# People Counter Write Up

## Model Used

**Name:** ssd_mobilenet_v2_coco
**Link:**
http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_coco_2018_03_29.tar.gz

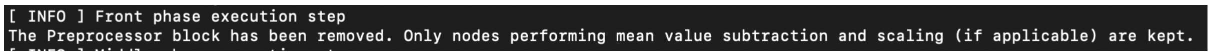
## 1. Explain the process behind converting any custom layers. Explain the potential reasons for handling custom layers in a trained model.

Custom layers are necessary to handle layers of a model that are unsupported by the OpenVINO™ Toolkit. If we don't handle unsupported layers using custom layers then we can't convert the models to OpenVINO IRs using the model optimiser. This would ultimately mean that we would be unable to use the model with the OpenVINO Toolkit. For example, when I optimised the ssd_mobilenet_v2_coco model using OpenVINOs model optimiser there were unsupported layers. I was able to overcome this by adding a CPU extension which is a form of custom layer. There are a number of different ways of adding custom layers depending on what framework the model comes from. Both TensorFlow and Caffe allow you to register the custom layers as extensions to the Model Optimizer. Another option for Caffe is to register the layers as Custom, then use Caffe to calculate the output shape of the layer. While Tensorflow's second option is to actually replace the unsupported subgraph with a different subgraph. Tensorflow also has one final option, which is to actually offload the computation of the subgraph back to TensorFlow during inference.

## 2. Run the pre-trained model without the use of the OpenVINO™ Toolkit. Compare the performance of the model with and without the use of the toolkit (size, speed, CPU overhead). What about differences in network needs and costs of using cloud services as opposed to at the edge?

The ssd_mobilenet_v2_coco_2018_03_29 model I used performed better using the OpenVino Toolkit than without it. The size of the model decreased, while the speed of the model increased when I converted it to an IR using the

OpenVino Toolkits Model Optimiser. This decrease in size and increase in speed is likely due to the model optimiser removing layers of the model that are only needed for model training, such as the Preprocessor block as can be seen from the console output below.



```
[ INFO ] Front phase execution step
The Preprocessor block has been removed. Only nodes performing mean value subtraction and scaling (if applicable) are kept.
```

After using the OpenVino Toolkit, the model I used was on the edge versus it being on a server when I didn’t use OpenVino. There were obviously no cloud services costs when running it on the edge while there would be a cost for using cloud services. It is also worth noting that when I ran the model on the edge, I didn’t have to deal with the latency from network communication, which also contributed to the model running faster.

## 3. Explain potential use cases of a people counter app, such as in retail applications. This is more than just listing the use cases - explain how they apply to the app, and how they might be useful.

The people counter app has a number of use cases. One of which that is very applicable in current times is to ensure social distancing is being obeyed within a given space. For example, in shops or public areas the app can be used to flag when there are too many people in that area at a single time. Likewise, in a restaurant, if you had a camera with a specific range of view you could identify if there are too many people sitting near each other in the given area. The app would know how many people should theoretically be allowed in its field of view and if this number has been exceeded it could potentially trigger some deterrent or notify the shop and restaurant owners.

Other use cases of the app would be to accurately identify peak hours in grocery stores. The app could be used in the cameras at the entrances and exits of grocery stores. It could then calculate the number of people coming in and out of the stores, thus it would know the number of people in a store at any given time. Using this data business owners can identify when they are going to need additional staff to handle the number of people in their store and conversely when they will need less staff. Potentially notifications could also be sent to employees when there is a large number of people in the store so that they could be aware that they may need to work on the tills instead of something else. These use cases would ultimately save store owners money from over staffing their stores and would ensure a consistent service in a store without delays.

## 4. Discuss lighting, model accuracy, and camera focal length/image size, and the effects these may have on an end user requirement.

Lighting, model accuracy, and image size all play an effect on the type of model an end user should choose to use. Depending on the end user requirement, the best fit model for a solution can vary greatly. For example, in this project I chose to use the ssd_mobilenet_v2_coco model because it had a good balance between being lightweight and producing a high-quality result. However, the results weren't perfect and for certain people that entered the frame (mainly people in dark clothing) the confidence threshold returned by inference greatly dropped. Whereas originally, I tried using the faster_rcnn_resnet101_coco model which is more accurate in identifying but inference took far too long running on my machine and greatly slowed down the output displayed to the user. It was therefore unsuitable for me to use this model. However, if I only needed to do inference on a couple of frames with no time limit and where accuracy was of the upmost importance, I would have used the more accurate faster_rcnn_resnet101_coco model instead. Similarly, models can be trained with different input sizes and different lighting and will therefore be more accurate and suitable under these conditions. So, depending on the user requirements the right choice of model varies.

## 5. If you were unable to find a suitable model and instead used an existing Intel® Pre-Trained Model, you should document the three non-IR models you tried first, how you converted them, and why they failed.

I was able to use the ssd_mobilenet_v2_coco model. I found this model was lightweight enough that it could stream and perform inference on the video at an acceptable rate while also returning results of a high enough quality from inference that I could accurately count the number of people that could be seen in the video at any given time. One thing I did notice however is that the model struggled to identify people wearing dark clothing. Due to this I needed to reduce the confidence threshold to 30% and above for it to identify everyone in the video successfully.

**Command to convert Model to IR:**

```
python /opt/intel/openvino/deployment_tools/model_optimizer/mo.py --
input_model
./ssd_mobilenet_v2_coco_2018_03_29/frozen_inference_graph.pb --
tensorflow_object_detection_api_pipeline_config
./ssd_mobilenet_v2_coco_2018_03_29/pipeline.config --
reverse_input_channels --tensorflow_use_custom_operations_config
/opt/intel/openvino/deployment_tools/model_optimizer/extensions/front/tf/
ssd_v2_support.json --output_dir ./model –steps
```

**Command used to run inference on video:**

```
python main.py -i ./resources/Pedestrian_Detect_2_1_1.mp4 -m
./model/frozen_inference_graph.xml -l
/opt/intel/openvino/deployment_tools/inference_engine/lib/intel64/libcpu_e
xtension_sse4.so -d CPU -pt 0.3 | ffmpeg -v warning -f rawvideo -pixel_format
bgr24 -video_size 768x432 -framerate 24 -i - http://0.0.0.0:3004/fac.ffm
```