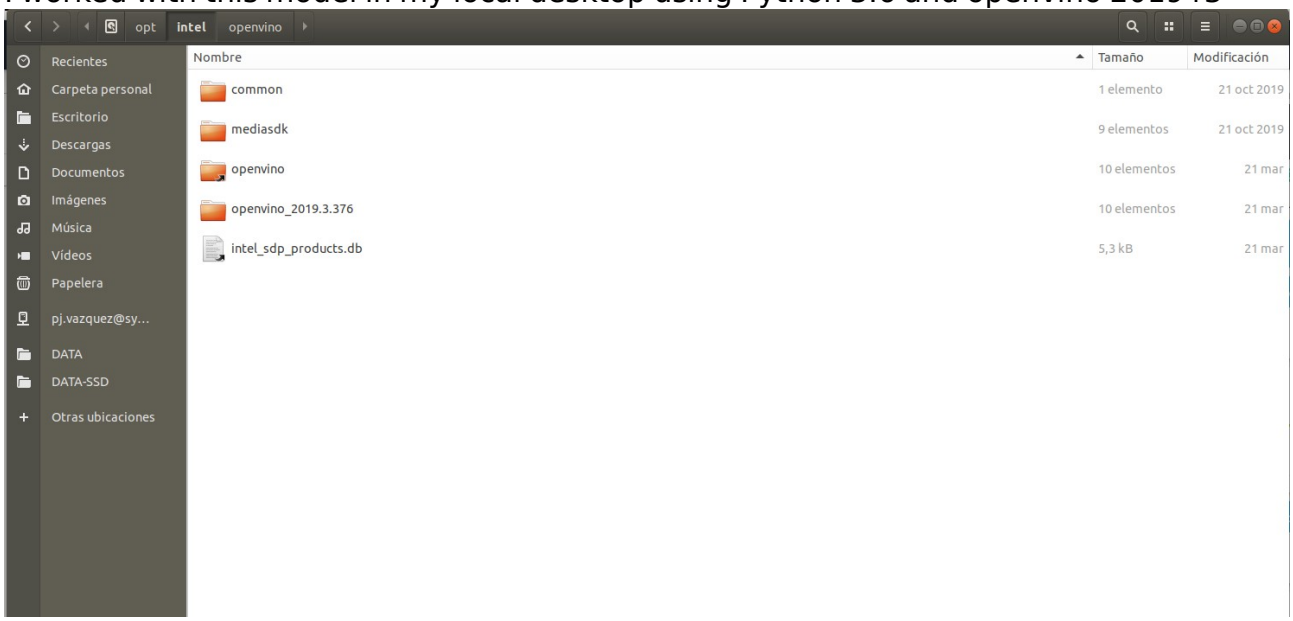
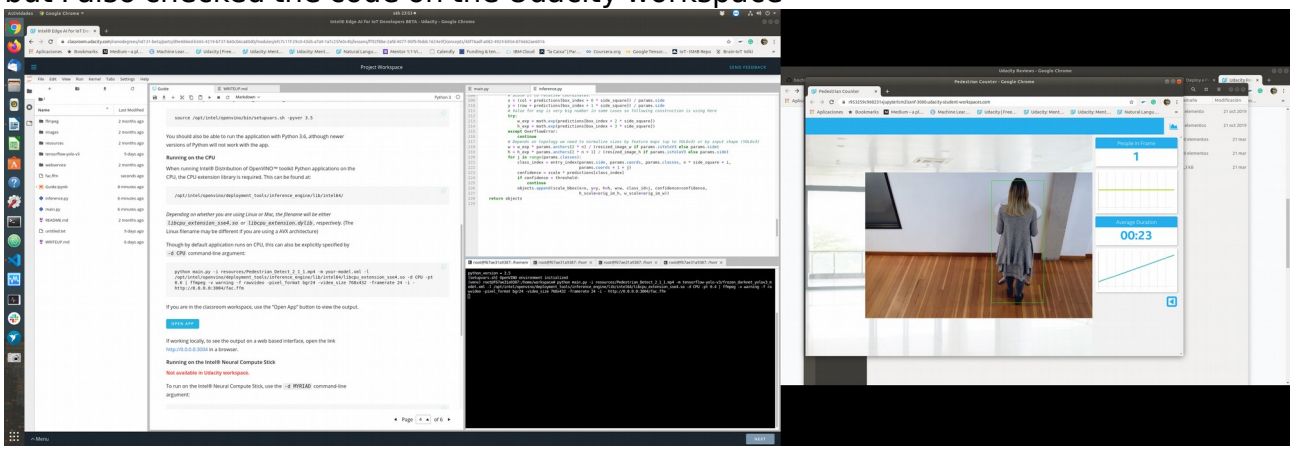
Project Write-Up

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.

I worked with this model in my local desktop using Python 3.6 and openvino 2019 r3



but I also checked the code on the Udacity workspace



About statistics

The model looks quite slow in the Udacity environment, in my computer is really fast running inly on the CPU. Probably it’s not the best model for this task, but I tried to use a complex model like a YOLO to see performance. I’ll try it on a raspberry with a movidius neural compute stick.

About the statistics, my experience on react web pages is 0, but the total count is showing the length of the list of elements received, as you say in the review, and the total number my system computes is 7, it’s because for some moments, the person on the video is not detected. I accumulate the 30 last detection values and use really low

threshold of 0.1 for **sum(number of last X detections)/X,** with low threshold and a history of last 30 detection, my system computes 7 as number of detected people.

I think that a couple of bugs on the UI should be fixed, but I do not feel confident with js code to do it.

Explaining Custom Layers

One of the reasons of the success of deep learning is the wide range of layers that can be used an 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 an specific problem or implement an 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.

For converting the darknet YOLO v3 model from a tensorflow implementation to the needed IR model the steps I followed instructions from https://docs.openvinotoolkit.org/2019_R3.1/_docs_MO_DG_prepare_model_convert_model_tf_specific_Convert_YOLO_From_Tensorflow.html

1- Download model files

To download model files from a linux system:
Get into tensorflow-yolo-v3 directory:

cd tensorflow-yolo-v3

Download the weights file:

wget https://pjreddie.com/media/files/yolov3.weights

2- Convert .weights to a .pb file

python convert_weights_pb.py --class_names coco.names --data_format NHWC --weights_file yolov3.weights

3- Convert the pb to a .bin and xml files using model optimizer

python /opt/intel/openvino/deployment_tools/model_optimizer/mo_tf.py --input_model frozen_darknet_yolov3_model.pb --tensorflow_use_custom_operations_config /opt/intel/openvino/deployment_tools/model_optimizer/extensions/front/tf/yolo_v3.json --batch 1

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.ffm

for running a webcam

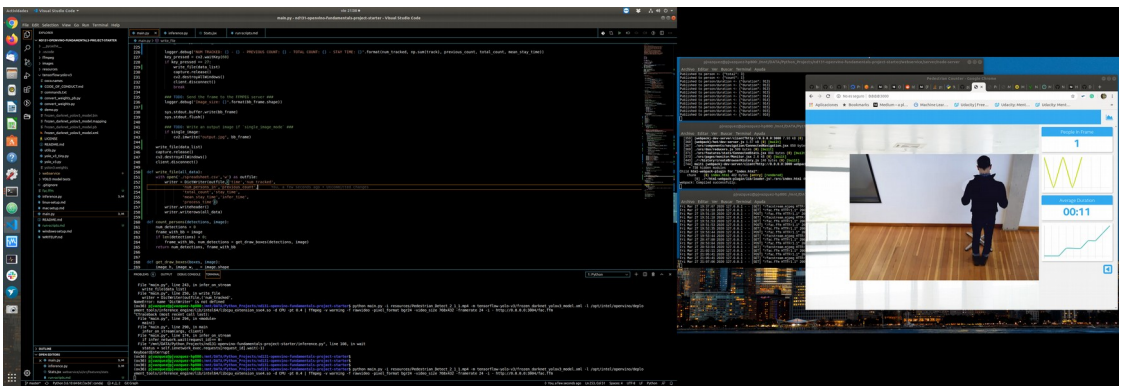
python main.py -i CAM -m tensorflow-yolo-v3/frozen_darknet_yolov3_model.xml -l /opt/intel/openvino/deployment_tools/inference_engine/lib/intel64/libcpu_extension_sse4.so -d CPU -pt 0.6 | ffmpeg -v warning -f rawvideo -pixel_format bgr24 -video_size 640x480 -framerate 24 -i - http://0.0.0.0:3004/fac.ffm

Comparing Model Performance

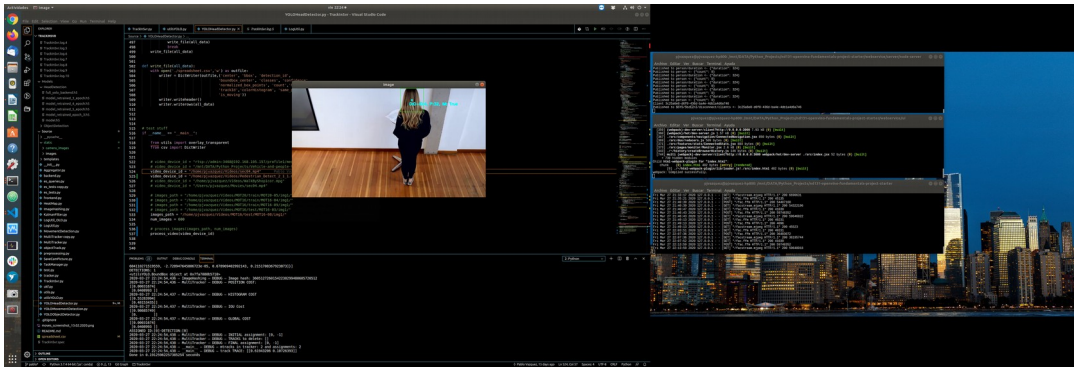
My method(s) to compare models before and after conversion to Intermediate Representations were:

Comparing processing speed in the same environment is usefull to see if 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 gt 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 correctlky detected the 6 persons with an accuracy of 95% and failing only on a small number of frames, YOLO IR loosed 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 usefull 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 a,d 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...