Graduation

ASSIGNMENT
PRESENTATION SLIDE

Human detection and counting

By
Amadu Bah
University of Management and Technology
Dive Into Code Machine Learning
2021

Introduction

My is Amadu Bah
I Born in Freetown, Sierra Leone, West African.

Educational Background

I attended the Albert Academy Secondary School,
and Proceed for B.Sc In Computer Science
At University of Management And Technology

Introduction/Background

- In this machine Learning project with the used of python programming Language, I am going to build the Human Detection and Counting System through Webcam or video input. This is an intermediate level machine learning project on computer vision, which will help us to master the concepts and make an expert in the field of Data Science. Let's build an exciting project.

I will be Using following libraries for this Project

1. **OpenCV:** A strong library used for machine learning
2. **Imutils:** For Image Processing
3. **Numpy:** Used for Scientific Computing. Image is stored in a numpy array.
4. **Argparse:** Used to give input in command line.

To install the required Library run the following code in the terminal

```
pip install opencv-python
pip install imutils
pip install numpy
```

Import The following Libraries

Let us import the following Libraries we need for this this project

```
import cv2
import imutils
import numpy as np
import argparse
```

Create a model which will detect Humans:

cv2.HOGDescriptor_getDefaultPeopleDetector() calls the pre-trained model for Human detection of OpenCV and then we will feed our support vector machine with it.

I will be using HOGDescriptor with SVM already implemented in OpenCV. Below code will do this work:

```
HOGCV = cv2.HOGDescriptor()
HOGCV.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

Here, the actual magic will happen.
A video combines a sequence of images to form a moving picture. I call these images as Frame. So in general it will detect the person in the frame. And show it one after another that it looks like a video.
That is exactly what our Detect() method will do. It will take a frame to detect a person in it. Make a box around a person and show the frame..and return the frame with person bounded by a green box.
List containing Coordinates of bounding Box of person. Coordinates are in form of X, Y, W, H. Where x, y. are starting coordinates of box and w, h are width and height of box respectively.

```
def detect(frame):
    bounding_box_cordinates, weights =  HOGCV.detectMultiScale(frame, winStride = (4, 4), padding = (8, 8), scale = 1.03)

    person = 1
    for x,y,w,h in bounding_box_cordinates:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 2)
        cv2.putText(frame, f'person {person}', (x,y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1)
        person += 1

    cv2.putText(frame, 'Status : Detecting ', (40,40), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)
    cv2.putText(frame, f'Total Persons : {person-1}', (40,70), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)
    cv2.imshow('output', frame)

    return frame
```

cv2.VideoCapture(0) passing 0 in this function means we want to record from a webcam. video.read() read frame by frame. It returns a check which is True if this was able to read a frame otherwise False.

Now, For each Frame, we will call detect() method. Then we write the frame in our output file.

```python
def detectByPathVideo(path, writer):

    video = cv2.VideoCapture(path)
    check, frame = video.read()
    if check == False:
        print('Video Not Found. Please Enter a Valid Path (Full path of Video Should be Provided).')
        return

    print('Detecting people...')
    while video.isOpened():
        #check is True if reading was successful
        check, frame =  video.read()

        if check:
            frame = imutils.resize(frame , width=min(800,frame.shape[1]))
            frame = detect(frame)

            if writer is not None:
                writer.write(frame)

            key = cv2.waitKey(1)
            if key== ord('q'):
                break
        else:
            break
    video.release()
    cv2.destroyAllWindows()
```

DetectByCamera() method

```
def detectByCamera(writer):
    video = cv2.VideoCapture(0)
    print('Detecting people...')

    while True:
        check, frame = video.read()

        frame = detect(frame)
        if writer is not None:
            writer.write(frame)

        key = cv2.waitKey(1)
        if key == ord('q'):
                break

    video.release()
    cv2.destroyAllWindows()
```

HumanDetector() method

In this project, it can take images also. So this program will check if a path is given then search for the video in the given path and operate. Otherwise, it will open the webCam.

```
def humanDetector(args):
    # image_path = args["image"]
    video_path = args['video']
    if str(args["camera"]) == 'true' : camera = True
    else : camera = False

    writer = None
    if args['output'] is not None and image_path is None:
        writer = cv2.VideoWriter(args['output'],cv2.VideoWriter_fourcc(*'MJPG'), 10, (600,600))

    if camera:
        print('[INFO] Opening Web Cam.')
        detectByCamera(ouput_path,writer)
    elif video_path is not None:
        print('[INFO] Opening Video from path.')
        detectByPathVideo(video_path, writer)
```

Argparse() method

The function argparse() simply parses and returns as a dictionary the arguments passed through the terminal to our script. There will be Two arguments within the Parser:

Video: The path to the Video file inside your system

Camera: A variable that if set to 'true' will call the cameraDetect() method.

```
def argsParser():
    arg_parse = argparse.ArgumentParser()
    arg_parse.add_argument("-v", "--video", default=None, help="path to Video File ")
    arg_parse.add_argument("-i", "--image", default=None, help="path to Image File ")
    arg_parse.add_argument("-c", "--camera", default=False, help="Set true if you want to use the camera.")
    arg_parse.add_argument("-o", "--output", type=str, help="path to optional output video file")
    args = vars(arg_parse.parse_args())

    return args
```

Main function

I declared the model below as the main function

```
if __name__ == "__main__":
    HOGCV = cv2.HOGDescriptor()
    HOGCV.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

    args = argsParser()
    humanDetector(args)
```

Running the Human Detection Project

To run the human detection project, please run below mentioned commands as per requirements

1. To give video file as input:

   python main.py -v ‘Path_to_video’

2. To use the camera:

python main.py -c True

Whereas the main.py is the name of the project, and it will be executed in the python terminal

The End.