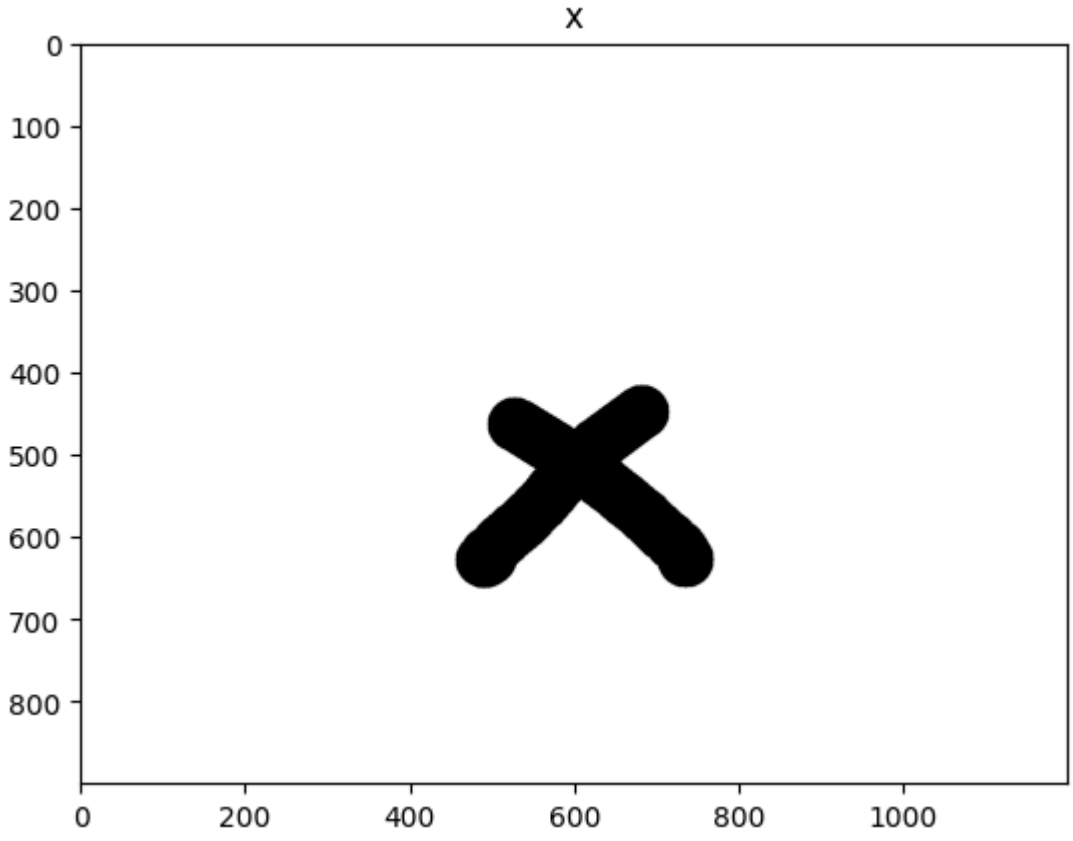

3
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

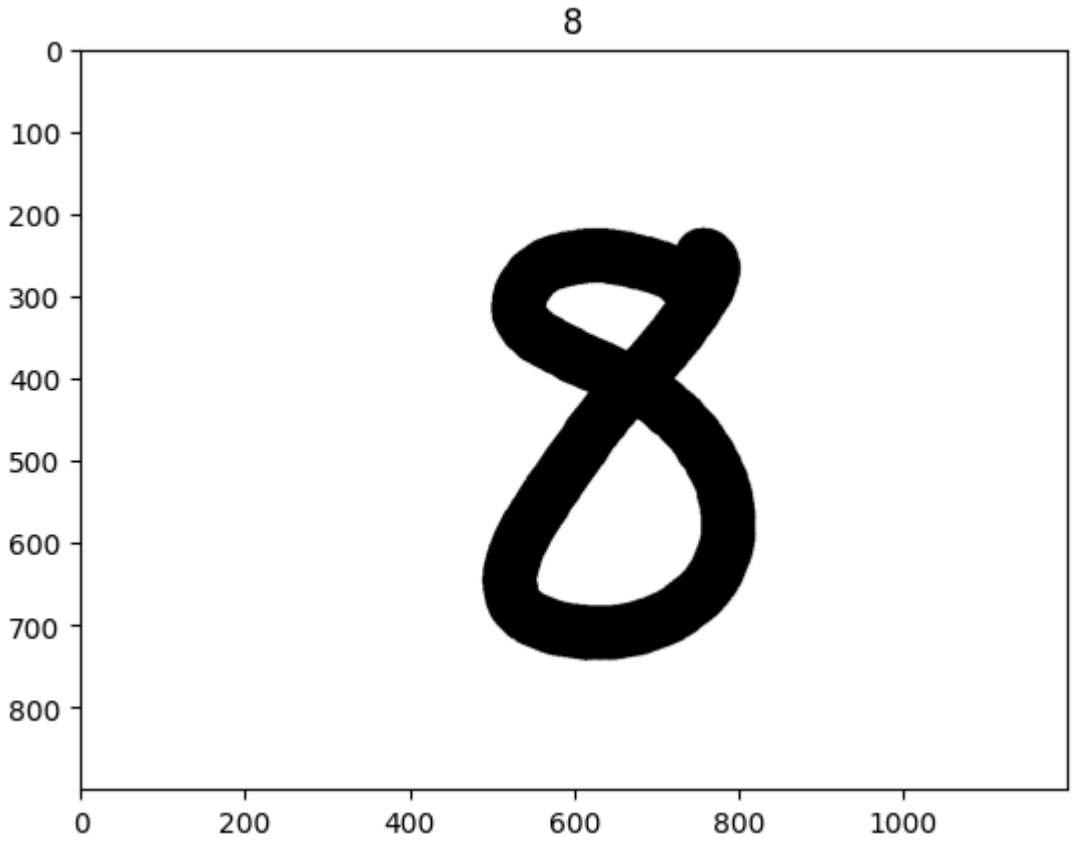

J
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

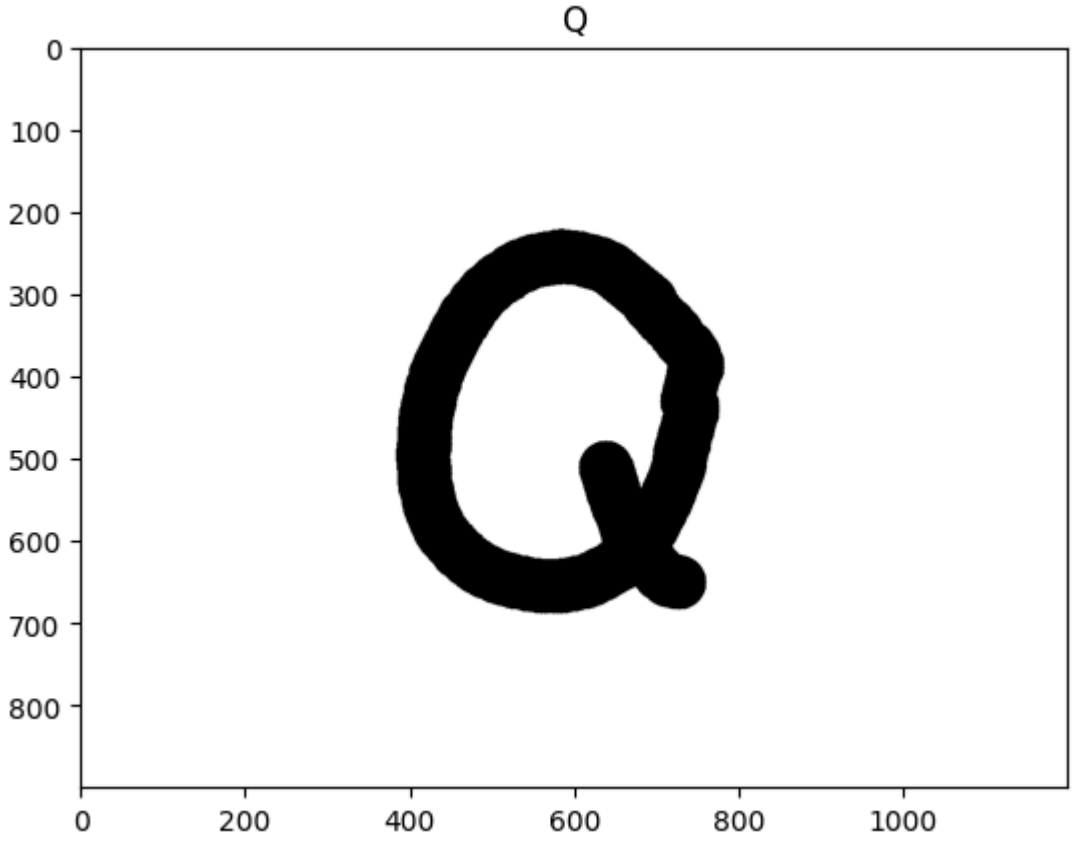

J
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

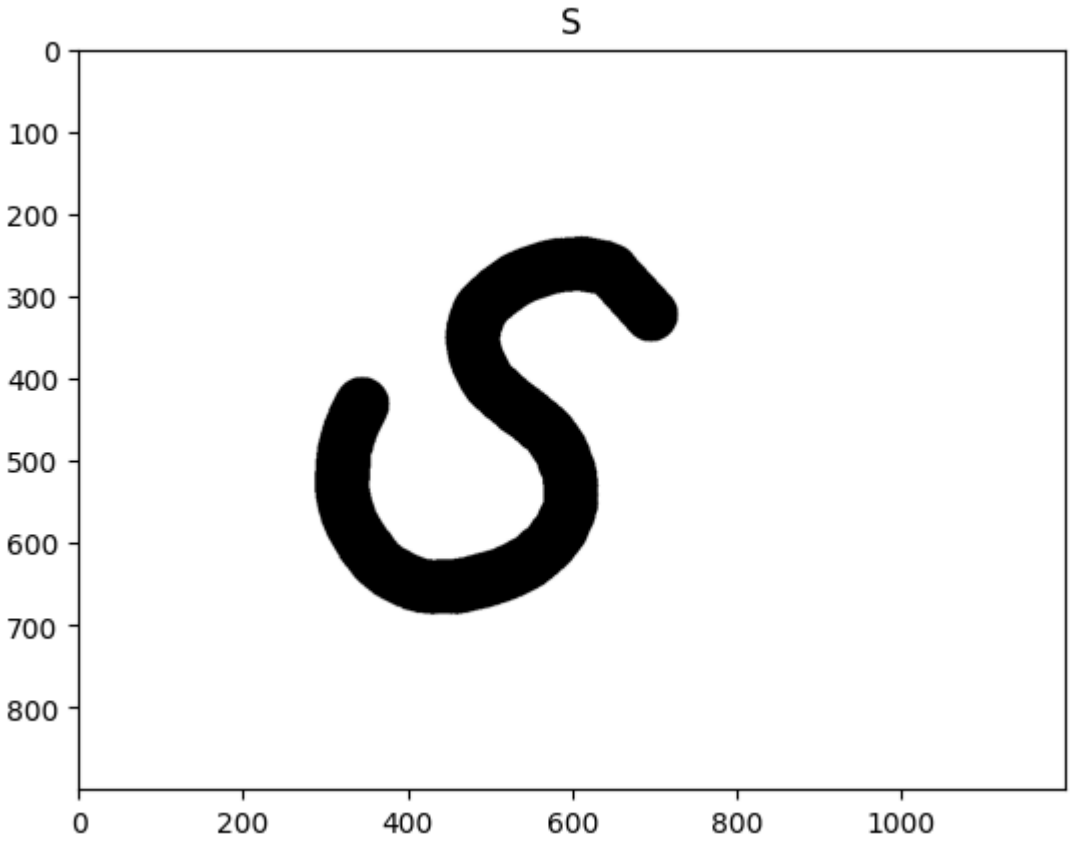

M
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


W
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


X
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


8
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


Q
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000



In [ ]:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import numpy as np

# Function to convert labels to one-hot encoding
def convert_to_one_hot(labels, num_classes):
    one_hot_labels = np.zeros((len(labels), num_classes))
    for i in range(len(labels)):
        one_hot_labels[i, labels[i]] = 1
    return one_hot_labels

# Convert labels to one-hot encoding for training and test sets
train_labels_one_hot = convert_to_one_hot(training_data_frame.classes, 62)
test_labels_one_hot = convert_to_one_hot(test_data_frame.classes, 62)

# Predict probabilities for the test set
test_pred_prob = cnn.predict(test_data_frame)

# Calculate ROC curve and AUC for each class
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(62):
    fpr[i], tpr[i], _ = roc_curve(test_labels_one_hot[:, i], test_pred_prob[:, i
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot ROC curve for each class
plt.figure(figsize=(10, 8))
for i in range(62):
    plt.plot(fpr[i], tpr[i], label='Class {} (AUC = {:.2f})'.format(i, roc_auc[i

plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random')
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for each class')
plt.legend()
plt.show()
```

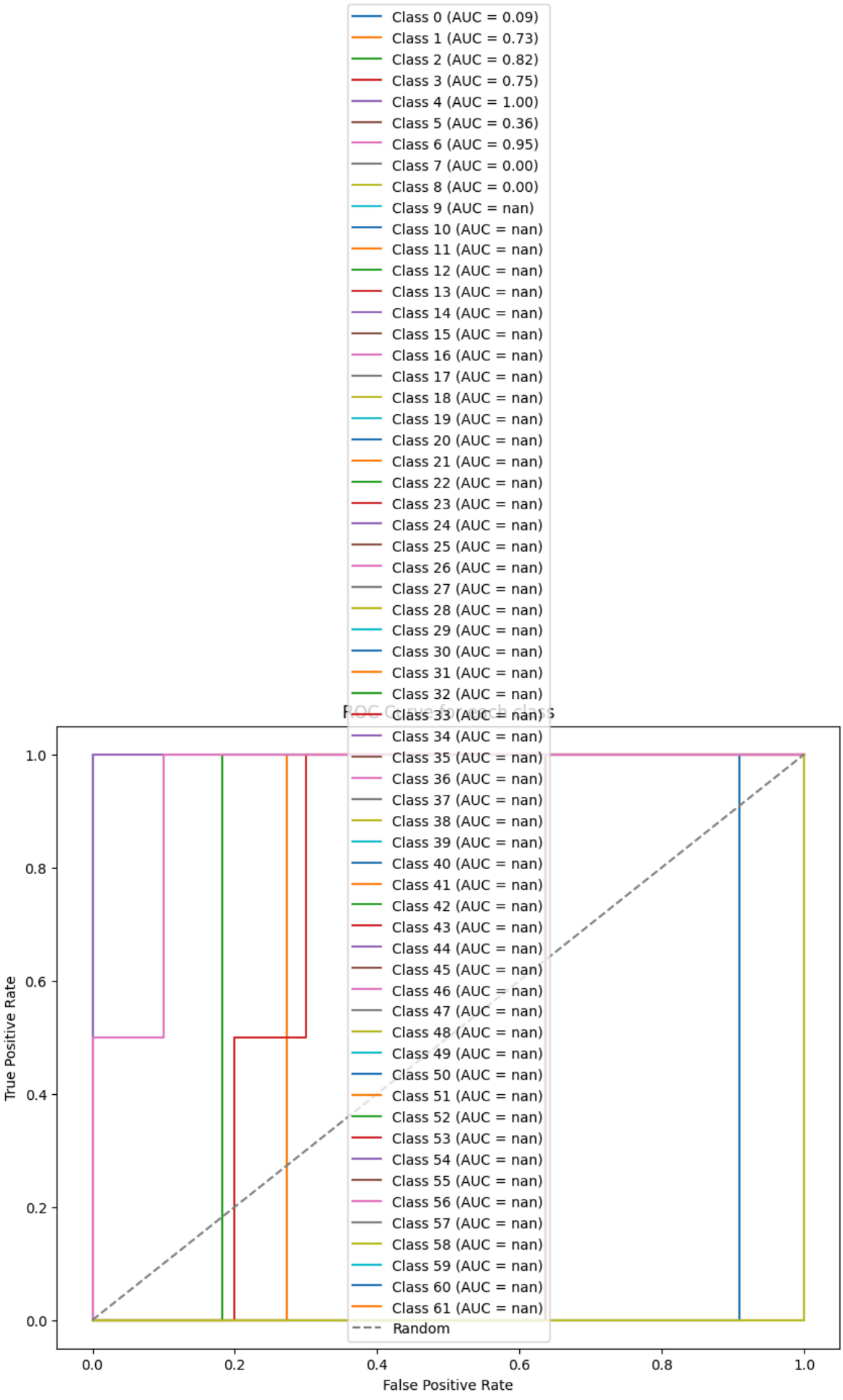
```
1/1 [==============================] - 0s 153ms/step
```

```
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
```

```
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
```

```
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
```

```
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
```



In [ ]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
```

```python
import numpy as np

def convert_to_one_hot(labels, num_classes):
    one_hot_labels = np.zeros((len(labels), num_classes))
    for i in range(len(labels)):
        one_hot_labels[i, labels[i]] = 1
    return one_hot_labels

# Convert labels to one-hot encoding for training and test sets
train_labels_one_hot = convert_to_one_hot(training_data_frame.classes, 62)
test_labels_one_hot = convert_to_one_hot(test_data_frame.classes, 62)

# Predict probabilities for the test set
test_pred_prob = cnn.predict(test_data_frame)

# Calculate ROC curve and AUC for each class
fpr = dict()
tpr = dict()
roc_auc = dict()

plt.figure(figsize=(10, 8))

for i in range(62):
    fpr[i], tpr[i], _ = roc_curve(test_labels_one_hot[:, i], test_pred_prob[:, i
    roc_auc[i] = auc(fpr[i], tpr[i])

    # Plot ROC curve for each class
    plt.plot(fpr[i], tpr[i], label=f'Class {i} (AUC = {roc_auc[i]:.2f})')

# Plot the random classifier
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for each class')
plt.legend()
plt.show()
```

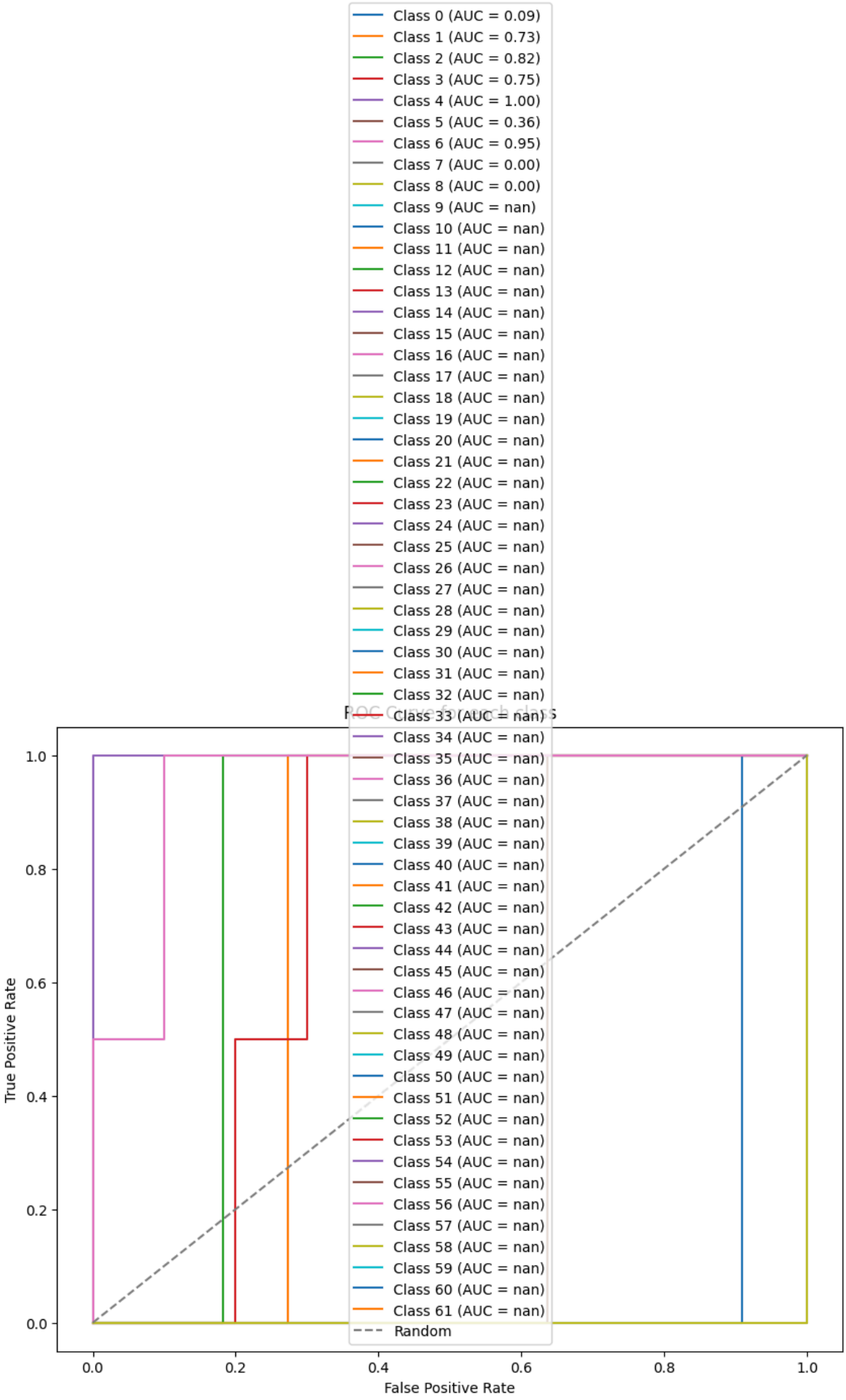
```
1/1 [==============================] - 0s 179ms/step
```

```
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
```

```
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
```

```
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
```

```
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
C:\Users\Ananya\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\metrics\_ranking.py:1132: UndefinedMetricWarning: No positive samples in y_true,
true positive value should be meaningless
  warnings.warn(
```


ROC Curve for each class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Class 0 (AUC = 0.09)
Class 1 (AUC = 0.73)
Class 2 (AUC = 0.82)
Class 3 (AUC = 0.75)
Class 4 (AUC = 1.00)
Class 5 (AUC = 0.36)
Class 6 (AUC = 0.95)
Class 7 (AUC = 0.00)
Class 8 (AUC = 0.00)
Class 9 (AUC = nan)
Class 10 (AUC = nan)
Class 11 (AUC = nan)
Class 12 (AUC = nan)
Class 13 (AUC = nan)
Class 14 (AUC = nan)
Class 15 (AUC = nan)
Class 16 (AUC = nan)
Class 17 (AUC = nan)
Class 18 (AUC = nan)
Class 19 (AUC = nan)
Class 20 (AUC = nan)
Class 21 (AUC = nan)
Class 22 (AUC = nan)
Class 23 (AUC = nan)
Class 24 (AUC = nan)
Class 25 (AUC = nan)
Class 26 (AUC = nan)
Class 27 (AUC = nan)
Class 28 (AUC = nan)
Class 29 (AUC = nan)
Class 30 (AUC = nan)
Class 31 (AUC = nan)
Class 32 (AUC = nan)
Class 33 (AUC = nan)
Class 34 (AUC = nan)
Class 35 (AUC = nan)
Class 36 (AUC = nan)
Class 37 (AUC = nan)
Class 38 (AUC = nan)
Class 39 (AUC = nan)
Class 40 (AUC = nan)
Class 41 (AUC = nan)
Class 42 (AUC = nan)
Class 43 (AUC = nan)
Class 44 (AUC = nan)
Class 45 (AUC = nan)
Class 46 (AUC = nan)
Class 47 (AUC = nan)
Class 48 (AUC = nan)
Class 49 (AUC = nan)
Class 50 (AUC = nan)
Class 51 (AUC = nan)
Class 52 (AUC = nan)
Class 53 (AUC = nan)
Class 54 (AUC = nan)
Class 55 (AUC = nan)
Class 56 (AUC = nan)
Class 57 (AUC = nan)
Class 58 (AUC = nan)
Class 59 (AUC = nan)
Class 60 (AUC = nan)
Class 61 (AUC = nan)
Random

Inference:

- Multi-Layer PLA yields around 57.04% accuracy over the test data
- CNN model yields around 92% accuracy over test data

Learning Outcome:

- Understood the importance of feature selection in machine learning.
- Gained proficiency in implementing and comparing classification algorithms.
- Developed skills in feature engineering to improve model performance.
- Learned how to tune hyperparameters effectively for better results.
- Developed critical thinking in evaluating model trade-offs for informed decision-making.