Project Write-Up

Command used for conversion :

python3 mo_tf.py --input_model /home/sallecom/partagewindows/inference_yolo/model/frozen_darknet_yolov3_model_1.pb

--tensorflow_use_custom_operations_config /opt/intel/openvino/deployment_tools/model_optimizer/extensions/front/tf/yolo_v3.json

--batch 1

--output_dir /home/base-marine/inference

Link to yolov3 model : https://github.com/mystic123/tensorflow-yolo-v3.git for conversion from pre-trained weights to checkpoint

Explaining Custom Layers

The openvino toolkit supports many neural network layers implementation in different supported frameworks. However there are some layers which are not supported. To handle this issue, custom layer implementation can be helpfull.

To implement custom layer, we need to provide different files for the model optimizer and the inference engine. At the level of model optimizer, we have to tell openvino how to convert the unsupported layer inside The intermediate representation and at the level of inference engine we have to acctualy provide the c++ implementation of the custum layer.

Using the model extension generator, we have to follow three (3) steps :

1. Generate custom layer template file with model extension generator

2. Edit template files to create the specific custom layer extension source code

3. Specify the location of custom layers extensions files, to be used by model optimizer and inference engine.

thee model extension generator, generates four(4) extensions, two(2) for inference engine and two(2) for model optimizer.

for model optimizer :

1. custom layer extractor: identify the custum layeroperation and extract parameters of each instance of the custom layer

2. custom layer operation : Specifies the attributes that are supported by the custom layer and computes the output shape for each instance of the custom layer from its parameters

for inference engine:

1. custom layer CPU extenxion which are compiled shared library (.so or .dll binary) needed by the CPU Plugin for executing the custom layer on the CPU.

2. custom layer GPU extension which are OpenCL source code (.cl) for the custom layer kernel that will be compiled to execute on the GPU along with a layer description file (.xml) needed by the GPU Plugin for the custom layer kernel

Comparing Model Performance

For this comparison, i used a yolov3 tensorflow checkpoint file. I used it with opencv and then after conversion, with openvino. I stored at each run, the system stats (CPU usage, memory, latency...) in order to compare the two run.

I Didn't noticed a real difference between pre and post conversion. I can say openvino is more efficient in detection.

Comparison of yolov3 implemented with openvino and opencv

Model/toolkit	Size before conversion	Size after conversion	CPU usage	Memory usage	Latency
Yolov3 with openvino	236 Mo	236 Mo	185%	19%	2686 ms
Yolov3 with opencv	236 Mo	-	263%	23%	3047 ms

The size of the model pre- and post-conversion remain the same for yolov3 model. However, at runtime, the size of the model in the memory is 12% less in memory than the opencv implementation. The CPU load is 14% less than the opencv implementation.

The inference time of the model pre- and post-conversion was 3047 ms before conversion and 2686 ms after conversion, then an efficiency of 11% in the openvino implementation ...

Assess Model Use Cases

Some of the potential use cases of the people counter app are : Counting voters during a poll, control attendance of class at school, of people at hospital and administration office. It can be used to monitor attendance of hight security facility.

Each of these use cases would be useful because of the security contexte and metrics needed to guide policy and rules makers at school, hospital and administration. It can be also usefull to monitor people in the contexte of COVID-19, whether the are following barriers gestures or social distanciation.

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 these are as follows...

lighting the model gives a fast inference but a lost of accuracy as observed with tiny yolov3 which gave 266 ms of inference time compare to 2686 ms for yolov3. But it fails to detect

correctly throughtout the video. The big image size will necessite more computing ressource due to the resolution. Due to the fact that some pre-trained model resize input images, it can have lost of accuracy if the input image is too big. So we need a model with larger image input size requiering a device with mor computing power.