Building & Analysis of Facial Video Dataset from Youtube

Load Dependecies

In [8]:
```
from pytube import YouTube
import cv2
import numpy as np
import os
import asyncio
```

Load Yolo weights file and read the classes from 'coco.names'

In [9]:
```
cat coco.names
```

```
person
bicycle
car
motorbike
aeroplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
backpack
umbrella
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
kite
baseball bat
baseball glove
skateboard
surfboard
tennis racket
bottle
wine glass
cup
fork
```

```
knife
spoon
bowl
banana
apple
sandwich
orange
broccoli
carrot
hot dog
pizza
donut
cake
chair
sofa
pottedplant
bed
diningtable
toilet
tvmonitor
laptop
mouse
remote
keyboard
cell phone
microwave
oven
toaster
sink
refrigerator
book
clock
vase
scissors
teddy bear
hair drier
toothbrush
```

In [10]:

```python
def load_yolo():
    """Load yolo weight file"""
    net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
    classes = []
    with open("coco.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]
#      classes = ['person'] # We only want to scan person in our video
    layers_names = net.getLayerNames()
    output_layers = [layers_names[i[0]-1] for i in net.getUnconnectedOutLayer
    colors = np.random.uniform(0, 255, size=(len(classes), 3))
    return net, classes, colors, output_layers
```

blobFromImage

1.Mean subtraction

2.Scaling

3.And optionally channel swapping

In [11]:

```python
def detect_objects(img, net, outputLayers):
    """
    Detect objects and normalize each pixel with a scaling factor of 0.00392
    Assuming RGB instead of BGR
    """
    blob = cv2.dnn.blobFromImage(img, scalefactor=0.00392, size=(320, 320), m
    net.setInput(blob)
    outputs = net.forward(outputLayers)
    return blob, outputs
```

In [12]:

```
def get_box_dimensions(outputs, height, width):
    """
    Creates a bounding box inside the image passed and return box dimensions
    """
    boxes = []
    confs = []
    class_ids = []
    for output in outputs:
        for detect in output:
            scores = detect[5:]
            class_id = np.argmax(scores)
            conf = scores[class_id]
            if conf > 0.3:
                center_x = int(detect[0] * width)
                center_y = int(detect[1] * height)
                w = int(detect[2] * width)
                h = int(detect[3] * height)
                x = int(center_x - w/2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confs.append(float(conf))
                class_ids.append(class_id)
    return boxes, confs, class_ids
```

In [13]:

```
def draw_labels(boxes, confs, colors, class_ids, classes, img, out):
    """Draw labels with the box in out cv2 object (videostream)"""
    indexes = cv2.dnn.NMSBoxes(boxes, confs, 0.5, 0.4)
    font = cv2.FONT_HERSHEY_PLAIN
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            color = colors[class_ids[i]]
            cv2.rectangle(img, (x,y), (x+w, y+h), color, 2)
            cv2.putText(img, label, (x, y - 5), font, 1, color, 1)
    out.write(img)
```

In [14]:

```
def start_video(video_path):
    """
    Check whether a person exists in the video clip passed
    video_path: param -> contains the video path of the file you downloaded
    """
    is_person_flag = False
    model, classes, colors, output_layers = load_yolo()
    cap = cv2.VideoCapture(video_path) # Capture video
    _, frame = cap.read()
    height, width, channels = frame.shape
#     print(frame.shape)
#     cap.release()
#     width = 640
#     height = 360

#     cap = cv2.VideoCapture(video_path)
#     codec = cv2.VideoWriter_fourcc(*'mp4v')
    codec = cv2.VideoWriter_fourcc(*'MJPG')
    out = cv2.VideoWriter('/home/sahil/ISM_PROJECT/object-detection-yolo-oper
    # save path of the video
    while cap.isOpened():
        # While video is open(until the end of the video)
        ret, frame = cap.read()
        if not ret or cv2.waitKey(1) & 0xFF == ord('q'):
            print(frame.all())
            break
        height, width, channels = frame.shape
        blob, outputs = detect_objects(frame, model, output_layers)
        boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
        if 0 in class_ids:
            # if a person exists in the video
            is_person_flag = True
        draw_labels(boxes, confs, colors, class_ids, classes, frame,out )

    cap.release()
    out.release()
    cv2.destroyAllWindows()

    if is_person_flag:
        print('This was a person speaking video')
    else:
        print('Removing this video')
```

Download youtube video from the path

In [15]:

```
def download_video(video_path):
    print(f'Downloading Youtube Video ')
    video_path =  YouTube(video_path).streams.filter(progressive=True, file_e
    print(f'Downloaded: {video_path}')
    return video_path
```

start_video('/home/sahil/ISM_PROJECT/object-detection-yolo-opencv/videos/pedestrians.mp4')

In [ ]:

```
def main():
    video_link = "https://www.youtube.com/watch?v=MRivVG0-GCg"
    video_path = download_video(video_link)
    print('Checking homo sapiens in the video...')
    start_video(video_path)
    print('Done')

if __name__ == "__main__":
    main()
```

```
Downloading Youtube Video
Downloaded: /home/sahil/ISM_PROJECT/object-detection-yolo-opencv/Great Speech
by Narendra Modi in Lok Sabha.mp4
Checking homo sapiens in the video...
```

In [ ]:

In [ ]: