**Instructions to Create a Docker Image with OpenVINO integration with TensorFlow**

**Creating the Dockerfile**

Use Ubuntu 18.04 as a base image by adding the line below to the top of the dockerfile.

FROM ubuntu:18.04 as tf\_builder

Then, add the lines below to update the packages, install the prerequisites.

RUN apt-get update \

&& apt-get -y upgrade \

&& apt-get -y install python3 python3-pip python3-venv git curl wget

Upgrade pip3, install TensorFlow 2.4.1, OpenVINO integration, and additional requirements for the demos by adding the lines below.

RUN pip3 install pip==21.0.1 \

&& pip3 install tensorflow==2.4.1 \

&& pip3 install openvino-tensorflow \

&& pip3 install pillow \

&& pip3 install opencv-python-headless \

&& pip3 install virtualenv

Finally, clone the OpenVINO integration with TensorFlow source code from the GitHub repo and download required models for the demos.

RUN git clone https://github.com/openvinotoolkit/openvino\_tensorflow.git \

&& cd openvino\_tensorflow \

&& git checkout notebook\_demo\_branch \

&& curl -L "https://storage.googleapis.com/download.tensorflow.org/models/inception\_v3\_2016\_08\_28\_frozen.pb.tar.gz" | tar -C ./examples/data -xz \

&& cd examples \

&& chmod +x convert\_yolov3.sh \

&& ./convert\_yolov3.sh \

&& chmod +x convert\_yolov3\_160.sh \

&& ./convert\_yolov3\_160.sh

**Building&Running the Docker Image**

Save the file as “Dockerfile” and build the docker image using the command below.

docker build -f Dockerfile -t ubuntu1804\_ovtf .

Then, run the image and activate bash.

docker run -it ubuntu1804\_ovtf:latest /bin/bash

**Testing the Docker Image and OpenVINO integration Installation**

Once the image is running, start a python interactive session to test OpenVINO integration.

python3

Once you are in the python interactive session, import TensorFlow and OpenVINO integration.

import tensorflow

import openvino\_tensorflow

After importing the modules, test OpenVINO integration installation by printing the OpenVINO integration version using the command below.

print(openvino\_tensorflow.\_\_version\_\_)

The output should look like this:

OpenVINO integration with TensorFlow version: b'0.5.0'

nGraph version used for this build: b'0.0.0+b33800a'

TensorFlow version used for this build: v2.4.0-49-g85c8b2a817f

CXX11\_ABI flag used for this build: 0

OpenVINO integration with TensorFlow built with Grappler: False

**Running The Classification Example**

Go into the “openvino\_tensorflow” directory first:

cd /openvino\_tensorflow

Then execute the classification example on the sample image:

python3 examples/classification\_sample.py --image=examples/data/grace\_hopper.jpg

The output should look like below.

Inference time in ms: 13.30

military uniform 0.7839222

mortarboard 0.027078012

pickelhaube 0.014100778

bearskin 0.010530521

bulletproof vest 0.009587267

Now, execute the same example on stock TensorFlow by disabling OpenVINO integration with TensorFlow:

python3 examples/classification\_sample.py --image=examples/data/grace\_hopper.jpg --disable\_ovtf

The output should look like below.

Inference time in ms: 49.78

military uniform 0.7839225

mortarboard 0.027078073

pickelhaube 0.014100777

bearskin 0.010530526

bulletproof vest 0.009587179

Please note the inference time with and without OpenVINO integration with TensorFlow.

**Running The Object Detection Example**

Go into the “openvino\_tensorflow” directory first:

cd /openvino\_tensorflow

Then execute the classification example on the sample image:

python3 examples/object\_detection\_sample.py --image=examples/data/grace\_hopper.jpg

The output should look like below.

Inference time in ms: 60.76

The output image showing the detected objects is written into “detections.jpg”.

Now, execute the same example on stock TensorFlow by disabling OpenVINO integration with TensorFlow:

python3 examples/object\_detection\_sample.py --image=examples/data/grace\_hopper.jpg --disable\_ovtf

The output should look like below.

Inference time in ms: 77.44

Similarly, the output image showing the detected objects is written into “detections.jpg”.