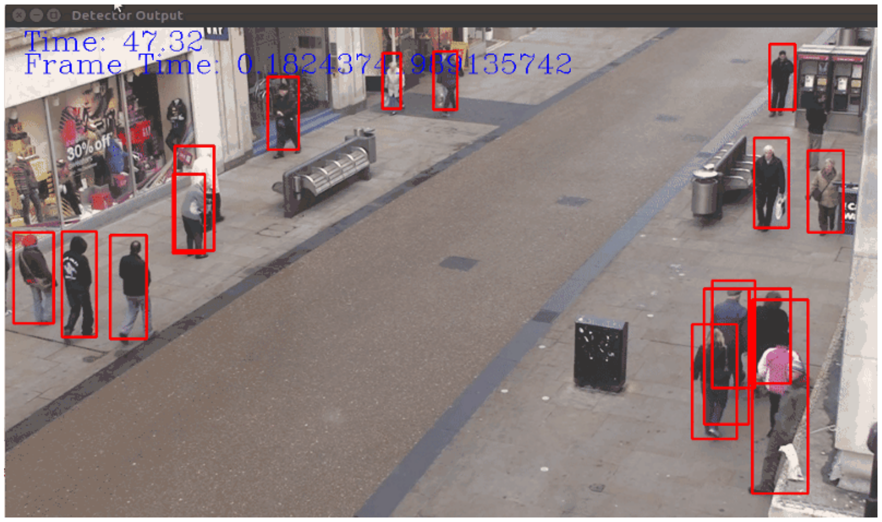
Madhawa Vidanapathirana Follow
Mar 28 · 8 min read

Real-time Human Detection in Computer Vision — Part 2

In the part 1 of this series, we discussed about the early approaches used for **Human Detection** such as **Viola Jones Object Detection Framework (Haar Cascades)** and **Histograms of Oriented Gradients for Human Detection (HOG).** We looked at some of the draw-backs of these earlier approaches such as **missed detections**, **false detections**, **duplicate detections** and **unreliable detection boundary.** In this part of the series, we will look at how modern approaches for Human Detection has addressed some of the issues of early approaches (up-to a considerable extent). We will also look at a **sample code** which would utilize some of the modern approaches for Human Detection.

Modern approaches for Human Detection



Human Detection using "Faster RCN Inception V2 COCO" model available with Tensorflow Object Detection API. Test Video from "Coarse Gaze Estimation in Visual Surveillance Project" by University of Oxford

Modern approaches for human detection we consider here are characterized by following special features.

They are "Deep Convolution Neural Networks"

Modern approaches for human detection are largely based on **Deep Convolution Neural Networks**. This trend started with **AlexNet** which won the **Imagenet Large Scale Visual Recognition Challenge (ILSVRC)** in year 2012 using a Deep Convolution Network (CNN) for Image Classification. Since then, CNNs were widely adapted for various computer vision problems such as **Image Classification** (identifying what type of an object an image contains), **Object Detection** (detecting different types of objects in a image)and **Object Localization** (determining locations of detected objects). “Human Detection” as we identified previously, is a special case of Object Detection and Object Localization.

They are "Multi-class Object Detectors"

Another key feature of modern CNN based Object Detection systems is that they are capable of detecting **multiple classes of objects.** Thus, modern state-of-the-art Human Detectors are not just Human Detectors, but accurate Object Detectors which can detect multiple types of objects including humans. With this context in mind, let me introduce to you “**Tensorflow Object Detection API**” and “**Tensorflow Detection Model Zoo”.**

Human Detection using Tensorflow Object Detection API

TensorFlow™ is an **open-source API** from **Google**, which is widely used for solving machine learning tasks that involve **Deep Neural Networks**. **Tensorflow Object Detection API** is an open-source library made based on Tensorflow for supporting training and evaluation of Object Detection models. Today we will take a look at “**Tensorflow Detection Model Zoo**”, which is a collection of **pre-trained models** compatible with **Tensorflow Object Detection API**.

At the time of this writing, **Tensorflow Detection Model Zoo** consists of 16 Object detection models pre-trained on **COCO Dataset**. Top 12 from this list of models provide “boxes” as output and they are compatible with the code linked to this article. These models are capable of detecting 80 types of objects including humans. Today, we will look at how these models can be used for Human Detection.

Setting up a Basic Human Detector

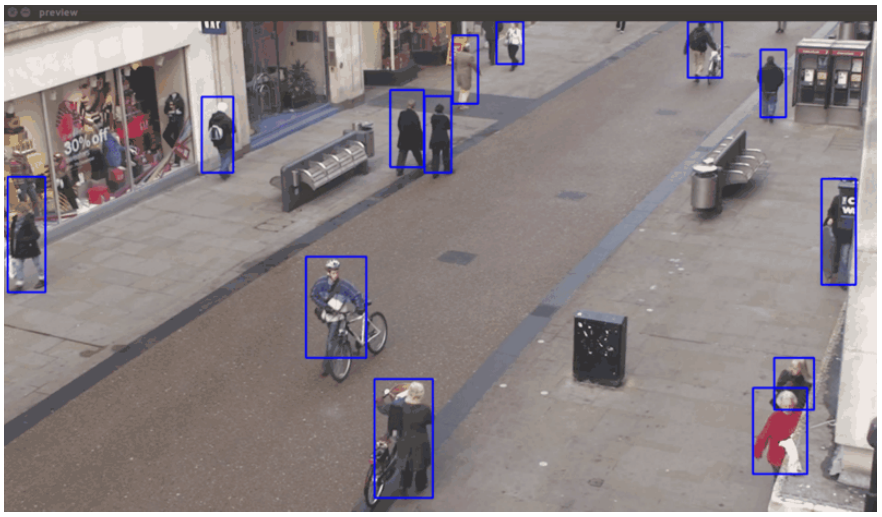
1. First and foremost, make sure **Open CV 3.0** (or above) and **Tensorflow 1.5** (or above) are installed. It is recommended to use the Tensorflow GPU version if you have a nVidia GPU. I have linked articles describing the installation process in the reference section.

2. Make a folder for this project.

3. Download **faster_rcnn_inception_v2_coco.tar.gz** from Tensorflow Detection Model Zoo. Extract the downloaded file to project directory.

4. Download **tensorflow-human-detection.py** from here and move it to project directory. Open the downloaded file and update the '/path/to/faster_rcnn_inception_v2_coco' to represent the extract location of the downloaded tar.gz file. Specify the '/path/to/input/video'. I have used the **TownCentre test video** downloaded from here.

5. Run the python file and observe the output on screen. (Press 'Q' to exit). You may adjust the threshold parameter to improve the results. Hopefully, you will see something like below.



Human Detection using faster_rcnn_inception_v2_coco Model (Threshold = 0.7)

You can try out the other 11 compatible models (listed under "COCO-trained models" and providing "boxes" as output) available in Tensorflow Detection Model Zoo by replacing the model file downloaded above. And of-course, the provided models are capable of detecting **all 80 types of objects** included in the **COCO dataset**. Only a small modification to the provided code is necessary to detect the other types of objects.

Test Bench

The mentioned results in this article are obtained using the provided code on a laptop with following specifications.

> Intel Core i7 7700 HQ (up-to 3.8 GHz), 16 GB Memory, nVidia Geforce GTX 1060 6GB VGA, Ubuntu 16.04, Open CV 3.4 and Tensorflow 1.5.

All the tests are carried out on "TownCentre" test video from "Coarse Gaze Estimation in Visual Surveillance Project" by University of Oxford.

Comparison among models from Tensorflow Detection Model Zoo

The model we considered above is *faster_rcnn_inception_v2_coco* model from Tensorflow Detection Model Zoo. Of-course, there are 11 other compatible models (listed under "COCO-trained models" and providing "boxes" as output) which we can consider for Human Detection. Google provides a **quantitative analysis** on these models in this table.

> *The COCO mAP figure provided in analysis table is a measure called "mean average precision". Higher mAP means more accurate results.*

I am more interested on the following 3 models from **Tensorflow Detection Model Zoo.**

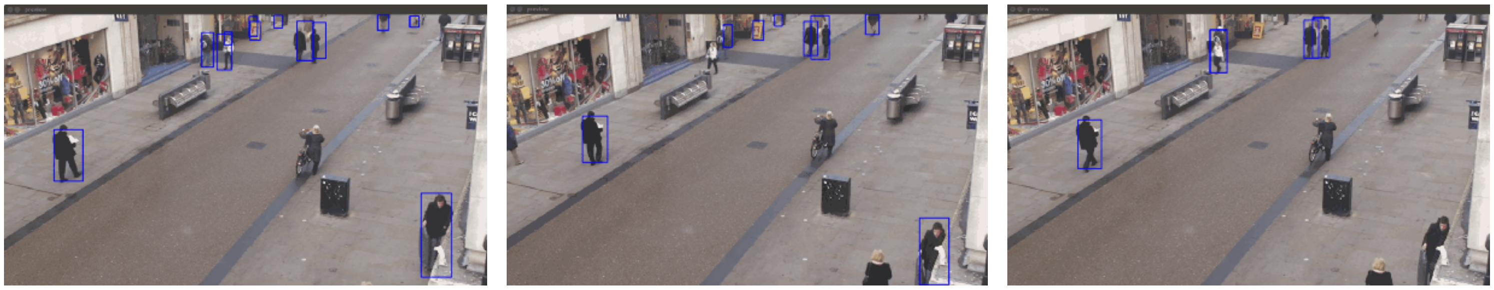
1. *ssd_mobilenet_v1_coco* (Can work in real-time on **Android phones**)
2. *faster_rcnn_inception_v2_coco* (Fair trade-off between accuracy and speed for GPU accelerated environments)
3. *faster_rcnn_nas* (Most accurate model provided)

Following topics cover a **qualitative analysis** on above 3 models.

SSD Mobilenet V1 COCO Model

This is the least accurate but the fastest from the list. It is capable of working in real-time on modern **Android Phones** as shown by **this**

android app which is based on this project.



Human Detection using ssd_mobilenet_v1_coco model (**Left**: Threshold = 0.2; **Center**: Threshold = 0.3; **Right**: Threshold = 0.5)

This model performs **reasonably well in detecting close-by-objects** occupying a large view space (such as a person standing in front of the camera). But it **performs poorly on our test video** since it contains a large number of persons each occupying a small space of the view. I had to reduce the detection threshold significantly to obtain reasonable detections for our test video.

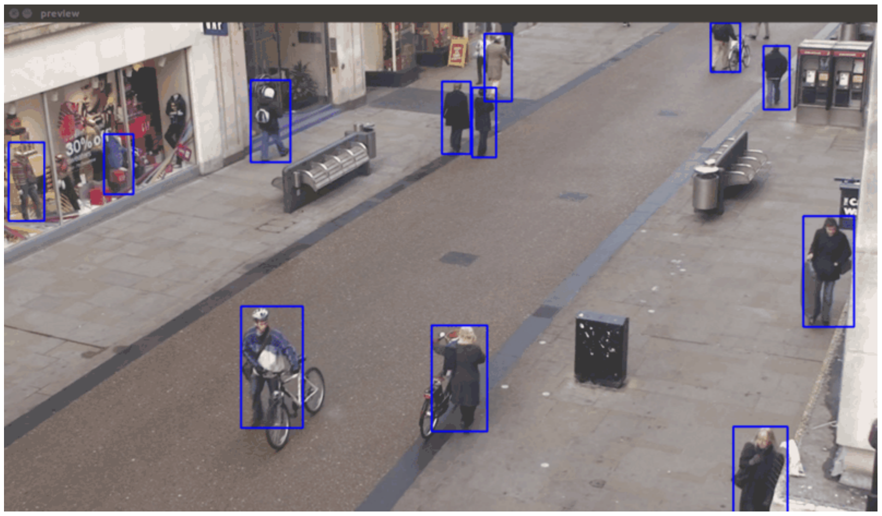
In my test bench with GPU Acceleration enabled, the **frame time** varied within the range 30–50 milliseconds. (Approximately 25 FPS). With GPU Acceleration disabled, the **frame time** varied within the range 60–80 milliseconds. (Approximately 15 FPS).

Faster RCN Inception V2 COCO Model

We already looked at this model when setting up the basic human detector. In my experience, this is the best model to choose if GPU acceleration is available and both frame-rate and accuracy matters. It gives **a fair trade-off between accuracy and speed** for GPU accelerated environments.

In my test bench with GPU Acceleration enabled, **frame time** varied within the range 200–300 milliseconds. (Approximately 4 FPS). With GPU Acceleration disabled, **frame time** varied within the range 1.5–1.6 seconds.

Faster RCN NAS COCO Model

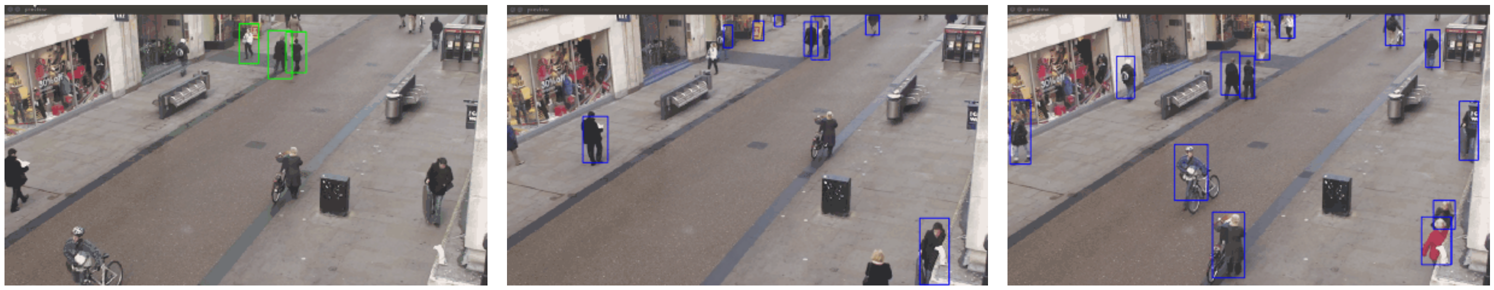


Human Detection using Faster RCN NAS COCO model (threshold = 0.5)

This model is the most accurate model from the set. Arguably one might say it is overly sensitive since it also detects the plastic human models in the left corner of the test video as humans. Hardly any **false detections**, **duplicate detections, flickering effects** or **missed detections** are noticed. However, this model is also the slowest. It consumes about 2.6 seconds to process a single frame with GPU Acceleration enabled.

Improvements from earlier approaches

In my observation, all the models that are more accurate than *Faster RCN Inception V2 COCO Model* (including it) performs significantly better than **HOG** and **Haar** approaches for Human Detection. These models provide a **tight consistent boundary** around a detected person. Additionally, these models are very less likely to result **false detections**. **Duplicate detections** may still occur in less accurate models at situations where multiple persons stay close-together. **Missed detections** and **flickers** may also occur occasionally if less accurate models are used. These problems are solved in more accurate models (such as *RCN NAS Model*) at the cost of computational time.


Haar cascade (left), SSD Mobilenet V1 (center) and Faster RCN Inception V2 (right)

The first two models in the list (ssd_mobilenet_v1_coco and ssd_inception_v2_coco) seem to perform marginally better than earlier approaches. These less accurate models are still susceptible to **missed detections**, **false detections**, **duplicate detections**, **flickers** and **inconsistent detection boundaries**.

Drawbacks compared to earlier approaches

In general, **deep neural network** based approaches provide **more accurate** results at cost of **more computations**. If GPU acceleration is not available, the lightest model discussed here *(ssd_mobilenet_v1_coco)* is capable of performing faster than earlier approaches discussed previously. But, the improved accuracy of this model is very marginal.

To obtain significant improvements in accuracy, it is necessary to consider ***Faster RCN Inception V2 COCO*** model or better. These models require GPU acceleration to provide comparable frame-rates to earlier approaches discussed previously.

Conclusion

Most of the issues present in early **human detection** approaches are significantly reduced in newer **deep learning** based approaches. These fixes are introduced at the cost of more computations. However, in the presence of **GPU acceleration**, modern **machine learning** libraries are capable of delivering these improved results with comparable frame-rates.

In the part 1, I promised to discuss about two recent trends **Human Pose Estimation** (identification of positions of joints of a person) and **Human Segmentation** (identification of polygonal boundary representing each person). I will cover these topics as separate articles in near future. So stay tuned for them.

Reference

1. All the tests are carried out on “TownCentre” test video from “Coarse Gaze Estimation in Visual Surveillance Project” by University of Oxford.
2. Official documentation from Tensorflow has a set-by-step guide on installation (Link).
3. Installation of OpenCV 3.4.