PRO-C123

Digit Recognition-2

What is our GOAL for this MODULE?

The goal of this module is to explore the image processing techniques to detect the numbers from the images.

What did we ACHIEVE in the class TODAY?

- We wrote a prediction algorithm which would detect the numbers from the given images.

Which CONCEPTS/CODING BLOCKS did we cover today?

- Using data from ML library
- Logistic regression
- Using open CV

How did we DO the activities?

1. We imported all the necessary libraries.

```
#Importing all the important models and install them if not installed on your device
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from PIL import Image
import PIL.ImageOps
import os, ssl, time
```

2. We fetched the data from the open ml library.

```
#Fetching the data
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
print(pd.Series(y).value_counts())
classes = ['0', '1', '2','3', '4','5', '6', '7', '8', '9']
nclasses = len(classes)
```

3. We were likely to get an error. So we introduced the concept of SSL to avoid the error.

```
#Setting an HTTPS context to fetch data from OpenML
if (not os.environ.get('PYTHONHTTPSVERIFY', '') and
    getattr(ssl, '_create_unverified_context', None)):
    ssl._create_default_https_context = ssl._create_unverified_context

#Fetching the data
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
print(pd.Series(y).value_counts())
classes = ['0', '1', '2','3', '4','5', '6', '7', '8', '9']
nclasses = len(classes)
```

4. Then we printed the data.

```
1    7877
7    7293
3    7141
2    6990
9    6958
0    6903
6    6876
8    6825
4    6824
5    6313
dtype: int64
```

5. We split the data to train and test the prediction model.

```
#Splitting the data and scaling it
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=9, train_size=7500, test_size=2500)
#scaling the features
X_train_scaled = X_train/255.0
X_test_scaled = X_test/255.0
```

6. We fit the data into the model and checked the accuracy of the model.

```
#Fitting the training data into the model
clf = LogisticRegression(solver='saga', multi_class='multinomial').fit(X_train_scaled, y_train)

#Calculating the accuracy of the model
y_pred = clf.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("The accuracy is :- ",accuracy)
```

7. We used the camera and captured every frame of it.

```
#Starting the camera
cap = cv2.VideoCapture(0)

while(True):
  # Capture frame-by-frame
  try:
    ret, frame = cap.read()
```

8. We created a rectangle at the center of the video as the model will only detect the digit inside that rectangle. And also changed the color of the video to gray.

```
while(True):
  # Capture frame-by-frame
  try:
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #Drawing a box in the center of the video
    height, width = gray.shape
    upper_left = (int(width / 2 - 56), int(height / 2 - 56))
    bottom_right = (int(width / 2 + 56), int(height / 2 + 56))
    cv2.rectangle(gray, upper_left, bottom_right, (0, 255, 0), 2)

    #To only consider the area inside the box for detecting the digit
    #roi = Region Of Interest
    roi = gray[upper_left[1]:bottom_right[1], upper_left[0]:bottom_right[0]]
```

9. We converted the image we got into PIL format to use it again and resized it to 28.

```
#Converting cv2 image to pil format
im_pil = Image.fromarray(roi)

# convert to grayscale image - 'L' format means each pixel is
# represented by a single value from 0 to 255
image_bw = im_pil.convert('L')
image_bw_resized = image_bw.resize((28,28), Image.ANTIALIAS)
```

10. We then inverted the image to make it scalar and give it an value between 0 and 255.

```
#invert the image
image_bw_resized_inverted = PIL.ImageOps.invert(image_bw_resized)
pixel_filter = 20
#converting to scalar quantity
min_pixel = np.percentile(image_bw_resized_inverted, pixel_filter)
#using clip to limit the values between 0,255
image_bw_resized_inverted_scaled = np.clip(image_bw_resized_inverted-min_pixel, 0, 255)
max_pixel = np.max(image_bw_resized_inverted)
```

11. Then we converted this data into an array and passed it to the model for prediction.

```
#converting into an array
image_bw_resized_inverted_scaled = np.asarray(image_bw_resized_inverted_scaled)/max_pixel
#creating a test sample and making a prediction
test_sample = np.array(image_bw_resized_inverted_scaled).reshape(1,784)
test_pred = clf.predict(test_sample)
print("Predicted class is: ", test_pred)
```

12. We displayed the resulting frame and added key control to stop the model.

```
# Display the resulting frame
cv2.imshow('frame',gray)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

13. We code to close the active windows when the program is closed.

```
    except Exception as e:
      pass

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

What's NEXT?

In the next class, we will learn to create our own api using the flask.

EXTEND YOUR KNOWLEDGE:

Explore more data sets and the usage of fetch_openml
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_openml.html.