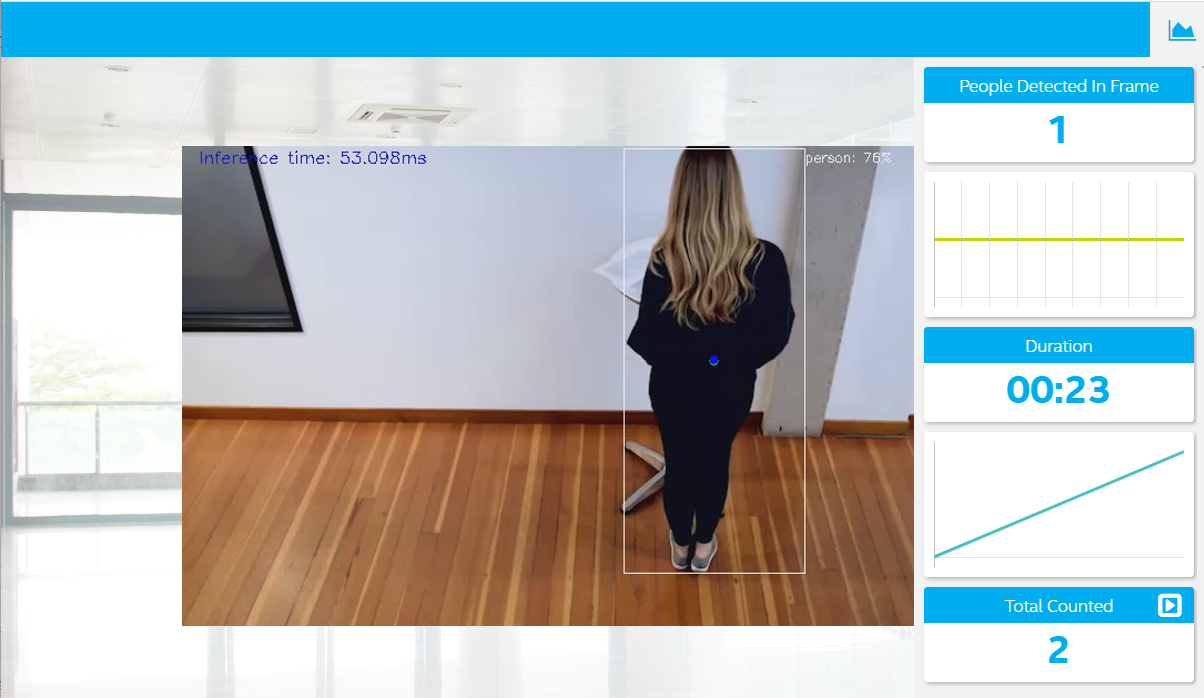
Project 1: Submission

Deploy a People Counter App at the Edge - Write-Up



*Due Date: May 26, 2020*

*By: Reginald Cobb*

**Table of Contents**

[Project 1: Deploy a People Counter App at the Edge 3](#_Toc41309962)

[Project Introduction 3](#_Toc41309963)

[Explaining Custom Layers 3](#_Toc41309964)

[What is a Custom Layer 3](#_Toc41309965)

[Identifying Custom Layers 3](#_Toc41309966)

[Custom Layer Processing 3](#_Toc41309967)

[Converting Custom Layers 3](#_Toc41309968)

[Reasons for Handling Custom Layers 4](#_Toc41309969)

[Comparing Model Performance 4](#_Toc41309970)

[Before and After Conversion to Intermediate Representation 4](#_Toc41309971)

[Size Comparison 5](#_Toc41309972)

[Inference Time Pre- and Post-Conversion 5](#_Toc41309973)

[Pre- and Post-Model Conversion Performance 5](#_Toc41309974)

[Size of Model Pre- and Post-Conversion 5](#_Toc41309975)

[Assess Model Use Cases 5](#_Toc41309976)

[Assess Effects on End User Needs 6](#_Toc41309977)

[Additional Model Research 6](#_Toc41309978)

# 

# Project 1: Deploy a People Counter App at the Edge

## Project Introduction

### Command used to Run Solution

python main.py -i resources/Pedestrian\_Detect\_2\_1\_1.mp4 -m example/MobileNetSSD\_deploy.xml -l /opt/intel/openvino/deployment\_tools/inference\_engine/lib/intel64/libcpu\_extension\_sse4.so -d CPU -pt 0.6 | ffmpeg -v warning -f rawvideo -pixel\_format bgr24 -video\_size 768x432 -framerate 24 -i - http://0.0.0.0:3004/fac.ffm

This project is a demonstration of a People Counter that detects people in the area of the video and collecting metrics for

* Number of people in each frame of the video at a given moment
* Average duration of the people in frames of the video (I changed this to the Duration.)
* Total Count of people appearing in the video (as defined as a person entering and exiting the video before another person enters the video.)

The application also evaluates the use and advantage of Intel software tools

## Explaining Custom Layers

### What is a Custom Layer

The Intel OpenVINO tools supports neural model layers for major frameworks. Each of the supported frameworks use “layers” to perform special purpose (relu, sigmoid, tanh, convolutional, etc.) calculations to sequentially build a neural network. Since each framework can supports a unique and different set of layers, any layers not natively supported by the OpenVINO toolset must be implemented as a custom layer.

#### Identifying Custom Layers

The OpenVINO toolkit supports a specific set of layers. If the model uses a layer that is not supported natively in the toolkit, e.g. unsupported, then that layer is said to be “unsupported” and the toolkit will report an error to the user.

#### Custom Layer Processing

Layers are processed by the Model Optimizer (MO) component of the toolkit. The MO uses topology information from the input model for each layer to build an optimized model that is used by other parts of the toolkit.

#### Converting Custom Layers

Any custom layer implemented for models need to enable two extensions in the MO and subsequent downstream processing: the Custom Layer Extractor and the Custom Layer Operation.

The [OpenVINO Toolkit Custom Layer Guide](https://docs.openvinotoolkit.org/latest/_docs_HOWTO_Custom_Layers_Guide.html) provides the overview diagram below, but in summary the MO extracts information from the input model, then the model is optimized based on that information. The optimized model serves as input for further downstream processing.

#### Reasons for Handling Custom Layers

A primary reasons for handling custom layers are to include specific processing in a model.

## Comparing Model Performance

As described below, I developed two Python scripts that implemented the pre- and post-conversion models.

### Initial Double Counting Problem

One of the initial problems was the fact that the model had intermittent frames where the person was not identified. While this did not affect the ability to correctly collect metrics, there was a problem with double counting people. To help avoid this problem, I implemented a centroid tracking algorithm on the current and previous position and used it to identify if a person was new if their current centroid was significantly different than the previous’ frame person centroid.

### Before and After Conversion to Intermediate Representation

To compare model performance before and after conversion to Intermediate Representations, I chose not to use the python script developed for Project 1 because it has MQTT, counters, time calculations, and other processing that while beneficial to the project’s objective, would hinder an apples to apples comparison. So I chose to simplify the activity and base my model comparison after the work on a webpage by dyan Mendez located at [this link](https://ebenezertechs.com/mobilenet-ssd-using-opencv-3-4-1-deep-learning-module-python/). .

I created two python scripts to evaluation performance and both are included in the zip file and github repository for Project 1 submission

* mobilnet\_ssd\_python.py
  + This python script contains the MobilNet-SSD object detection for the Caffe deep learning framework. The script uses the OpenCV Deep Neural Network (DNN) module. The model is converted includes a mean subtraction to normalize the inputs
* mobilnet\_ssd\_python\_OpenVINO.py
  + This python script uses the MobilNet-SDD object detection model for the Caffe framework but with the model converted to Intermediate Representation before inference is performed.

### Size Comparison

|  |  |
| --- | --- |
| **Model Size Comparison** | |
|  | ***MobileNetSDD*** |
| **Pre-Conversion** | 22.6MB |
| **Post-Conversion** | 22.6MB |

### Inference Time Pre- and Post-Conversion

|  |  |
| --- | --- |
| **Inference Times** | |
|  | *MobileNetSDD* |
| **Pre-Conversion** | 41.7 ms |
| **Post-Conversion** | 41.1 ms |

## Size of Model Pre- and Post-Conversion

|  |  |  |  |
| --- | --- | --- | --- |
| **Model Size Comparison - Caffe Implementaiton** | | | |
|  | ***Google MobileNet v1*** | ***Google MobileNet v2*** | ***MobileNetSDD*** |
| **Pre-Conversion** | 16.6MB | 13.8MB | 22.6MB |
| **Post-Conversion** | 16.4MB | 13.6MB | 22.6MB |

## Assess Model Use Cases

Some of the potential use cases of the people counter app are...

Each of these use cases would be useful because...

## Assess Effects on End User Needs

Lighting, model accuracy, and camera focal length/image size have different effects on a

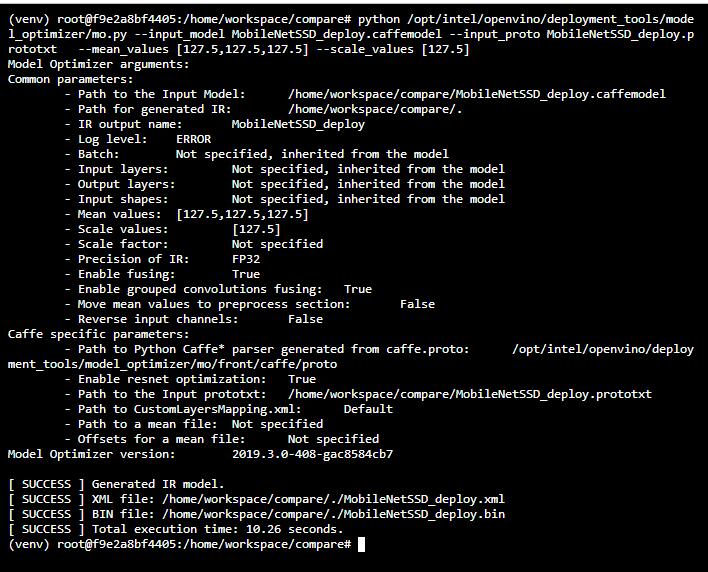
deployed edge model. The potential effects of each of can cause the model to infer the wrong shape for models, the classify wrong edges and be thrown off by contrast or gain or level pixels.

## Additional Model Research

I used the MobileNetSDD model for this project. I converted the model (and based on research adjusted for mean values) to an Intermediate Representation with the following arguments…

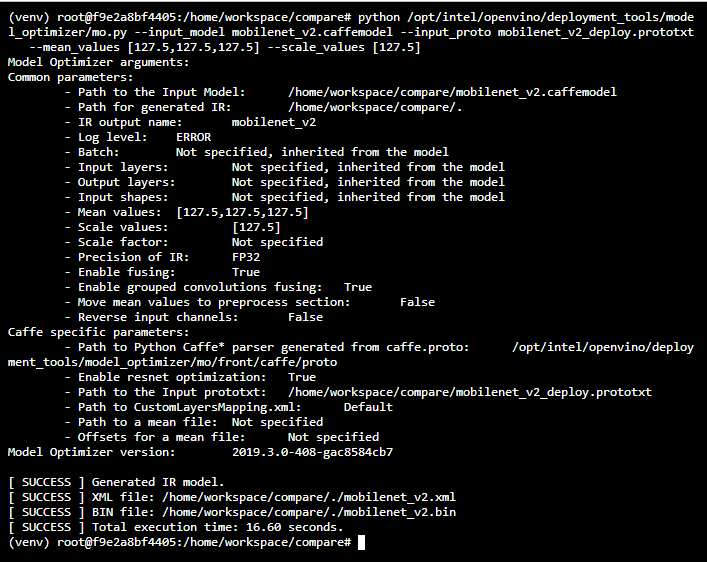
MobileNetSDD

python /opt/intel/openvino/deployment\_tools/model\_optimizer/mo.py --input\_model MobileNetSSD\_deploy.caffemodel --input\_proto MobileNetSSD\_deploy.prototxt --mean\_values [127.5,127.5,127.5] --scale\_values [127.5]



Mobilenet v2

python /opt/intel/openvino/deployment\_tools/model\_optimizer/mo.py --input\_model mobilenet\_v2.caffemodel --input\_proto mobilenet\_v2\_deploy.prototxt   --mean\_values [127.5,127.5,127.5] --scale\_values [127.5]



Mobilenet v1

python /opt/intel/openvino/deployment\_tools/model\_optimizer/mo.py --input\_model mobilenet.caffemodel --input\_proto mobilenet\_deploy.prototxt   --mean\_values [127.5,127.5,127.5] --scale\_values [127.5]

