HANDWRITTEN CHARACTER RECOGNITION WITH NEURAL NETWORK

A Project done by students
Ms. Pratiksha Bagli
Ms. Sujata Bagli
Ms. Divya Gazinkar
Mr. Hanumant Godekar
Ms. Kajal Kataria

T.Y.MCA
2020-2021

PROJECT SUPERVISOR
Mr. S. Baskar

GOA UNIVERSITY

TABLE OF CONTENTS

INTRODUCTION

Image processing could be a manipulation of images within the computer vision. With the event of technology, there are many techniques for the manipulation of the photographs. The character recognition includes a huge role in many areas. But it's difficult to try and do such a task by a machine.

The objective of this project is to take handwritten English characters as input, process the character image, train the neural network algorithm to recognize the pattern or structure of characters, and match the character or the output with the desired input.

Pattern recognition is perhaps the most common use of neural networks.
The neural network is presented with an input which contains the pattern information, this could be an image, and handwritten data. The neural network then attempts to determine if the input data matches a pattern that the neural network has mentioned. A neural network trained for classification is designed to take input samples and classify them into groups. These input groups may be fuzzy, without clearly defined boundaries. This project concerns detecting free handwritten characters.

OVERVIEW OF THE PROJECT

This application is useful for recognizing all English alphabets given as in the input image. Once input image of character is given to proposed model, then it will recognize input character which is given as image.Recognition and classification of characters are done by Neural Network.The main aim of this project is to effectively recognize a particular character of type format using the Convolutional Neural Network approach.

DATASET

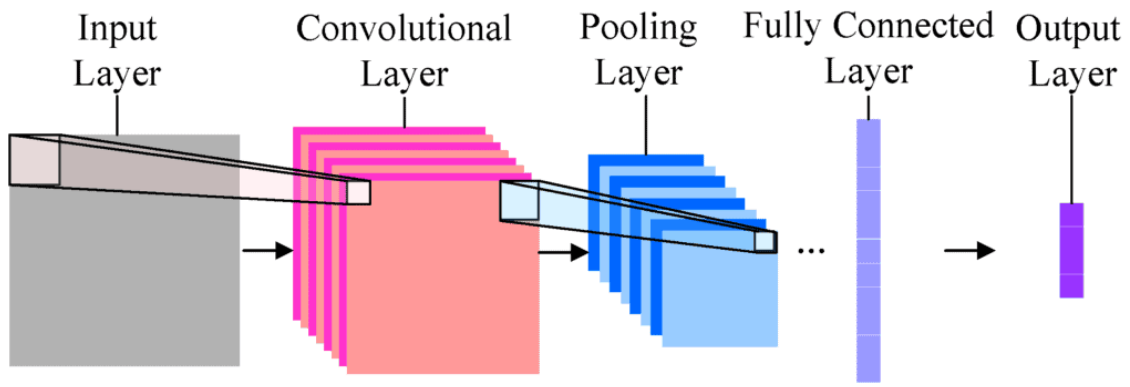
Nowadays Handwritten Character Recognition is a major remarkable and difficult research domain in the area of Image processing. Recognition of Handwritten English alphabets have been broadly studied in the previous years. Presently various recognition methodologies are in well-known utilized for recognition of handwritten English alphabets (character). Application domain of HCR is digital document processing such as mining information from data entry, cheque, applications for loans, credit cards, tax, health insurance forms etc. During this survey we present an outline of current research work conducted for recognition of handwritten English alphabets.

In Handwritten manuscript there is no restriction on the writing technique. Handwritten alphabets are complicated to recognize because of miscellaneous human handwriting technique, difference in size and shape of letters, angle. A variety of recognition methodologies for handwritten English alphabets are conferred here alongside with their performance.

In our project we chose the dataset from kaggle.The dataset contains 26 folders (A-Z) containing handwritten images in size 28 x 28 pixels, each alphabet in the image is centre fitted to the 20 x 20 pixel box. Each image is stored as Gray-level.

MODEL

We have applied a convolutional neural network model to handwritten character recognition. Convolutional Neural Networks are used to extract the features of the images using several layers of filters. Consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, RELU i.e. activation function, pooling layers, fully connected layers and softmax layers.



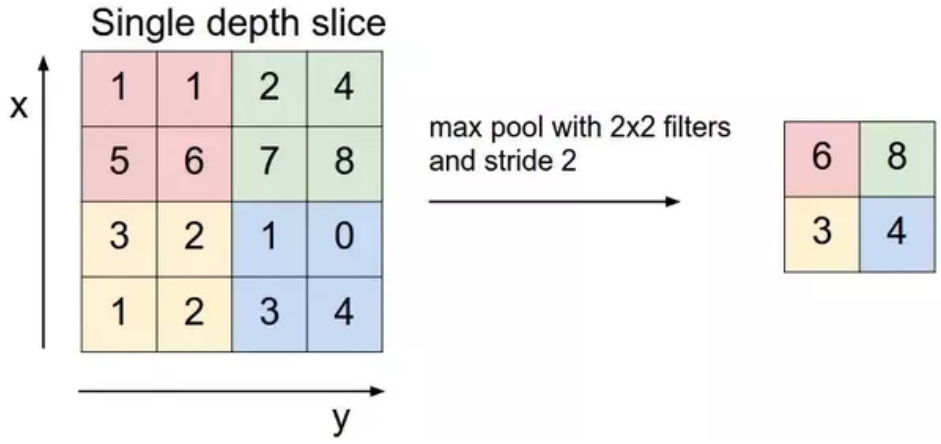
Convolution Layer :

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.
In our project we have used a convolution 2D layer.

Pooling Layer :

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling is also called subsampling or downsampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types: Max Pooling, Average Pooling, Sum Pooling. In

our project we have used maxpooling 2D .



Dropout Layer :

Dropout layers in CNN's is used to prevent **overfitting** by increasing testing accuracy.

Flatten Layer:

A layer that converts the 2D matrix data to a vector and feeds it into a fully connected layer like a neural network. It allows the output to be processed by standard fully connected layers.

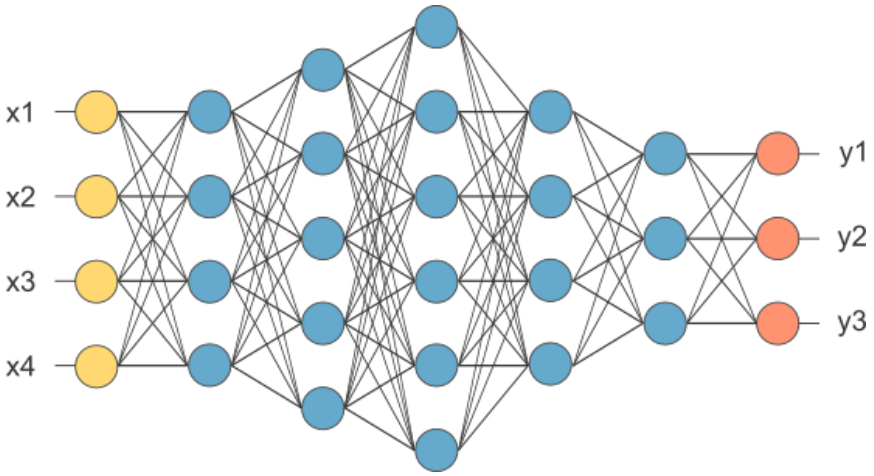


Figure : After pooling layer, flattened as FC layer

Dense layer :
A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected.

RELU :
Rectifier Linear Unit is an activation function.
The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. Thus it gives an output that has a range from 0 to infinity.

IMPLEMENTATION

Project Prerequisites

1. **Google Collab with GPU enabled**

Importing Required Libraries:

```
!pip install -U imbalanced-learn
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
from imblearn.under_sampling import NearMiss

from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix

import keras
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout ,
BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
import tensorflow as tf
%load_ext tensorboard
import datetime, os
```

Importing Data

```
from google.colab import drive
drive.mount('/content/drive')

alpha_data =
pd.read_csv(r"/content/drive/MyDrive/MLProject/A_Z_Handwritten.csv").a
stype('float32')

alpha_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 372450 entries, 0 to 372449
Columns: 785 entries, 0 to 0.648
dtypes: float32(785)
memory usage: 1.1 GB
```

```
alpha_data.head()
```

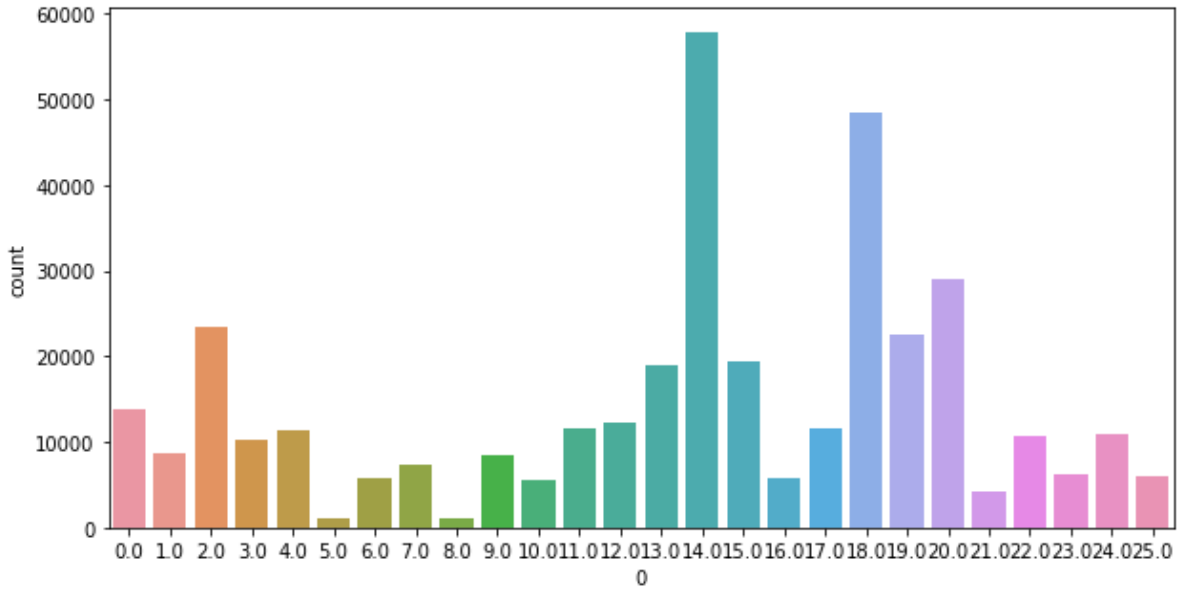
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 785 columns

```
# Separating dependent variable from independent variables
y = alpha_data['0']
del alpha_data['0']
```

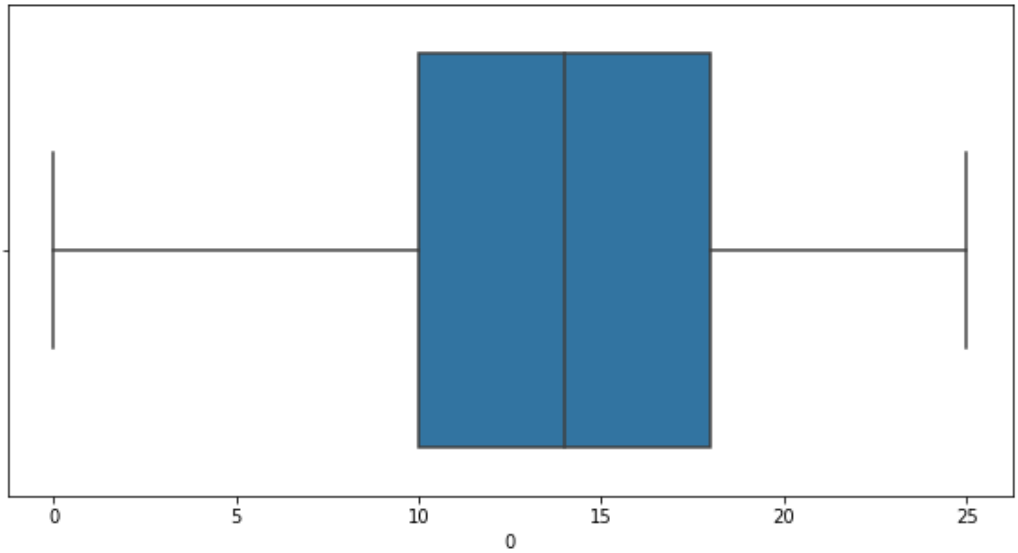
Data Insights

plt.figure(figsize = (10,5))
sns.countplot(y)



plt.figure(figsize = (10,5))
sns.boxplot(y)

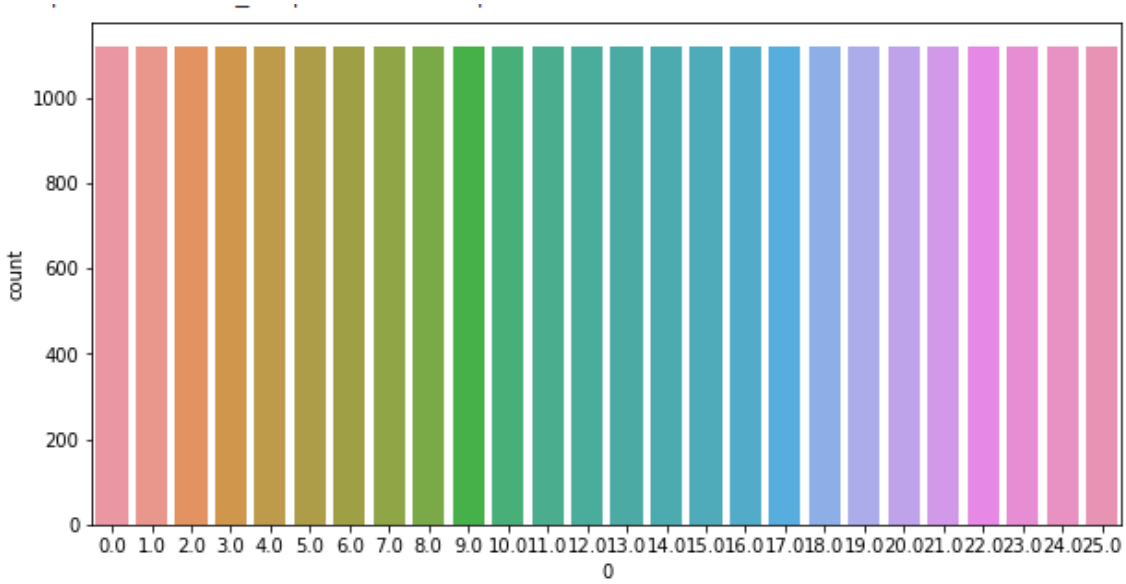
```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b2238b4a8>
```



Data Balancing

```
nM = NearMiss()
X_data, y_data = nM.fit_sample(alpha_data, y)
```

```
plt.figure(figsize = (10,5))
sns.countplot(y_data)
```



One hot encoding 'Labels'

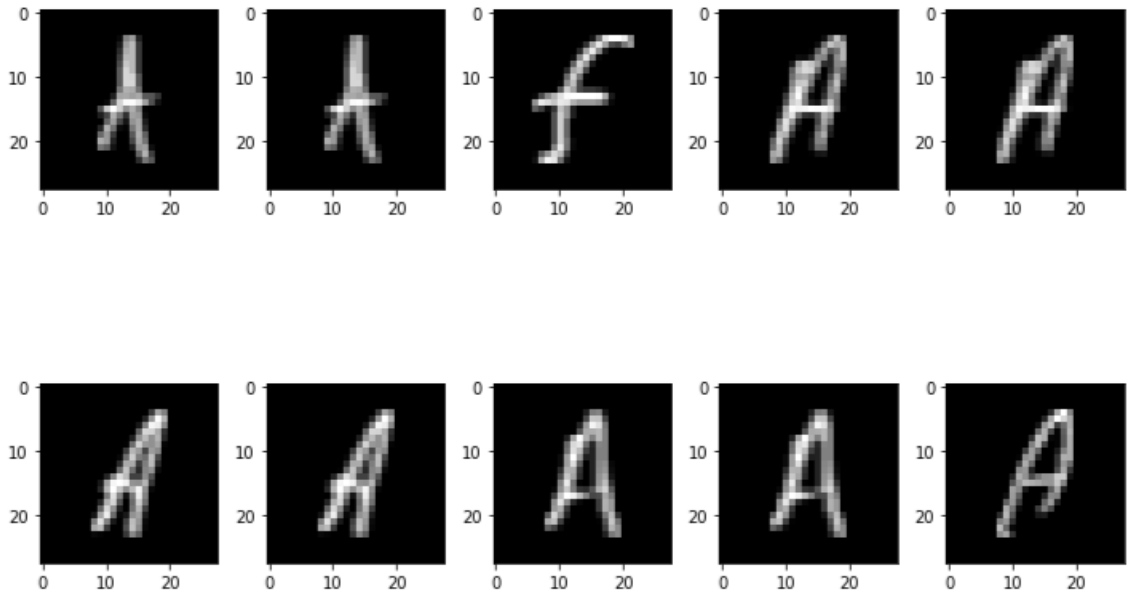
```
lB = LabelBinarizer()
y = lB.fit_transform(y_data)
y
```

```
X_data = X_data / 255
X_data
```

```
X_data = np.array(X_data)
X_data = X_data.reshape(-1,28,28,1)
```

```
# Showing few images
```

```
f, ax = plt.subplots(2,5)
f.set_size_inches(10,10)
k = 0
for i in range(2):
   for j in range(5):
      ax[i,j].imshow(X_data[k].reshape(28,28), cmap='gray')
      k += 1
   plt.tight_layout()
```



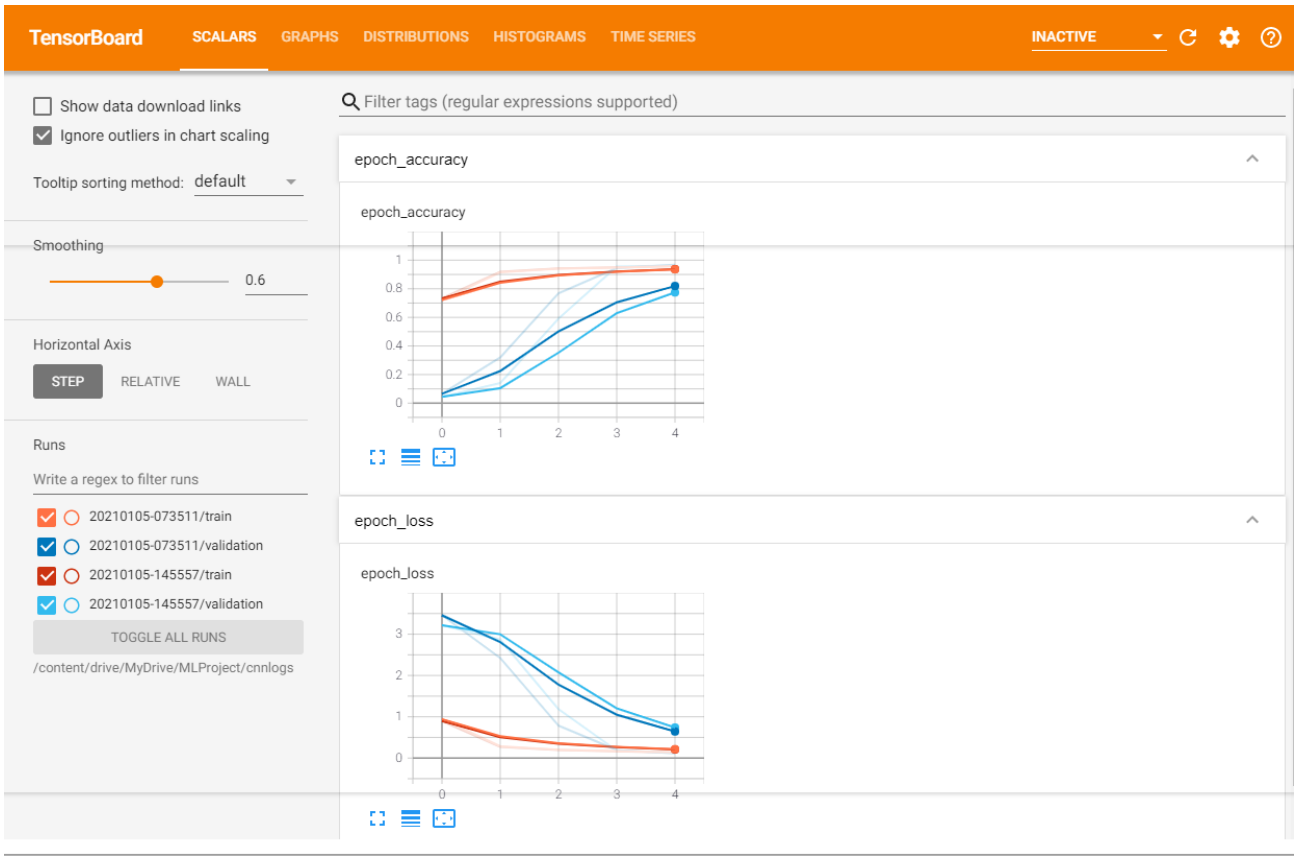
```
X_train, X_test, y_train, y_test = train_test_split(X_data, y, test_size=0.2)
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,
test_size=0.2)
X_train.shape, X_test.shape, X_valid.shape, y_train.shape, y_test.shape,
y_valid.shape
```

Data Augmentation

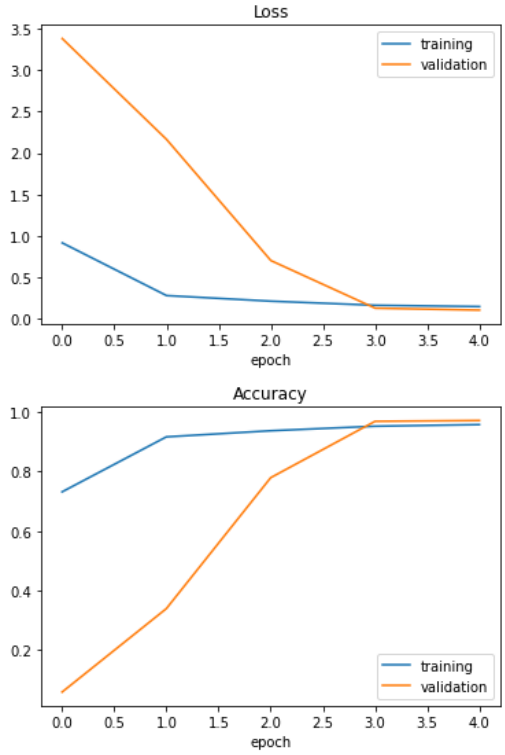
```
dataGen = ImageDataGenerator(rotation_range=10,
                             zoom_range=0.1,
                             width_shift_range=0.1,
                             height_shift_range=0.1)
dataGen.fit(X_train)

model = Sequential()

model.add(Conv2D(75 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu' ,
input_shape = (28,28,1)))
model.add(BatchNormalization())

model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(50 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(25 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(BatchNormalization())

model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Flatten())

model.add(Dense(units = 512 , activation = 'relu'))
model.add(Dropout(0.3))

model.add(Dense(units = 26 , activation = 'softmax'))

model.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics =
['accuracy'])
logdir = os.path.join("/content/drive/MyDrive/MLProject/cnnlogs",
datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
model.summary()
```

```
history = model.fit(dataGen.flow(X_train,y_train, batch_size = 128) ,epochs = 5 ,
                    validation_data = (X_valid, y_valid),
                    callbacks = [tensorboard_callback]
                    )
```

```
%tensorboard --logdir /content/drive/MyDrive/MLProject/cnnlogs
```



```
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('Loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.hiory['val_accuracy'])
plt.legend(['training','validation'])
plt.title('Accuracy')
plt.xlabel('epoch')
plt.show()
```



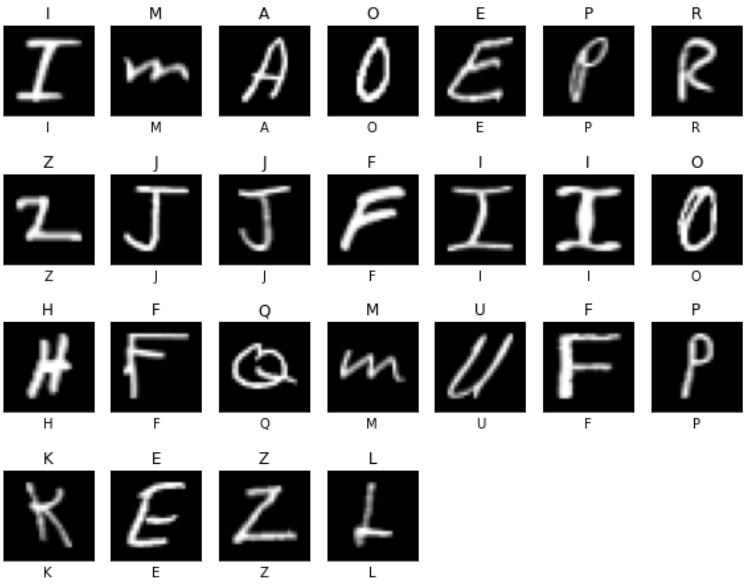
```
score = model.evaluate(X_test,y_test,verbose=0)
print('Test Score = ',score[0])
print('Test Accuracy =', score[1])
```

```
Test Score =  0.10473567992448807
Test Accuracy = 0.9737293720245361
```

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

Testing our Model

```
predictions = model.predict_classes(X_test,batch_size=128,verbose=0)
predictions[:5]

Y_test[0:5]

plt.figure(figsize=(10,10))
for i in range(25):
   plt.subplot(5,7,i+1)
   plt.xticks([])
   plt.yticks([])
   plt.grid(False)
   plt.imshow(X_test[i].reshape(28,28), cmap='gray')
   plt.title(className[np.argmax(y_test[i])])
   plt.xlabel(className[predictions[i]])
plt.show()
```




```
model.save(r'/content/drive/MyDrive/MLProject/Alpha.model')
```

Performance Metrics

from sklearn.metrics import confusion_matrix,accuracy_score
confusion_matrix(y_labels,predictions)

```
[77]  1  from sklearn.metrics import confusion_matrix,accuracy_score
      2  confusion_matrix(y_labels,predictions)
```

```
array([[202,   0,   0,   0,   1,   0,   1,   1,   0,   0,   2,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0, 221,   0,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   1,   0,   0,   0,   0,   0,   0,   0,   1],
       [  0,   1, 221,   1,   0,   0,   0,   0,   0,   0,   0,   1,   0,
          1,   1,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0, 219,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   1,   0,   0, 231,   0,   1,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   1,   0,   0,   0,   0,   1,   0,   1],
       [  0,   0,   0,   0,   2, 237,   1,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   4,   2,   0,   0,   0, 188,   2,   0,   0,   0,   0,   0,
          0,   2,   0,   6,   0,   0,   0,   0,   0,   3,   1,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0, 227,   0,   0,   1,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0, 217,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   1,   0,   0,   3,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   3, 224,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   3,   0,   0, 227,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   3,   0,   0],
       [  0,   0,   2,   1,   0,   0,   0,   0,   0,   0,   0,   0, 231,
          0,   1,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
        223,   1,   0,   0,   0,   0,   0,   0,   0,   4,   0,   0],
       [  1,   0,   0,   1,   0,   0,   0,   8,   0,   0,   1,   0,   3,
        208,   0,   0,   0,   0,   0,   0,   0,   0,   3,   4,   1,   0],
       [  3,   1,   0,   9,   0,   2,   0,   1,   0,   0,   0,   0,   0,
          0, 172,   0,   4,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  1,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0, 232,   0,   0,   0,   0,   0,   0,   0,   1,   0],
       [  1,   0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0,   0,
          0,   0,   0, 236,   0,   0,   0,   0,   0,   0,   0,   0],
       [  1,   0,   1,   0,   0,   0,   0,   0,   0,   0,   5,   0,   0,
          0,   0,   0,   0, 203,   0,   0,   0,   0,   0,   0,   0],
       [  0,   1,   0,   1,   0,   0,   3,   0,   0,   3,   0,   0,   0,
          0,   1,   0,   0,   0, 224,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0, 213,   0,   0,   0,   0,   0],
       [  0,   1,   0,   0,   0,   0,   0,   1,   0,   0,   0,   1,   0,
          1,   1,   0,   0,   0,   0,   1, 234,   0,   4,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0, 227,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0,
          1,   0,   0,   0,   0,   0,   0,   0,   0, 198,   0,   0],
       [  1,   0,   0,   0,   0,   0,   0,   2,   0,   0,   1,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   1,   0,   0, 210,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   1, 224],
       [  0,   1,   0,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   2,   0, 222]])
```

accuracy_score(y_labels,predictions)

from sklearn.metrics import classification_report

print(classification_report(y_labels,predictions))

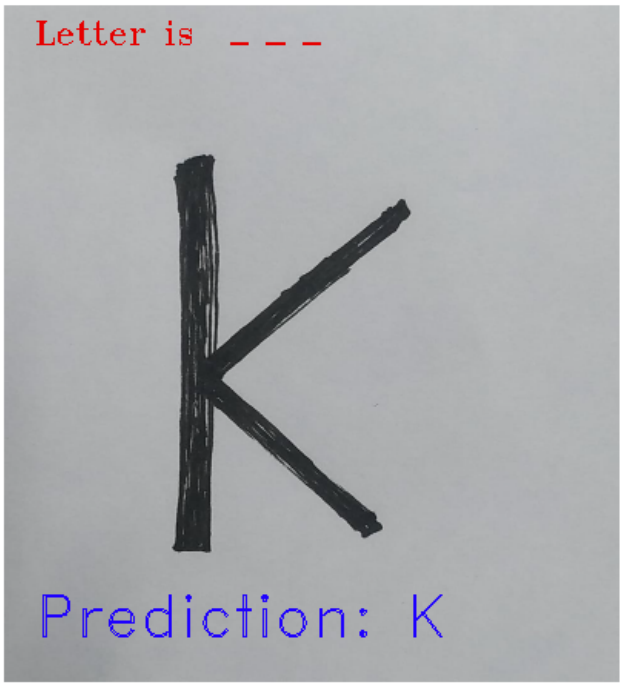
```
[78]  1  from sklearn.metrics import classification_report
      2  print(classification_report(y_labels,predictions))
```

	precision	recall	f1-score	support
0	0.96	0.98	0.97	207
1	0.95	0.99	0.97	224
2	0.98	0.97	0.98	227
3	0.94	1.00	0.96	220
4	0.99	0.98	0.98	236
5	0.99	0.99	0.99	240
6	0.96	0.90	0.93	208
7	0.92	1.00	0.96	228
8	0.99	0.98	0.98	221
9	0.99	0.99	0.99	227
10	0.96	0.97	0.97	233
11	0.99	0.98	0.99	236
12	0.99	0.98	0.98	228
13	0.98	0.90	0.94	230
14	0.97	0.90	0.93	192
15	1.00	0.99	0.99	235
16	0.96	0.99	0.97	238
17	1.00	0.97	0.98	210
18	1.00	0.96	0.98	233
19	0.99	1.00	1.00	213
20	0.99	0.96	0.97	244
21	1.00	1.00	1.00	228
22	0.93	0.99	0.96	200
23	0.93	0.98	0.95	215
24	0.99	1.00	0.99	225
25	0.99	0.98	0.99	226
accuracy			0.97	5824
macro avg	0.97	0.97	0.97	5824
weighted avg	0.97	0.97	0.97	5824

Prediction on external image

```
img = cv2.imread(r'/content/drive/MyDrive/MLProject/demo_K.jpg')
img_copy = img.copy()

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (400,440))

img_copy = cv2.GaussianBlur(img_copy, (7,7), 0)
img_gray = cv2.cvtColor(img_copy, cv2.COLOR_BGR2GRAY)
_, img_thresh = cv2.threshold(img_gray, 100, 255,
cv2.THRESH_BINARY_INV)

img_final = cv2.resize(img_thresh, (28,28))
img_final =np.reshape(img_final, (1,28,28,1))
img_pred = className[np.argmax(model.predict(img_final))]

cv2.putText(img, "Letter is  _ _ _ ", (20,25), cv2.FONT_HERSHEY_TRIPLEX,
0.7, color = (0,0,230))
```

```
cv2.putText(img, "Prediction: " + img_pred, (20,410),
cv2.FONT_HERSHEY_DUPLEX, 1.3, color = (255,0,30))
cv2_imshow(img)
```



Explanation of Libraries and Method used:

1.OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2. imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

2. pandas (all lowercase) is a popular Python-based data analysis toolkit which can be imported using import pandas as pd . It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array.

3.Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

4.matplotlib. pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

5.TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

6.KERAS is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation

Methods

1. NearMiss()

   Near Miss refers to a collection of undersampling methods that select examples based on the distance of majority class examples to minority class examples.

2. ImageDataGenerator()

   Keras ImageDataGenerator class provides a quick and easy way to augment your images. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change, and many more.

3. Adam

   Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data.

4. ReduceLROnPlateau

   Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

   - monitor: quantity to be monitored.
   - factor: factor by which the learning rate will be reduced. new_lr = lr * factor.
   - patience: number of epochs with no improvement after which learning rate will be reduced.
   - verbose: int. 0: quiet, 1: update messages.
   - mode: one of {'auto', 'min', 'max'}. In 'min' mode, the learning rate will be reduced when the quantity monitored has stopped decreasing; in 'max' mode it will be reduced when the quantity

monitored has stopped increasing; in 'auto' mode, the direction is automatically inferred from the name of the monitored quantity.

- min_delta: threshold for measuring the new optimum, to only focus on significant changes.
- cooldown: number of epochs to wait before resuming normal operation after lr has been reduced.
- min_lr: lower bound on the learning rate.

5. cv2.imshow()
   We can use the imshow() method of the cv2 library to display an image in a window. In order to use the cv2 library, we need to import the cv2 library using the import statements.

6. train_test_split
   is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.
7. LabelBinarizer
   At learning time, this simply consists in learning one regressor or binary classifier per class. In doing so, one needs to convert multi-class labels to binary labels (belong or does not belong to the class). LabelBinarizer makes this process easy with the transform method.

Performance Metrics:
The accuracy_score function computes the accuracy

The confusion_matrix function evaluates classification accuracy by computing the confusion matrix with each row corresponding to the true class.

The classification_report function builds a text report showing the main

classification metrics.
precision is the ability of the classifier not to label as positive a sample that is negative, and recall is the ability of the classifier to find all the positive samples.The F-measure can be interpreted as a weighted harmonic mean of the precision and recall.

Doing Prediction on External Image:

We have read an external image that is originally an image of the alphabet ‘k’ and made a copy of it that is to go through some processing to be fed to the model for the prediction that we will see in a while.
The img read is then converted from BGR representation (as OpenCV reads the image in BGR format) to RGB for displaying the image, & is resized to our required dimensions that we want to display the image in.
We do some processing on the copied image (img_copy).
We convert the image from BGR to grayscale and apply thresholding to it. We don’t need to apply a threshold we could use the grayscale to predict, but we do it to keep the image smooth without any sort of hazy gray colors in the image that could lead to wrong predictions.
The image is to be then resized using cv2.resize() function into the dimensions that the model takes as input, along with reshaping the image using np.reshape() so that it can be used as model input. Later we make a prediction using the processed image & use the np.argmax() function to get the index of the class with the highest predicted probability. Using this we get to know the exact character through the word_dict dictionary.
This predicted character is then displayed on the frame.

LIMITATION OF THE WORK

The model does work with small case alphabets and the alphabet size needs to be bigger.
It needs a GPU powered system to fit the model and to get the faster balancing of data.

RESULTS

From the classification report, the f1 score is 0.97 and Prediction on external images shows accurate results.

CONCLUSION

We have successfully developed Handwritten character recognition with Python, Tensorflow, and Machine Learning libraries.

The Experimental results showed that the machine has successfully recognized the Handwritten English characters with the highest recognition accuracy of 90.2%. This can be also further extended to identifying the handwritten characters of other languages too.

The handwritten recognition system will find applications in handwritten names recognition, document reading, conversion of any handwritten document into structural text form and postal address recognition.

References

1. https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format
2. https://link.springer.com/article/10.1007/s11036-019-01243-5
3. https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/
4. https://stackoverflow.com/questions/54589669/confusion-matrix-error-classification-metrics-cant-handle-a-mix-of-multilabel