

# Guided Project Report

## Object Detection and Recognition

Name: Shruti Verma

Course: AI and ML

(Batch 4)

Duration: 10 months

Problem Statement: Implement a simple object detection network using any deep learning framework of your choice for the detection and recognition of COCO dataset.

### Prerequisites

What things you need to install the software and how to install them:

Python 3.8 or higher versions This setup requires that your machine has latest version of python. The following url <https://www.python.org/downloads/> can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: <https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/>. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url <https://www.anaconda.com/download/> You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.8 then run below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy . From Yolo download the Yolo config and weights files.

### Video Link

[https://drive.google.com/file/d/1EgHviGbNQL4SR1\\_yEysIcv4AYmgM0kXt/view?  
usp=sharing](https://drive.google.com/file/d/1EgHviGbNQL4SR1_yEysIcv4AYmgM0kXt/view?usp=sharing)

## Dataset used

Used COCO dataset - common objects in context. COCO is a large-scale object detection, segmentation, and captioning dataset with 330K images and 80 object categories.

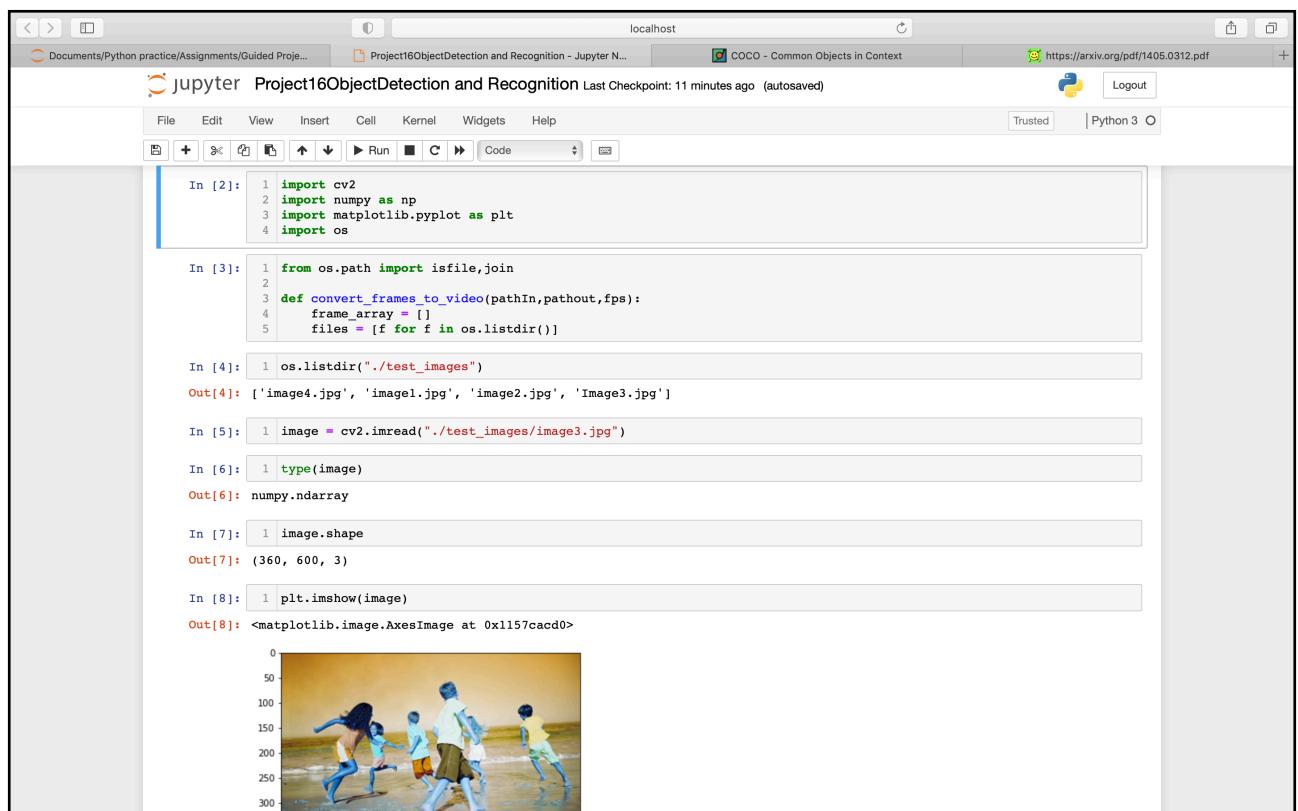
## Method used for detection

- Reading the image3, I have considered 4 images.
- Image preprocessing
- Applying Yolo model

Importing the libraries and capturing images:

Importing the necessary libraries and reading the image

Splitting the data into train and test and checking the labels and features



The screenshot shows a Jupyter Notebook interface running on localhost. The top bar includes tabs for 'localhost', 'COCO - Common Objects in Context', and a PDF link. The menu bar has options like File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. Below the menu is a toolbar with icons for file operations. The main area contains several code cells and their outputs:

- In [2]:

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import os
```
- In [3]:

```
1 from os.path import isfile,join
2
3 def convert_frames_to_video(pathIn,pathout,fps):
4     frame_array = []
5     files = [f for f in os.listdir()]
```
- In [4]:

```
1 os.listdir("./test_images")
```

Out[4]: ['image4.jpg', 'image1.jpg', 'image2.jpg', 'Image3.jpg']
- In [5]:

```
1 image = cv2.imread("./test_images/image3.jpg")
```
- In [6]:

```
1 type(image)
```

Out[6]: numpy.ndarray
- In [7]:

```
1 image.shape
```

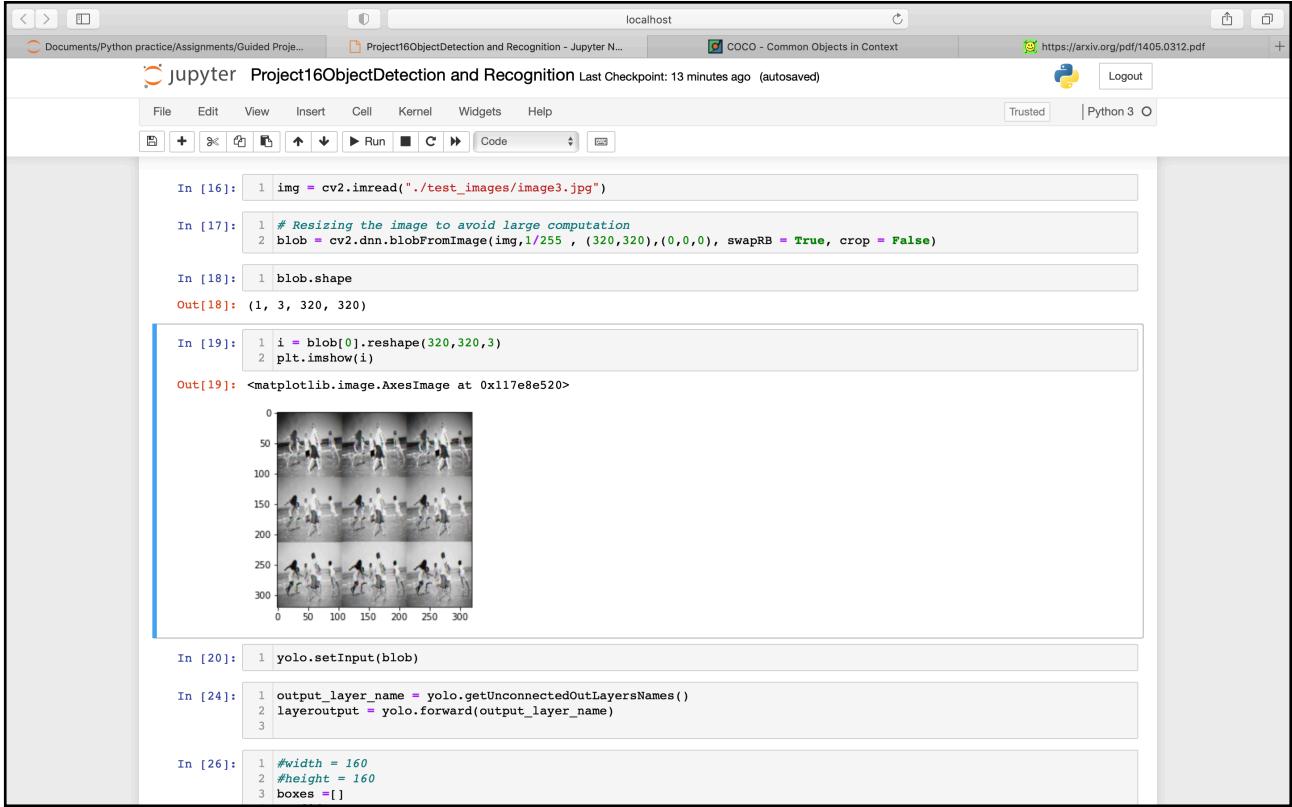
Out[7]: (360, 600, 3)
- In [8]:

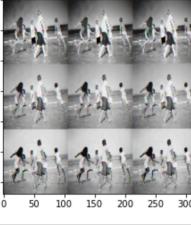
```
1 plt.imshow(image)
```

Out[8]: <matplotlib.image.AxesImage at 0x1157cacd0>

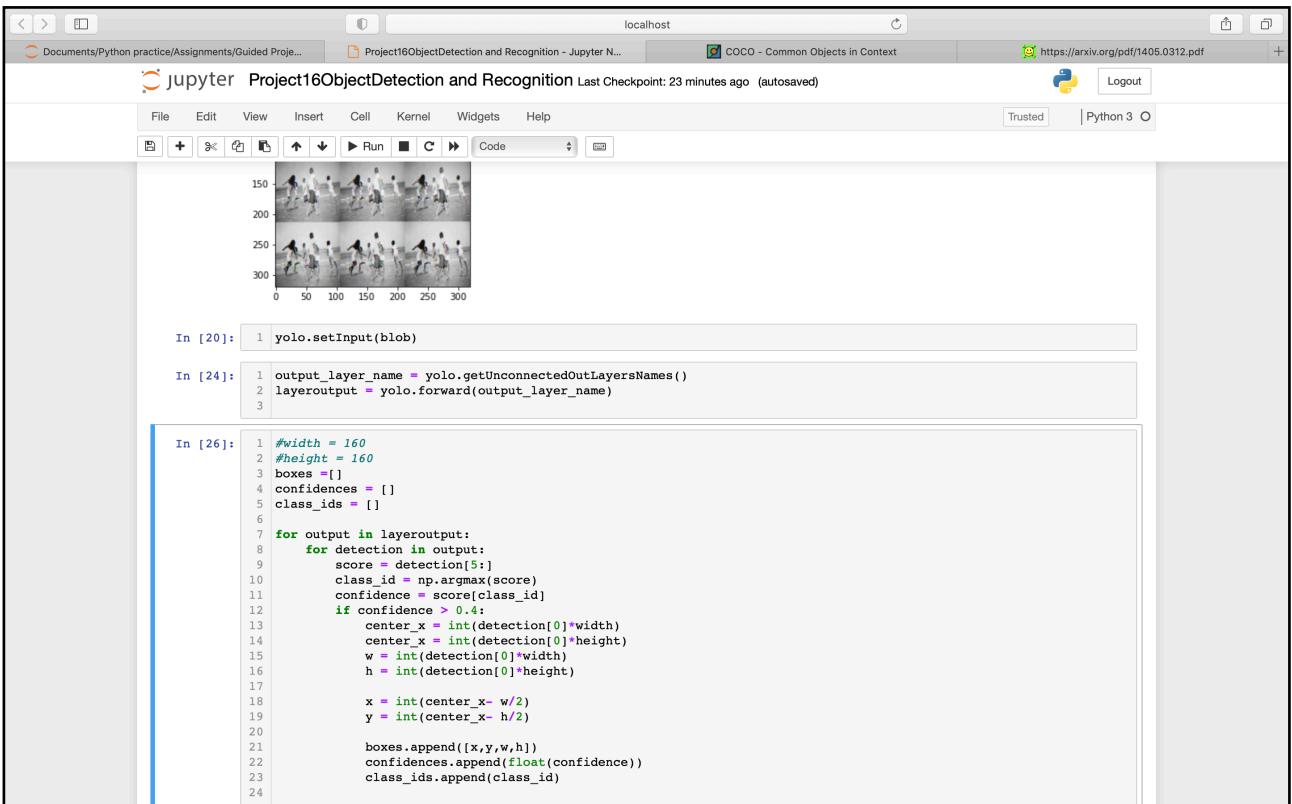


## Data processing and resizing the image to 320



In [16]: 1 img = cv2.imread("./test\_images/image3.jpg")  
In [17]: 1 # Resizing the image to avoid large computation  
2 blob = cv2.dnn.blobFromImage(img, 1/255, (320,320), (0,0,0), swapRB = True, crop = False)  
In [18]: 1 blob.shape  
Out[18]: (1, 3, 320, 320)  
In [19]: 1 i = blob[0].reshape(320,320,3)  
2 plt.imshow(i)  
Out[19]: <matplotlib.image.AxesImage at 0x117e8e520>  
  
In [20]: 1 yolo.setInput(blob)  
In [24]: 1 output\_layer\_name = yolo.getUnconnectedOutLayersNames()  
2 layeroutput = yolo.forward(output\_layer\_name)  
3  
In [26]: 1 #width = 160  
2 #height = 160  
3 boxes = []

## Using Yolo for input , setting the boxes, confidence and class\_ids



In [20]: 1 yolo.setInput(blob)  
In [24]: 1 output\_layer\_name = yolo.getUnconnectedOutLayersNames()  
2 layeroutput = yolo.forward(output\_layer\_name)  
3  
In [26]: 1 #width = 160  
2 #height = 160  
3 boxes = []  
4 confidences = []  
5 class\_ids = []  
6  
7 for output in layeroutput:  
8 for detection in output:  
9 score = detection[5]  
10 class\_id = np.argmax(score)  
11 confidence = score[class\_id]  
12 if confidence > 0.4:  
13 center\_x = int(detection[0]\*width)  
14 center\_x = int(detection[1]\*height)  
15 w = int(detection[2]\*width)  
16 h = int(detection[3]\*height)  
17  
18 x = int(center\_x - w/2)  
19 y = int(center\_x - h/2)  
20  
21 boxes.append([x,y,w,h])  
22 confidences.append(float(confidence))  
23 class\_ids.append(class\_id)

It will detect 6 objects in the given image

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** localhost, COCO - Common Objects in Context, https://arxiv.org/pdf/1405.0312.pdf
- Kernel:** Python 3
- Code Cell 27:** A snippet of Python code for processing detections. It filters detections based on confidence, extracts center coordinates and dimensions, and appends them to lists for boxes, confidences, and class IDs.
- Output 27:** Shows the result of len(boxes), which is 6.
- Code Cell 28:** cv2.dnn.NMSBoxes(bboxes, confidences, 0.5, 0.4)
- Code Cell 29:** Sets font to cv2.FONT\_HERSHEY\_PLAIN and generates random colors for each box.
- Code Cell 31:** Loops through the indexes, extracting x, y, w, h for each box. It then creates a label (class name + confidence) and draws a rectangle on the image at (x, y) with width w and height h. The confidence value is also printed next to the box.
- Code Cell 32:** Displays the image using plt.imshow(img).
- Output 32:** <matplotlib.image.AxesImage at 0x11815eeb0>

Setting the Font and colour of the boxes and then write the image

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** localhost, COCO - Common Objects in Context, https://arxiv.org/pdf/1405.0312.pdf
- Kernel:** Python 3
- Code Cell 27:** Shows the result of len(boxes), which is 6.
- Code Cell 28:** cv2.dnn.NMSBoxes(bboxes, confidences, 0.5, 0.4)
- Code Cell 29:** Sets font to cv2.FONT\_HERSHEY\_PLAIN and generates random colors for each box.
- Code Cell 31:** Loops through the indexes, extracting x, y, w, h for each box. It then creates a label (class name + confidence) and draws a rectangle on the image at (x, y) with width w and height h. The confidence value is also printed next to the box.
- Code Cell 32:** Displays the image using plt.imshow(img).
- Output 32:** <matplotlib.image.AxesImage at 0x11815eeb0>
- Code Cell 33:** cv2.imwrite("./img.jpg", img)
- Output 33:** True

The image shows a scene with several people. Three individuals are highlighted with blue bounding boxes and labeled with their respective confidence scores: "Person 0.95", "person 0.9", and "person 0.81".





localhost

Project15 Handwritten digit recognition - Jupyter Notebook

jupyter Project15 Handwritten digit recognition Last Checkpoint: 23 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [38]:

```
6 X = train_data.drop(columns = 'label')
7 ## Printing the size of data
8 print(train_data.shape)
9
(42000, 785)
```

In [39]:

```
1 ## Normalization
2
3 X = X/255.0
4 test_data = test_data/255.0
5
6 print("X:", X.shape)
7 print("test_data:", test_data.shape)
8
X: (42000, 784)
test_data: (28000, 784)
```

In [40]:

```
1 calling the features
2 m sklearn.preprocessing import scale
3 scaled = scale(X)
4
5 rain test split
6 rain, X_train, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.3, train_size = 0.2 ,random_state = 10)
```

In [41]:

```
1 # linear model
2
3 model_linear = SVC(kernel='linear')
4 model_linear.fit(X_train, y_train)
5
6 # predict
7 y_pred = model_linear.predict(X_test)
```

In [42]:

```
1 # confusion matrix and accuracy
2
3 from sklearn import metrics
4 from sklearn.metrics import confusion_matrix
5 # accuracy
6 print("accuracy:", metrics.accuracy_score(y_true=y_test, y_pred=y_pred), "\n")
7
8 # cm
```

localhost

Project15 Handwritten digit recognition - Jupyter Notebook

jupyter Project15 Handwritten digit recognition Last Checkpoint: 23 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [43]:

```
1 # model with optimal hyperparameters
2
3 # model
4 model = SVC(C=10, gamma=0.001, kernel="rbf")
5
6 model.fit(X_train, y_train)
7 y_pred = model.predict(X_test)
8
9 # metrics
10 print("accuracy", metrics.accuracy_score(y_test, y_pred), "\n")
11 print(metrics.confusion_matrix(y_test, y_pred), "\n")
```

accuracy 0.9438888888888888

```
[[1163  0   4   1   1   2   8   6   3   0]
 [ 0 1389  4   2   4   0   1   9   4   0]
 [ 1  4 1184  14  5   1   9   30  7   5]
 [ 0  3  15 1263  0   14  2   23  8   3]
 [ 1  2  20  3 1149  0   10  10  2   21]
 [ 2  8  3  30  4 1064  15  9  11  3]
 [ 8  1  3  0   3 13 1167  23  1   0]
 [ 4  9  10  8  12  0   0 1255  2  30]
 [ 5  18  17  23  8  20  5  13 1098  10]
 [ 5  3  2  27  21  1  1  51  3 1161]]
```

In [ ]: 1