Explaining the custom layers:

The optimizer of the openvino while optimizing the model into intermediate representation, it list the supported and unsupported layers, these unsupported layers are said to be custom layers.

Potential reasons for handling custom layers:

Networks training is typically done on high-end data centers, using popular training frameworks like Pytorch or TensorFlow*. Deploy (or inference), can also take place on the embedded, low-power platforms. Memory consumed by the model, time taken to load the model and inference time are the facts that makes challenging the deployment of IoT app at the edge. For this, we use different platforms to optimize the model for example openvino toolkit developed by intel. Model Optimizer tool enables automatic and seamless transition from the training environment to the deployment environment. The limitations of these target platforms make the deployment of the final solution very challenging, both with respect to the data types and API support.

The Model Optimizer converts the original TrnsorFlow formats to the Intermediate Representation (IR) file that describes the topology accompanied by a binary file with weights. These files are consumed by the Inference Engine and used for inference. If a certain function or a layer from the supported framework by openvino is not supported by the inference engine and declare as a custom operations or layers.

Custom Layers Workflow:

The Inference Engine has a notion of plugins (device-specific libraries to perform hardware-assisted inference acceleration). We need to select a target device, before creating any custom layer with the Inference Engine.

When creating new kernels in the Inference Engine, we must connect custom layers to these kernels as follows:

1.Register the custom layer for the Model Optimizer tool. This is required to generate correct Intermediate Representation (IR) file with the custom layers.

2.Implement the kernel in OpenCL™ for GPU acceleration or C++ for CPU that can be plugged into the Inference Engine.

3.Register the kernel in the Inference Engine, so that each time it meets the layer of the specific type in the IR, it inserts a call to the custom kernel into the internal graph of computations.

Comparing the model performance:

I am using ssdmobilenetV2 model from the tesorflow model API. The link is given bellow:

http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz

I used this command to optimize the model into IR format.

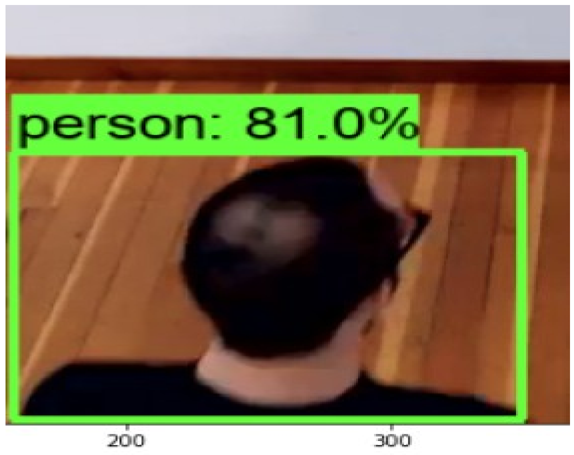
```
python3 /opt/intel/openvino/deployment_tools/model_optimizer/mo_tf.py –input_model
ssd_mobilenet_v2_coco_2018_03_29/frozen_inference_graph.pb        --tensorflow_use_custom_operations_config
/opt/intel/openvino/deployment_tools/model_optimizer/extensions/front/tf/ssd_v2_support.json --
tensorflow_object_detection_api_pipeline_config ssd_mobilenet_v2_coco_2018_03_29/pipeline.config --
reverse_input_channels -o ssd_mo_model
```

I used this command to run the optimized model on project workspace.

```
python3 main.py -m optimized_model/frozen_inference_graph.xml -i resources/Pedestrian_Detect_2_1_1.mp4 | ffmpeg -v
warning -f rawvideo -pixel_format bgr24 -video_size 768x432 -framerate 24 -i - http://0.0.0.0:3004/fac.ffm
```

Accuracy:

Model Accuracy comparison is based on the images in which person detected. By running the original model on 100 images from the video, note the accuracy, and take the average of all accuracies that is 88 percent. After conversion of model using openvino by running the optimized model on the same images. It has been observed the average accuracy is 83 percent.



Size:

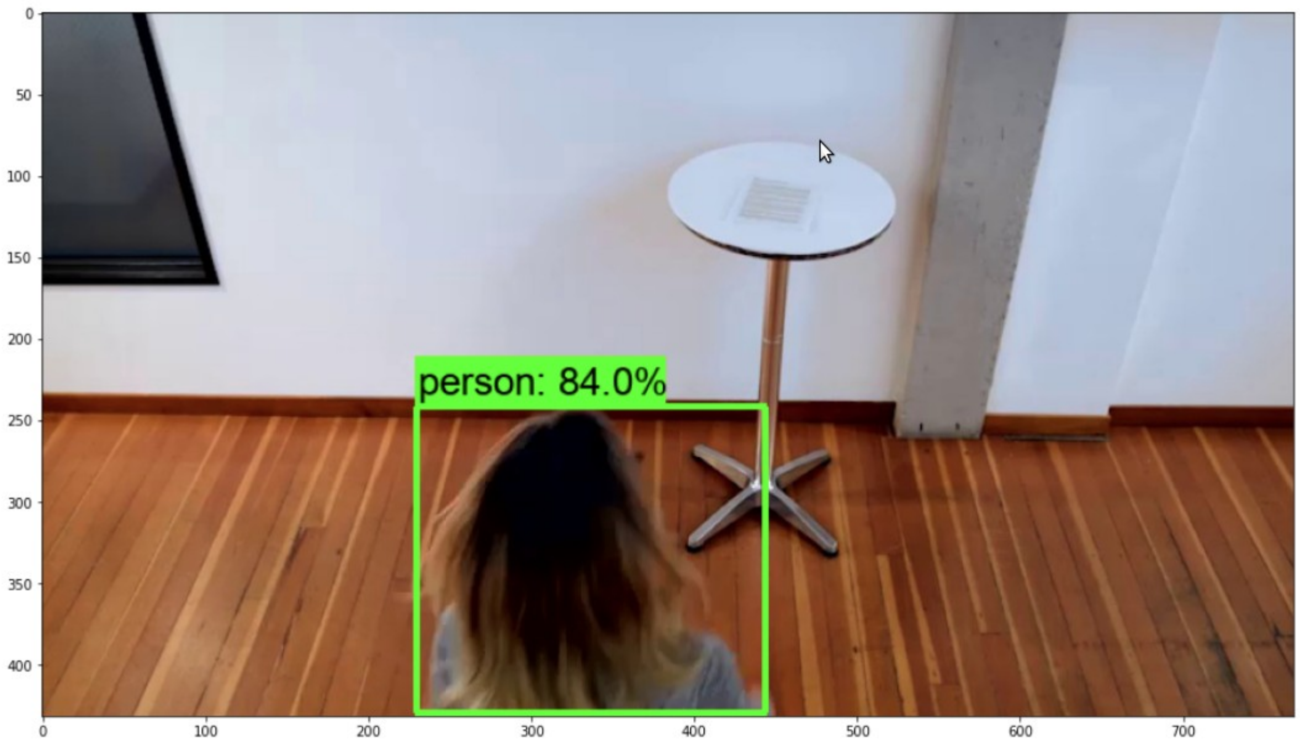
The size comparison table is given bellow:

Size before the model conversion:	69.7 MB
Size after the model conversion:	67.7 MB

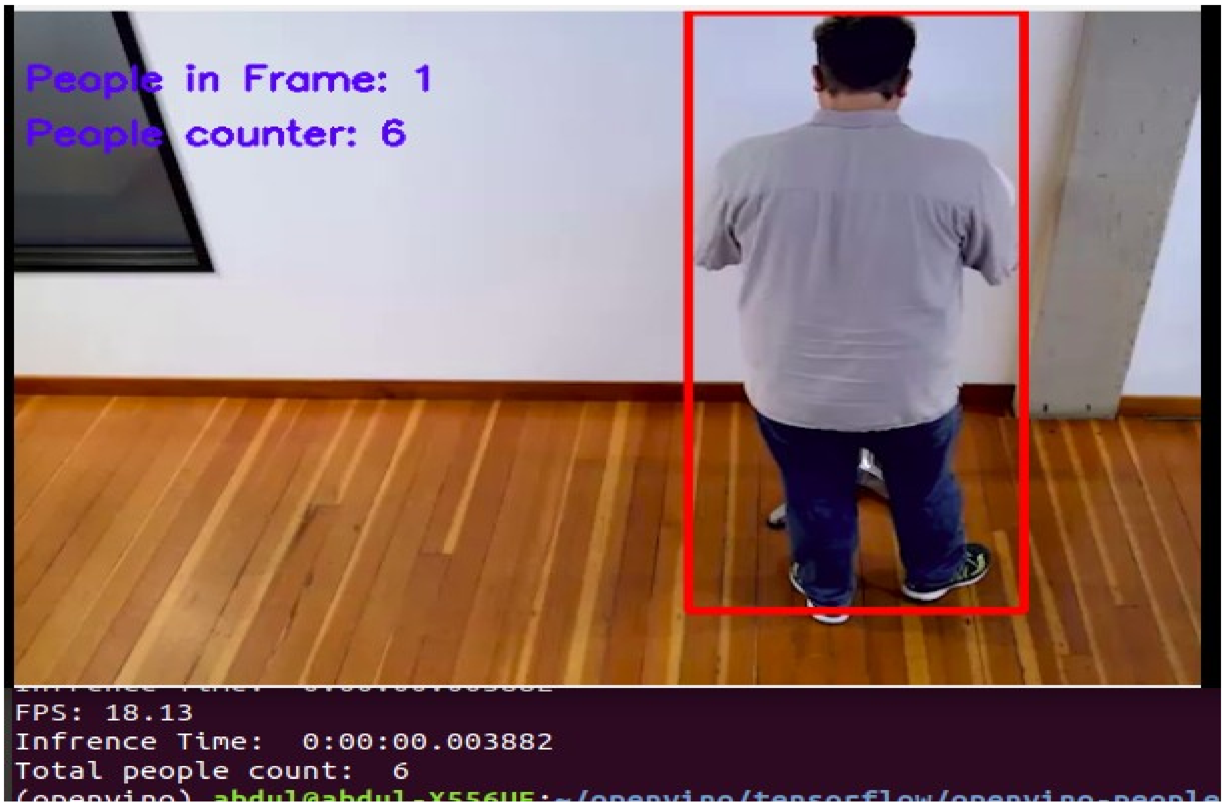
Inference time and FPS:

Inference time before the model conversion Inference time: 0:00:0.53158

```
Inference time: 0.5315890312194824 second per image
FPS:  1.881152434063524
```



Inference time after the model conversion Time: 0:00:00.003882



```
FPS: 18.13
Infrence Time:  0:00:00.003882
Total people count:  6
```

Applications of People Counter:

There are many applications of people counter app few of them are listed bellow.

1) **Retail**

   To understand traffic in the retail to organize merchandise in passageway, optimize store layout, understanding peak times and potentially even protect against theft is important. AI platform can get real time data from the camera in the store that can be analyzed by the people counter app.

2) **Covd-19**

   Government agencies could use people counter app in this covid-19 to check various things like how crowded are public places at a given time or how many people are using a particular street crossing in the lock-down.

3) **Safety**

   People counter application can be used for safety purposes. If police is searching for someone who is lost or looking for thief. It can be useful if a business would like to ensure an area is human free before starting a big machine.

4) **Education**

We can deploy people counter app in our educational institutes for attendance system. We can also use this app by modification to send the text to parents, on their childern reaching to the school or leaving the school.

5) **Voting system**

Voting system are usually prone to attacks and a group of people may manipulate the results against the people will. The people counter app at the election both will be helpful to control the theft in the voting system

Effects on End User Needs:

In everyday use Deep Learning is quickly bringing computer vision models to a spot where businesses can start integrating. The challenges at the end user side are who will build and maintain these models, how do he deploy the models, are they accurate enough, are now addressed in a new platform from that aims to connect model developers to end users.

Light effect: If people counter app is working on the camera located outdoor. There sunligh, cloudy or rainy condition, and more shining lights of vehicles on the road can effects the model performance at the edge app.

Accuracy means how well the models predict all of the labels correctly. They believe that higher accuracy means the better performance. Accuracy counts all of the true predicted values, but not specific for each label that exists. If we just focus on accuracy, we are not sure whether the model predicts well on some kind of labels that exist. If we already know what the metric number is, we can improve the model, by doing hyper-parameter tuning, feature selection, feature engineering, or even sampling the data to make a balanced-class data set.

Lens focal length tells us the angle of view, how much of the scene will be captured and the magnification, how large individual elements will be. The longer the focal length, the narrower the angle of view and the higher the magnification. The shorter the focal length, the wider the angle of view and the lower the magnification.

Image size: Increasing image resolution/size for CNN training often has a trade-off with the maximum possible batch size, yet optimal selection of image resolution has the potential for further increasing neural network performance for machine learning tasks.