

## # Project Write-Up

You can use this document as a template for providing your project write-up. However, if you have a different format you prefer, feel free to use it as long as you answer all required questions.

## ## Running the project

For running the video file

```
python main.py -i resources/Pedestrian_Detect_2_1_1.mp4 -m tensorflow-yolo-v3/frozen_darknet_yolov3_model.xml -l /opt/intel/openvino/deployment_tools/inference_engine/lib/intel64/libcpu_extension_sse4.so -d CPU -pt 0.4 | ffmpeg -v warning -f rawvideo -pixel_format bgr24 -video_size 768x432 -framerate 24 -i - http://0.0.0.0:3004/fac.ffmpeg
```

for running a webcam

```
python main.py -i CAM -m tensorflow-yolo-v3/frozen_darknet_yolov3_model.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 640x480 -framerate 24 -i - http://0.0.0.0:3004/fac.ffmpeg
```

## ## Explaining Custom Layers

One of the reasons of the success of deep learning is the wide range of layers that can be used and combined to create a model. But this is because developers are not constrained to the existent layers, but can create their own ones so they can solve a specific problem or implement a specific operation you can need in your model, or creating a customized version of an existing layer.

To create a custom layer for OpenVino, you must add extensions to the Model Optimizer and the Inference Engine.

For this, the first step is to use the Model Extension Generator tool

The MEG is going to create templates for Model Optimizer extractor extension, Model Optimizer operations extension, Inference Engine CPU extension and Inference Engine GPU extension.

Once customized the templates, next step is to generate the IR files with the Model Optimizer.

Finally, before using the custom layer in your model with the Inference Engine, you must: first, edit the CPU extension template files, second, compile the CPU extension and finally, execute the model with the custom layer.

## ## Comparing Model Performance

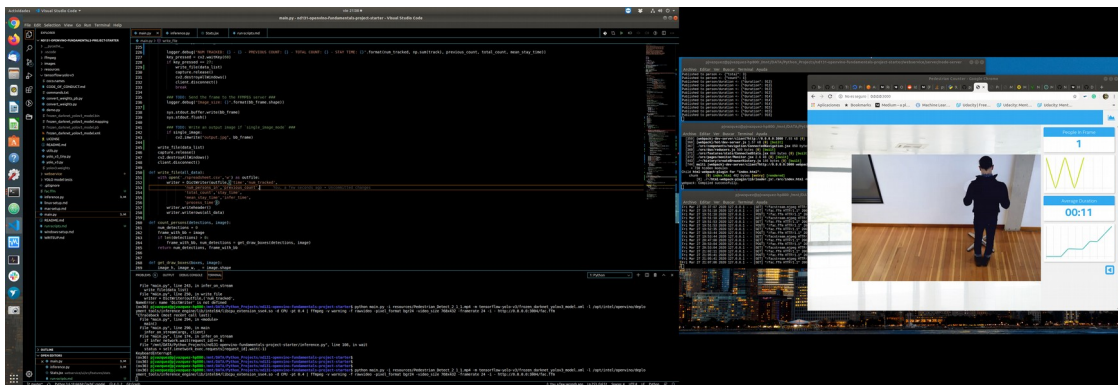
My method(s) to compare models before and after conversion to Intermediate Representations were:

Comparing processing speed in the same environment is useful to see if a model performs faster after transformation in IR. In my case, I'm using YOLOv3 model for a people detection and tracking task on images taken in a store and I run it on a GPU Nvidia GTX 1080

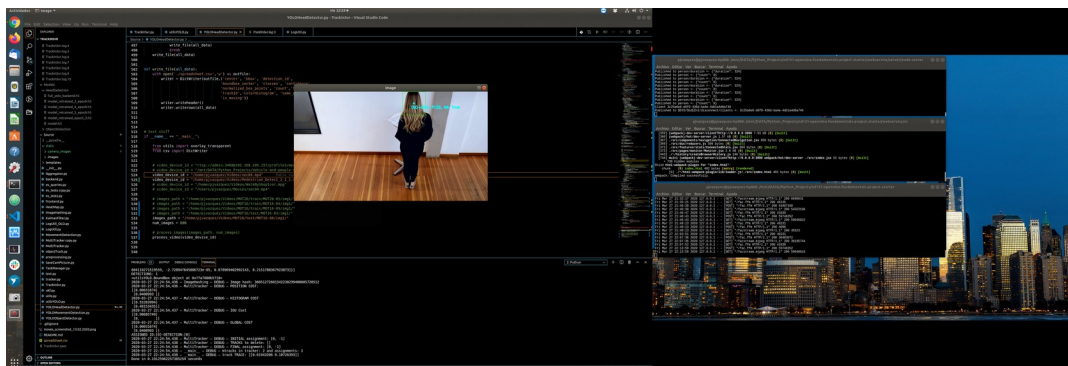
The difference obtained is a mean process time of the original YOLO v3 model in a Nvidia GTX 1080 of 0.20 seconds while the OpenVino model with the IR representation obtained is running on the i7 CPU of around 0.19 seconds

This means that OpenVino is faster.

Image with a failed detection



The difference between model accuracy pre- and post-conversion was quite big, while YOLO v3 model correctly detected the 6 persons with an accuracy of 95% and failing only on a small number of frames, YOLO IR lost accuracy and detected 8 persons, failing with two subjects on most of the time he was present



The size of the model pre- and post-conversion was similar, the YOLO v3 model pb file is about 248,2 MB while the IR bin file is about 247,7

## ## Assess Model Use Cases

Some of the potential use cases of the people counter app are of interest for retail commerce to know how and when customers get into the store or are in some point of

interest. It's also useful for security control, measuring how people performs in restricted or controlled spaces.

Each of these use cases would be useful because allows to improve marketing and control strategies both in the retail store or in the security control.

## ## Assess Effects on End User Needs

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

Lighting, focal length and image size are relevant to system behavior, a bad lighting can decrease model performance by diffusing image info, image size is relevant although YOLO models are quite size independent and the same happens with focal length.

Camera vision angle is also relevant for this kind of tasks and for performance of the system. Depending on the dataset used, (COCO in this model) some kind of angles can decrease model accuracy and also increase number of occlusions with the problems this generates in detection.

Model accuracy is relevant due to the amount of false positives or negatives it can generate degrading system performance.

## ## Model Research

[This heading is only required if a suitable model was not found after trying out at least three different models. However, you may also use this heading to detail how you converted a successful model.]

In investigating potential people counter models, I tried each of the following three models:

- Model 1: [Name]
- [Model Source]
- I converted the model to an Intermediate Representation with the following arguments...
- The model was insufficient for the app because...
- I tried to improve the model for the app by...
- Model 2: [Name]
- [Model Source]
- I converted the model to an Intermediate Representation with the following arguments...
- The model was insufficient for the app because...
- I tried to improve the model for the app by...
- Model 3: [Name]
- [Model Source]
- I converted the model to an Intermediate Representation with the following arguments...
- The model was insufficient for the app because...
- I tried to improve the model for the app by...