Project Write-Up

Project: Deploy a People Counter App at the Edge
Nanodegree: Intel Edge AI for IoT Developers

Explaining Custom Layers

When converting a model from one particular Deep Learning framework (i.e. TensorFlow, PyTorch, Keras) to the Intel's Intermediate Representation (IR), some layers are not supported by the Intel's Intermediate Representation (IR), due to exotic activation functions, exotic layers, exotic neural architectures, and so on.

Those unsupported layers are then represented as custom layers. Custom layers are made of fine granularity and customized operations, which are carefully described by the IR.

Comparing Model Performance

Now, let's compare models before and after conversion to Intermediate Representations, by using 3 methods:

Accuracy: Both the TensorFlow model (ssd_mobilenet_v2_coco_2018_03_29) and its Intermediate Representation (IR) have similar accuracies, almost the same.

Size: The TensorFlow model (ssd_mobilenet_v2_coco_2018_03_29) has 69,688,296 bytes and its Intermediate Representation (IR) has its Intermediate Representation (IR) has 67,272,876 bytes (frozen_inference_graph.bin) plus 111,552 bytes (frozen_inference_graph.xml), which sum up 67,384,428 bytes.

Inference time: The Intermediate Representation (IR) is 27% faster than the TensorFlow model (ssd_mobilenet_v2_coco_2018_03_29).

Computing in edge devices is a way to save costs and computations of cloud services, which are expensive. Edge devices are cheaper and more abundant than cloud services.

Assess Model Use Cases

Potential Use Cases of the People Counter App	Usefulness
Alarm to protect private places	It's cheaper than hiring too many security guards.
Social Distancing Edge App	Due to the Covid19 pandemia, social distancing is encouraged and this kind of app could help monitoring social distancing. So that, owners of businesses could take actions in case social distancing is not respected.
People counter for street cameras	It's a way to measure how many people are in the streets. If implemented in many street cameras of a city, it could help to create a heat map of the most populated areas.
People counter for businesses	For a franchise, this could help to determine which places are more lucrative.

Assess Effects on End User Needs

Differences in lighting, camera focal length, image size, camera's angle and perspective can produce undesired effects on a deployed edge model.

In my opinion, the Intel's Intermediate Representation (IR) person-detection-retail-0013 is more accurate and robust to variances than the TensorFlow model ssd_mobilenet_v2_coco_2018_03_29.

Regarding lighting, too dark and too bright images are prone to be misclassified. Outliers are always a problem.

Different camera focal length and camera distortion can affect the quality of images. Some objects can appear unfocused due to bad cameras or distorted at the corners of videos.

Obviously the image size and the camera's resolution plays a key role. If the resolution is low, object recognition is complicated. And if the resolution is too high and the hardware is too slow, performance problems and glitches can arise.

Weird camera's angles and perspectives can deteriorate the accuracy in the recognition of objects.

Model Research

I tried 2 models.

First, I perfectly programmed the Intel's Intermediate Representation (IR) person-detection-retail-0013, whose accuracy is pretty high: 88.62%
Here is its website:
https://docs.openvinotoolkit.org/2018_R5/_docs_Retail_object_detection_pedestrian_rmnet_ssd_0013_caffe_desc_person_detection_retail_0013.html
I downloaded its files by using the Unix script:

copy-intel-model.sh
cd /opt/intel/openvino/deployment_tools/tools/model_downloader python downloader.py --name person-detection-retail-0013 -o /home/workspace/

Then, I read the project's rubric at: https://review.udacity.com/#!/rubrics/2773/view
There I noticed that we should not use an Intel's Intermediate Representation (IR). That's why I decided to use the TensorFlow model ssd_mobilenet_v2_coco_2018_03_29, whose website is:
http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_coco_2018_03_29.tar.gz

I downloaded and transformed the TensorFlow model ssd_mobilenet_v2_coco_2018_03_29 by using the Unix script:

download-and-transform-model.sh
rm -r models mkdir models cd models wget http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_coco_2018_03_29.tar.gz

```
tar xvzf ssd_mobilenet_v2_coco_2018_03_29.tar.gz
cd ssd_mobilenet_v2_coco_2018_03_29

python /opt/intel/openvino/deployment_tools/model_optimizer/mo_tf.py --input_model frozen_inference_graph.pb --tensorflow_object_detection_api_pipeline_config pipeline.config --tensorflow_use_custom_operations_config /opt/intel/openvino/deployment_tools/model_optimizer/extensions/front/tf/ssd_v2_support.json --reverse_input_channel
```

Surprisingly, both models work well with the same code. I didn't modify the code and both models work well. Switching models is done by only changing the model parameter in the Unix commands:

```
python main.py -i resources/Pedestrian_Detect_2_1_1.mp4 -m intel/person-detection-retail-0013/FP32/person-detection-retail-0013.xml -l /opt/intel/openvino/deployment_tools/inference_engine/lib/intel64/libcpu_extension_sse4.so -d CPU -pt 0.25 | ffmpeg -v warning -f rawvideo -pixel_format bgr24 -video_size 768x432 -framerate 24 -i - http://0.0.0.0:3004/fac.ffm
```

```
python main.py -i resources/Pedestrian_Detect_2_1_1.mp4 -m models/ssd_mobilenet_v2_coco_2018_03_29/frozen_inference_graph.xml -l /opt/intel/openvino/deployment_tools/inference_engine/lib/intel64/libcpu_extension_sse4.so -d CPU -pt 0.25 | ffmpeg -v warning -f rawvideo -pixel_format bgr24 -video_size 768x432 -framerate 24 -i - http://0.0.0.0:3004/fac.ffm
```

However, the accuracy of the Intel's Intermediate Representation (IR) person-detection-retail-0013 is much better than the accuracy of the TensorFlow model ssd_mobilenet_v2_coco_2018_03_29.

Screenshot of my Web Application