Detect people and track them throughout the view

Github link: jarvis1200/Count-Track_People (github.com)

Libraries used in code:

- Opencv
- Pytorch
- Scipy
- Numpy
- Imutils

Execution time of code:

- My laptop has I5 7gen processor, 8GB ram and Nvidia 940MX 2gb.
- In CPU, it takes 0.11 sec for each frame
- In GPU, it takes 0.07 sec for each frame

Code refactoring and optimization:

- Altered some line of codes without changing the behavior.
- My final source code optimized from 4 python files into single python file with better and fast result.

Nizamudeen A

Code Scalability:

- Performance, Accuracy and Speed is really matter.
- Multithreading in python is not robust at all. My laptop is not capable of loading heavy works.
- So, I choosed light weight model and for training I used Google colab.

Computation of Code:

- Time Complexity: In worst case it takes O(N^2)
- Space Complexity: O(N)

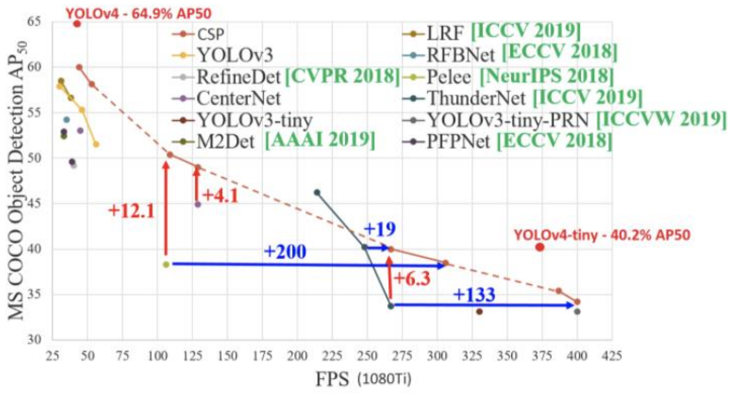
Data structure and Algorithm:

- I have used linear data structure like Array, HashMap

Documentation and Visualization:

- To detect human, I decided to use YoloV4 tiny model and crowd human dataset.
- For training in google colab, I used 416x416 yolo tiny model which is 22mb lightweight model.

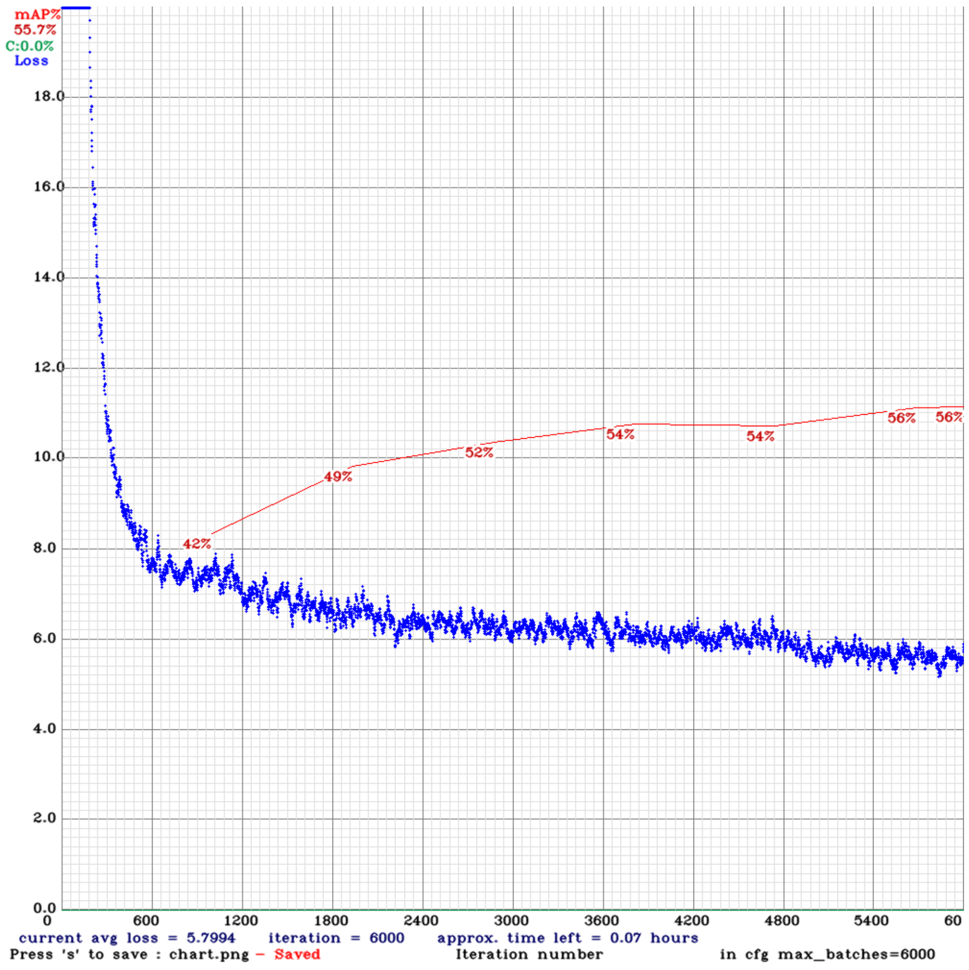
Nizamudeen A



- CrowdHuman is a benchmark dataset to better evaluate detectors in crowd scenarios. The CrowdHuman dataset is large, rich-annotated and contains high diversity. CrowdHuman contains 15000, 4370 and 5000 images for training, validation, and testing, respectively. There are a total of 470K human instances from train and validation subsets and 23 persons per image, with various kinds of occlusions in the dataset. Each human instance is annotated with a head bounding-box, human visible-region bounding-box and human full-body bounding-box. We hope our dataset will serve as a solid baseline and help promote future research in human detection tasks.
- For training the yolov4 model to detect 2 classes of object: "head" (0) and "person" (1), where the "person" class corresponds to "full body" (including occluded body portions) in the original "Crowd Human" annotations. Take a look at "data/crowdhuman-416x416.data", "data/crowdhuman.names", and "data/crowdhuman-416x416/" to gain a better understanding of the data files that have been generated/prepared for the training.

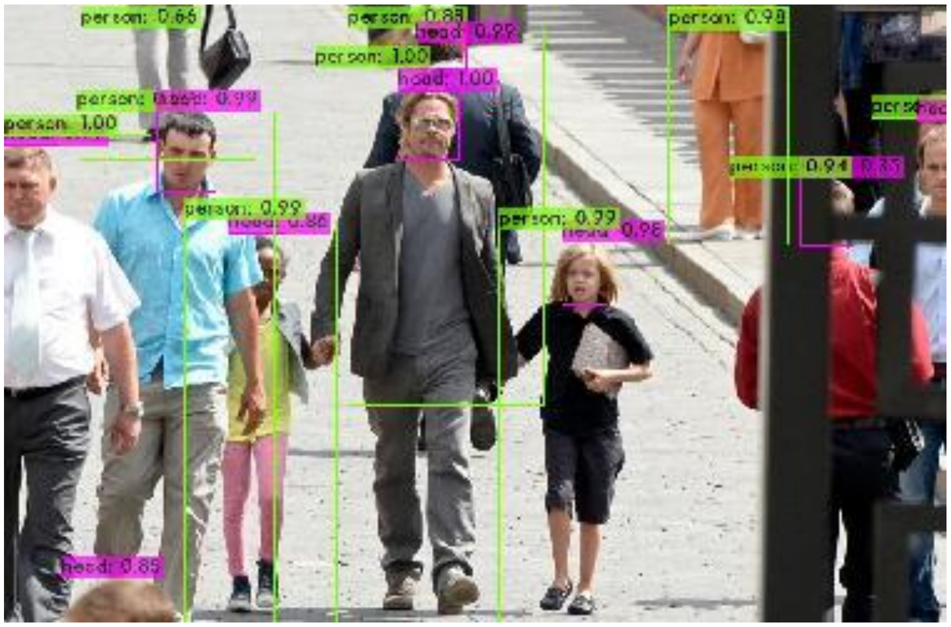
Nizamudeen A

- When the model is being trained, I monitor its progress on the loss/mAP chart.
- I used google colab's Tesla K20 GPU, for training it takes 7hrs.
- For the reference this is Training MAP chart.

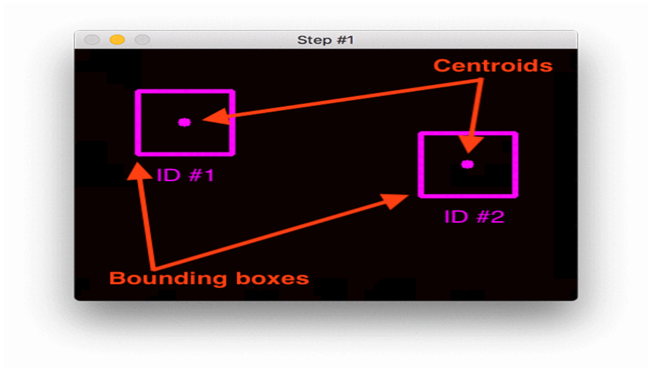


- I got 56% MAP and 5.7994 loss in 6000 epoch.
- Finally, the accuracy is pretty good.
- Look out the example image output.

Nizamudeen A

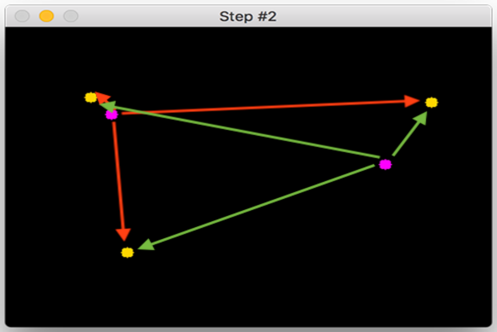


- In my local machine its pretty fast and good accuracy.
- Detecting object is done next, tracking and counting
- Assign a unique ID to that particular object
- Track the object as it moves around a video stream, *predicting* the new object location in the next frame based on various attributes of the frame (gradient, optical flow, etc.)
- Combining object detection and tracking using centroid tracking algorithm.

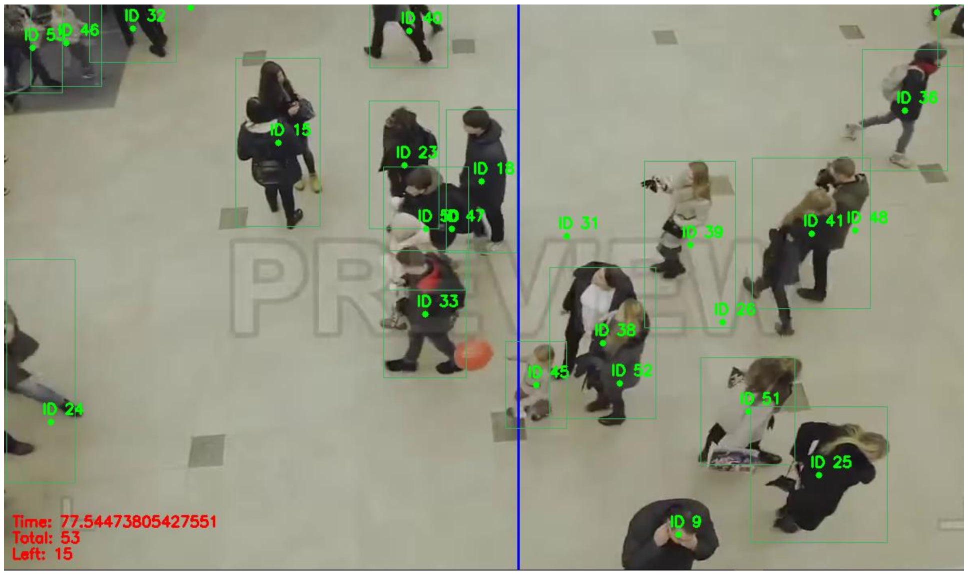


Nizamudeen A

- we accept a set of bounding boxes and compute their corresponding centroids
- we compute the Euclidean distance between any *new* centroids (yellow) and *existing* centroids (purple)



- The centroid tracking algorithm makes the assumption that pairs of centroids with minimum Euclidean distance between them *must be the same object ID.*
- centroid tracker has chosen to associate centroids that minimize their respective Euclidean distances.
- Assigned it a new object ID and, Storing the centroid of the bounding box coordinates for the new object is done.
- In order to track and count an object in a video stream, first find the object ID and then find It's previous centroids (so we can easily to compute the direction the object is moving) and each object is counted.
- Now let's draw vertical line to find the person who moving from left to right.
- We have trackable centroid list. In that we easily find the object which move from left to right of the line.
- The output has been shown in corner of video stream.
- Let's wrapped up with final screenshot of output video.



- YoloV4 tiny model give good result has you can see in this image.
- To implement this code in your machine, clone my github link in your machine.
- Install requirements libraries and run people_counter.py.
- Video stream is shown as above picture.